2000

Investigation of the effectiveness of interface constraints in the solution of hyperbolic second-order differential equations

Paul Jerome Silva

Follow this and additional works at: http://scholarworks.lib.csusb.edu/etd-project

Part of the Geometry and Topology Commons

Recommended Citation
http://scholarworks.lib.csusb.edu/etd-project/1953

This Thesis is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
INVESTIGATION OF THE EFFECTIVENESS OF INTERFACE CONSTRAINTS IN THE SOLUTION OF HYPERBOLIC SECOND-ORDER DIFFERENTIAL EQUATIONS

A Thesis
Presented to the faculty of
California State University,
San Bernardino

In Partial Fulfillment of the Requirements for the Degree
Master of Arts
in
Mathematics

by
Paul Jerome Silva
September 2000
INVESTIGATION OF THE EFFECTIVENESS OF INTERFACE
CONSTRAINTS IN THE SOLUTION OF HYPERBOLIC SECOND-ORDER
DIFFERENTIAL EQUATIONS

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino

by
Paul Jerome Silva
September 2000

Approved by:

Dr. Chetan Prakash,
Committee Chair

Dr. Charles Stanton,
Committee Member

Dr. John Sarli,
Committee Member

Dr. Peter Williams, Chair
Department of Mathematics

Dr. Terry Hallett,
Graduate Coordinator
Department of Mathematics

September 8, 2000
Date
Solutions to differential equations describing the behavior of physical quantities (e.g., displacement, temperature, electric field strength) often only have a finite range of validity over a subdomain. Interest beyond the subdomain often arises. As a result, the problem of making the solution compatible across the connecting subdomain interfaces must be dealt with.

Four different compatibility methods are examined here for hyperbolic (time varying) second-order differential equations. These methods are used to match two different solutions, one in each subdomain along the connecting interface. The entire domain that is examined here is a unit square in the Cartesian plane.

The four compatibility methods examined are: point collocation; optimal least square fit; penalty function; Ritz-Galerkin weak form. Discretized L2 convergence is used to examine and compare the effectiveness of each method.
# TABLE OF CONTENTS

Abstract .......................................................................................................................... iii

List of Figures .................................................................................................................. vi

List of Tables ................................................................................................................... vii

1.0 Introduction ............................................................................................................. 1

2.0 Field Equations ...................................................................................................... 4

3.0 Description of Field Variable Expansions and Their Incorporation into the $J_1, J_2$, and $J_3$ Functionals

3.1 Classical Solution, Region $\tau_1$ ................................................................. 12

3.1.1 Mass Matrix Formulation ................................................................. 14

3.1.2 Stiffness Matrix Formulation ......................................................... 15

3.2 Finite Element Method, Region $\tau_2$ ...................................................... 16

3.2.1 Mass Matrix Formulation ................................................................. 21

3.2.2 Stiffness Matrix Formulation ......................................................... 22

3.2.3 Forcing Matrix Formulation ............................................................ 24

3.3 Incorporation of Field Variable Expansion into $J_3$ ........................................ 25

3.3.1 Penalty Function Matrix Formulation ........................................... 26

3.3.2 Ritz-Galerkin Matrix Formulation ............................................... 32
LIST OF FIGURES

Figure 1. Generalized Geometry ........................................ 5
Figure 2. Specific Domain .................................................. 13
Figure 3. General Domain with Finite Elements ..................... 17
Figure 4. Three Node Linear Element ................................. 18
Figure 5. A typical element k that lies on the interface ............ 29
Figure 6. Outward normal Vector on an edge for a typical element k ................................................................. 33
Figure 7. Collocation of points Between $\tau$, and $\tau_H$ ............ 46
Figure 8. Geometry model used to analyze results .................. 65
Figure 9. L2 Comparison, Four Methods,
Resolution 8:4:5 .................................................................. 77
Figure 10. L2 Comparison, Four Methods,
Resolution 12:6:5 .............................................................. 78
Figure 11. L2 Comparison, Four Methods,
Resolution 16:8:5 .............................................................. 79
Figure 12. L2 Comparison, Four Methods,
Resolution 16:8:5 (20 sec) ................................................ 80
Figure 12a. L2 Comparison, Four Methods,
Resolution 16:8:5 (50 sec) ................................................ 81
Figure 13. L2 Error, Ritz-Galerkin for Various Resolutions ........................................ 82

Figure 14. L2 Error, Penalty Function for Various Resolutions ........................................ 83

Figure 15. L2 Error, Least Squares for Various Resolutions ........................................ 84

Figure 16. L2 Error, Collocation for Various Resolutions ........................................ 85

Figure 17. L2 Comparisons for Various $\mu$ ........................................ 86

Figure 18. L2 Comparisons for Various $\mu$ (continued) ........................................ 87

Figure 19. Field Variable vs. Time Pt. (.75,.25) Resolution 8:4:5 ........................................ 88

Figure 20. Field Variable vs. Time Pt. (.75,.25) Resolution 12:6:5 ........................................ 89

Figure 21. Field Variable vs. Time Pt. (.75,.25) Resolution 16:8:5 ........................................ 90

Figure 22. Field Variable vs. Time Pt. (.5,.5) Resolution 8:4:5 ........................................ 91

Figure 23. Field Variable vs. Time Pt. (.5,5) Resolution 16:8:5 ........................................ 92

Figure 24. Field Variable vs. Time Pt. (.25,.75) Resolution 8:4:5 ........................................ 93

Figure 25. Field Variable vs. Time Pt. (.25,.75) Resolution 16:8:5 ........................................ 94

Figure 26. Field Variable vs. Time Pt. (.25,.75) Resolution 8:4:5, $\mu = 100$ ........................................ 95

Figure 27. Time Variance of Load Point ........................................ 96
Figure 28. L2 Comparisons with point load error
excluded .................................................. 97

Figure 29. L2 Comparison w/pt load and
w/o pt. load .................................................. 98
LIST OF TABLES

Table I. Comparison of the First Four Eigenvalues ....... 76
1 Introduction

Second-order field equations have a range of applications that include acoustics, propagation of waves through elastic media, propagation of electro-magnetic waves, and incompressible fluid flow, just to name a few. The solutions to differential equations often only have a finite range of validity. The range of validity is called a sub-domain. Interest beyond the sub-domain often arises. Methods are employed to make the solution compatible across sub-domain interfaces. In creating sub-domain interfaces, however, artificial boundaries are introduced. As a result, spurious reflections arise which reduce the effectiveness of the solution.

Before further proceeding, brief definitions of field variable, classical solution and, the finite element method are in order. The field variable is the dependent variable of the field equation. It is this variable, when determined, that represents the solution to the field equation. A classical solution is a closed form solution of the field equation. The finite element method, which will later be discussed in much greater detail, involves dividing up a domain into small subregions known as
elements which can be triangular or quadrilateral in shape. Because of the smallness of the elements a simple interpolating polynomial can be used to predict the variation of the field variable within it. At the vertices of each element lies a node variable which represents the value of the field variable at its location. Energy functionals (defined later) are discretized for each element and then assembled throughout the region where elements reside in a matrix equation. During the assembly phase constraints are imposed by equating nodal values of elements that share the same nodes. Once the assembly is complete a single global functional is created that is in matrix form. The variation of the global functional is then performed to give the final discretized form. The resulting matrix is then solved by whatever appropriate technique is applicable. The resulting solution yields the values of the nodal variables.

Several finite element methods designed to make the solution compatible across sub-domain interfaces have been employed in various applications. Some of these compatibility methods will be formulated and discussed here. They include:
1) point collocation
2) least square error fit
3) Ritz-Galerkin/ Lagrange Multiplier (it will be shown these two methods are the same).
4) Penalty function method (this will be referred to as $\mu$-method for brevity)

The formulation of each method and how it is implemented into the solution will be discussed.

In this project, each of the above methods will be applied to linear second-order hyperbolic (time varying) differential equations in 2D space. Two sub-domains, which make up an entire domain, will be investigated. The field variable of one sub-domain is described by the solution of a classical problem (e.g., solution to the Helmholtz Equation) and is continuous throughout its sub-domain. The other sub-domain consists of a discretized solution that is obtained by the finite element method. For simplicity, boundary and initial conditions are assumed homogenous. A point impulse in the finite element domain will be used to initiate the excitation.
The presentation of this project is as follows. The field equations and their associated energy functionals, for each compatibility method, are presented first. Field variable expansions for each sub-domain are then described. Incorporation of the expansions into the energy functionals (for each method) then follows. Application of Hamilton's principle is then employed to yield a system of linear differential equations, which are decoupled and solved via modal analysis. A comparison of the effectiveness of each method is then presented for three test cases by way of the discretized $L_1$ method. A description of the finite element method that is employed here is also described.

2 Field Equations

On domains with simply shaped boundaries, e.g., squares and circles, a classical solution may be employed. On domains with irregular boundaries the finite element is suitable as classical, closed-form solutions are not available. A domain divided into two sub-domains, defined $\tau_1$ and $\tau_2$ respectively, is presented below in Figure 1.
\( \Gamma \) and \( \Gamma_{II} \) have respective boundaries \( \sigma_1 \) and \( \sigma_2 \), and are separated by the interface \( \Gamma \). Let \( v = v(x,y,t) \) represent the field variable in the domain \( \Gamma_I \), and \( u = u(x,y,t) \) represent the field variable in \( \Gamma_{II} \). The field equations on \( \Gamma_I \) and \( \Gamma_{II} \) are then

\[
\nabla^2 v - \frac{\partial^2 v}{\partial t^2} = 0 \quad \text{in} \quad \Gamma_I, \quad v(x,y,t) = 0 \quad \text{on} \quad \sigma_1 \quad (1)
\]

\[
\nabla^2 u - \frac{\partial^2 u}{\partial t^2} = f(x,y,t) \quad \text{in} \quad \Gamma_{II}, \quad u(x,y,t) = 0 \quad \text{on} \quad \sigma_2 \quad (2)
\]

\[
u = v \quad \text{on} \quad \Gamma \quad (3)
\]

where

\[
\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}
\]
We have chosen the boundary conditions to be homogeneous on $\sigma_1$ and $\sigma_2$. The excitation term is, for the purposes of this thesis
\[
f(x,y,t) = F_0 \delta(x-\xi, y-\zeta) \, e^{-\alpha t}, \text{ for } t > 0
\]
where $F_0$ and $\alpha$ are constants and $\delta(x-\xi, y-\zeta)$ is the Dirac delta function applied at the point $(\xi,\zeta)$ in the Cartesian plane. The coordinates $(\xi,\zeta)$ lie in $\tau_n$ for this application. The field variable functions $v(x,y,t)$ and $u(x,y,t)$ are twice differentiable in $\tau_1$ and $\tau_2$ respectively, and once differentiable on $\sigma_1 \cup \Gamma$ and $\sigma_2 \cup \Gamma$ respectively. The forcing function $f(x,y,t)$ is, in general, piecewise continuous in $\tau_n$.

**Theorem 1.**

Assume $v(x,y,t)$, $u(x,y,t)$, and $f(x,y,t)$ satisfying the following conditions
\[
\bar{D}_1 = \tau_1 + \sigma_1 + \Gamma, \quad \bar{D}_2 = \tau_2 + \sigma_2 + \Gamma,
\]
\[
S_1 = \{ v \in C^1[\bar{D}_1] : v|_{\sigma_1} = 0; v|_{\Gamma} = u|_{\Gamma} \} \forall x,y \in \bar{D}_1, t \in [0,1],
\]
\[
S_2 = \{ u \in C^1[\bar{D}_2] : u|_{\sigma_2} = 0; u|_{\Gamma} = v|_{\Gamma} \} \forall x,y \in \bar{D}_2, t \in [0,1],
\]
$v \in S_1 \cap C^2[\tau_1]$, $u \in S_2 \cap C^2[\tau_2]$, and $f \in C^0[\tau_n]$

Let
\[
J_I = \int_{t_1}^{t_2} \left\{ \frac{1}{2} \int_{\Omega} \dot{v}^2 \, dx \, dy - \frac{1}{2} \int_{\partial \Omega} \nabla v \cdot \nabla \dot{v} \, dx \, dy \right\} \, dt, \quad \text{where} \quad \dot{v} = \frac{\partial v}{\partial t} \quad (5)
\]

\[
J_u = \int_{t_1}^{t_2} \left\{ \frac{1}{2} \int_{\Omega} \dot{u}^2 \, dx \, dy - \frac{1}{2} \int_{\partial \Omega} (\nabla u \cdot \nabla u - 2u f) \, dx \, dy \right\} \, dt, \quad \text{where} \quad \dot{u} = \frac{\partial u}{\partial t} \quad (6)
\]

with \( \nabla = \frac{\partial}{\partial x} i + \frac{\partial}{\partial y} j \)

And let the interface functional \( J_I \) be chosen, in the various methods as:

\[
J_r = 0 \quad \text{Point Collocation}
\]

\[
J_r = 0 \quad \text{Least square fit}
\]

\[
J_r = \frac{1}{2} \mu \int_{t_1}^{t_2} \left\{ \int_{\Gamma} (u - v)^2 d\Gamma \right\} dt, \quad \mu \text{ method} \quad (7)
\]

\[
J_r = \int_{t_1}^{t_2} \left\{ - \int_{\Gamma} (u - v) \left( \frac{\partial u}{\partial n} - \frac{\partial v}{\partial n} \right) d\Gamma \right\} dt, \quad \text{Ritz Galerkin} \quad (8)
\]

Then the differential equations (1) and (2) can be obtained as Euler-Lagrange equations by setting the variation of the dual energy functionals (5) and (6) to zero, i.e., by employing Hamilton's principle.
Remark. In the μ-method, as μ → ∞, it is expected that u → v along Γ, thus meeting our objective, namely satisfying the interface constraint. While the first two methods do not explicitly contain a Jᵢ term, constraint equations are imposed. A description of these constraints and how they are implemented will be discussed later.

Proof of Theorem 1. We claim that the variation δJ₁ = 0 yields equation (1), δJₙ = 0 yields equation (2), and δJᵢ = 0 yields the interface condition.

The proof is carried out for each of the penalty function and the Ritz Galerkin methods.

Part 1. μ method (penalty function)

Let us allow arbitrary variations δv and δu, in the functions u and v respectively. Then the condition that the total variation in J, i.e., ΔJ = ΔJ₁ + ΔJₙ + ΔJᵢ, is zero gives, after integrating by parts
Since $\delta v$ is arbitrary in $\tau_i$ and $\delta u$ is arbitrary in $\tau_i$,

\[ \nabla^2 v - \frac{\partial^2 v}{\partial t^2} = 0 \text{ in } \tau_i \quad \text{and} \quad \nabla^2 u - \frac{\partial^2 u}{\partial t^2} = f(x,y,t) \text{ in } \tau_i, \]

equations (1) and (2). This proves the duality between the functionals and the interior equations. The following steps prove the duality between the functionals and the interface conditions.

We have that $\delta v = 0$ on $\sigma_1$, and $\delta u = 0$ on $\sigma_2$ since $v$ and $u$ are specified on $\sigma_1$ and $\sigma_2$ respectively. This leads to

\[
\int_{\Gamma} \frac{\partial v}{\partial n} \delta v d\Gamma = \int_{\Gamma} \frac{\partial v}{\partial n} \delta v d\Gamma + \int_{\Gamma} \frac{\partial v}{\partial n} \delta v d\Gamma = \int_{\Gamma} \frac{\partial v}{\partial n} \delta v d\Gamma,
\]

similarly \[
\int_{\Gamma} \frac{\partial u}{\partial n} \delta u d\Gamma = \int_{\Gamma} \frac{\partial u}{\partial n} \delta u d\Gamma
\]

Thus we have remaining

\[
\int_{\Gamma} \frac{\partial v}{\partial n} \delta v d\Gamma + \int_{\Gamma} \frac{\partial u}{\partial n} \delta u d\Gamma + \mu \int_{\Gamma} (u-v) \delta (u-v) d\Gamma = 0
\]
The first two integrals are finite. Letting \( \mu \) approach infinity would force the condition \( u = v \) on \( \Gamma \), thus satisfying the interface condition. This also leads to \( \delta u = \delta v \) on \( \Gamma \) giving

\[
\int_r \left( \frac{\partial v}{\partial n} + \frac{\partial u}{\partial n} \right) \delta u d\Gamma = 0 \quad \text{or} \quad \int_r \left( \frac{\partial v}{\partial n} + \frac{\partial u}{\partial n} \right) \delta v d\Gamma = 0
\]

Since either \( u \) or \( v \) alone are arbitrary on \( \Gamma \) we have that \( \frac{\partial v}{\partial n} + \frac{\partial u}{\partial n} = 0 \) or \( \frac{\partial v}{\partial n} = -\frac{\partial u}{\partial n} \) across \( \Gamma \).

**Part 2. Ritz-Galerkin Weak Form**

The proof here is similar for that above except after showing the conditions in \( \tau_1, \tau_\|, \sigma_1 \& \sigma_2 \) are met, we have remaining,

\[
\int_r \frac{\partial v}{\partial n} \delta v d\Gamma + \int_r \frac{\partial u}{\partial n} \delta u d\Gamma - \int_r (u-v) \delta \left( \frac{\partial v}{\partial n} - \frac{\partial u}{\partial n} \right) d\Gamma - \int_r \delta (u-v) \left( \frac{\partial v}{\partial n} - \frac{\partial u}{\partial n} \right) d\Gamma = 0
\]

The third integral above vanishes on \( \Gamma \) since \( \frac{\partial v}{\partial n} - \frac{\partial u}{\partial n} \) is arbitrary on \( \Gamma \). So \( u \rightarrow v \), thus leading to the satisfaction of the interface condition. This further leads to

\[
\int_r \frac{\partial v}{\partial n} \delta v d\Gamma + \int_r \frac{\partial u}{\partial n} \delta u d\Gamma - \int_r \frac{\partial u}{\partial n} \delta u d\Gamma - \int_r \frac{\partial v}{\partial n} \delta v d\Gamma + \int_r \frac{\partial v}{\partial n} \delta v d\Gamma + \int_r \frac{\partial u}{\partial n} \delta u d\Gamma = 0
\]
The first and second integrals cancel the third and fourth integrals, respectively.

Accounting for \( u = v \) on \( \Gamma \), this yields the condition

\[
\frac{\partial v}{\partial n} = -\frac{\partial u}{\partial n}
\]

across \( \Gamma \) by the same reasoning as for the penalty function case.

**Theorem 2.** The Ritz-Galerkin and the Lagrangian multiplier method yield the same equations.

**Proof:**

Lagrange:

\[
I = \int_{\Omega} \left\{ \frac{1}{2} \int (v^2 - \nabla v \cdot \nabla v) \, dx \, dy + \frac{1}{2} \int (u^2 - \nabla u \cdot \nabla u + 2 uf) \, dx \, dy - \int \lambda (u - v) \, d\Gamma \right\} \, dt
\]

Galerkin:

\[
J = \int_{\Omega} \left\{ \frac{1}{2} \int (v^2 - \nabla v \cdot \nabla v) \, dx \, dy + \frac{1}{2} \int (u^2 - \nabla u \cdot \nabla u + 2 uf) \, dx \, dy - \int \left( \frac{\partial u}{\partial n} - \frac{\partial v}{\partial n} \right) (u - v) \, d\Gamma \right\} \, dt
\]

Comparing, we see that \( \lambda = \frac{\partial u}{\partial n} - \frac{\partial v}{\partial n} \) in either case.
3 Description of Field Variable Expansions and Their Incorporation into the $J_I$, $J_{II}$, and $J_I$ Functionals

3.1 Classical Solution, Region $\tau_I$

The classical solution as applied to the region $\tau_I$ is described by a series that is chosen to exactly satisfy the boundary conditions on $\sigma_I$.

$$v(x,y) = \sum A_i \psi_i(x,y) = \Psi(x,y)A$$  \hspace{1cm} (9)

where $\Psi$ and $A$ are column matrices representing the shape functions $\psi_i$ and the coefficients $A_i$, respectively. The coefficients $A_i$ are functions of time and the $\psi_i(x,y)$ are spacial functions that can take several forms by choice. They can be eigenfunction solutions to the Helmholtz equation such as trigonometric terms of Bessel functions, bi-harmonic polynomials, or simple monomial terms.

Substituting Eq.(9) into Eq.(5) the energy functional $J_I$ takes the form

$$J_I = \int_{t_1}^{t_2} \left\{ \frac{1}{2} \dot{A}^T \left[ \int_{\tau_I} \nabla \psi \cdot \nabla \psi^T dx dy \right] \dot{A} - A^T \left[ \int_{\tau_I} \nabla \psi \cdot \nabla \psi^T dx dy \right] A \right\} dt$$  \hspace{1cm} (10)
For this particular application we will choose a rectangular region as our $\tau_i + \tau_n$ domain as shown in figure 2.

Eigenfunction solutions to the Helmoltz Equation to be used in Eq. (9) take the form

$$v(x, y, t) = \sum_{n=1}^{n_{\text{max}}} \sum_{m=1}^{m_{\text{max}}} A_{mn}(t) \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right)$$

(11a)

$$\Psi^T(x, y)A(t)$$

(11b)

where

$$\Psi^T(x, y) = \{\psi_{11}(x, y), \psi_{12}(x, y), \ldots, \psi_{mn}(x, y), \ldots\}$$

(11b')

$$A^\tau(t) = \{A_{11}, A_{12}, \ldots, A_{mn}, \ldots\}$$

and

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

(11c)

Note that each of these functions exactly satisfy homogenous boundary conditions in Fig.1, i.e., $\psi_{mn}(x, y) = 0$ on
the boundary of the rectangle. In order that the coefficients $A_{mn}$ and the spacial functions $\psi_{mn}(x,y)$ be expressed in the form of (9) the double subscripts $m$ & $n$ must be put in terms of a single subscript, i.e.,

$$v(x,y,t) = \sum_{k=1}^{K_{\text{max}}} A_k(t) \psi_k(x,y) = A^T \Psi$$ \hspace{1cm} (12)

Thus a one-to-one correspondence must exist between the subscript $k$ in Eq.(12) and each pair $m$ & $n$ in Eq.(11c), that is,

$$(m,n) \rightarrow k$$

One such mapping is

$$k = n_{\text{max}}(m-1) + n$$ \hspace{1cm} (13)

where the upper limit of the sum in Eq.(12) is $K_{\text{max}} = m_{\text{max}} n_{\text{max}}$

Specific examples of this mapping are illustrated in Appendix A.

### 3.1.1 Mass Matrix Formulation

The expansion of Eq.(11a) is substituted into Equation (5) to obtain the mass and stiffness matrix formulations. We will first obtain the mass matrix by considering the first spacial integral on the right hand side of Eq.(9')
Substituting the specific terms of Eq. (11c) for \( \Psi \), individual matrix elements of Eq (14) become

\[
M_{ij} = \int_{\Omega} \Psi \Psi^T dx dy
\]  

where from Eq. (13), \( i = m_{\max} (n-1) + m, \quad j = q_{\max} (p-1) + q \),

and \( m_{\max} = q_{\max} \).

3.1.2 Stiffness Matrix Formulation

Formulation of the stiffness matrix is carried out as follows. Eq. (11c) is substituted into the second integral of Eq. (5) which yields the second spatial integral of Eq. (10) where

\[
\nabla \Psi^T = \begin{bmatrix}
\frac{\partial \psi_1}{\partial x} i + \frac{\partial \psi_1}{\partial y} j, \frac{\partial \psi_2}{\partial x} i + \frac{\partial \psi_2}{\partial y} j, ..., \frac{\partial \psi_0}{\partial x} i + \frac{\partial \psi_0}{\partial y} j
\end{bmatrix}
\]

and where \( Q = K_{\max} \).
Substituting the explicit terms of Eq. (11c) into Eq. (16) gives the individual matrix elements of $K_i$

$$K_{ij} = \iint_{\Omega} \left( \frac{\partial \psi_i}{\partial x} \frac{\partial \psi_j}{\partial x} + \frac{\partial \psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} \right) \, dx \, dy$$

$$= \iint_{\Omega} \left[ \frac{\pi^2}{a^2} \cos \left( \frac{n \pi x}{a} \right) \cos \left( \frac{p \pi x}{a} \right) \sin \left( \frac{m \pi y}{b} \right) \sin \left( \frac{q \pi y}{b} \right) + \right.$$  

$$+ \frac{\pi^2}{b^2} \sin \left( \frac{n \pi x}{a} \right) \sin \left( \frac{p \pi x}{a} \right) \cos \left( \frac{m \pi y}{b} \right) \cos \left( \frac{q \pi y}{b} \right) \right] \, dx \, dy$$  (17)

Here again we define $i = m_{\text{max}}(n-1)+m$, $j = q_{\text{max}}(p-1)+q$ and $m_{\text{max}} = q_{\text{max}}$. The RHS of Eqs (15) and (17) are integrated numerically using Gauss quadrature.

### 3.2 Finite Element Method, Region $\tau_{ii}$

The finite element method involves discretization of a general domain (i.e., $\tau_{ii}$ in this case) into sub-regions sufficiently small such that the field variable can be described by a simple polynomial expansion regardless of the complexity of the general domain. This is illustrated in figure 3 below.
The term $\Delta_k$ represents the kth finite element, i.e., $\tau_{II} = \bigcup \Delta_k$.

We define the vector of coefficients $u^{(k)} = (u_1^{(k)}, u_2^{(k)}, u_3^{(k)})$, which represents the field variable value at the nodes (located at the triangle's vertices) of element $\Delta_k$; these are functions of time. The field variable over the general finite element domain can be described in terms of natural triangular coordinates $\eta_1, \eta_2, \eta_3$ (defined below; see Fig.4) in each element

$$u(x, y, t) = \sum u^{(k)} T \phi = u^{(1)} T \phi^{(1)} + u^{(2)} T \phi^{(2)} + \ldots + u^{(n)} T \phi^{(n)} \text{ on } \tau_{II} \quad (18)$$

where $u^{(k)} T = (u_1^{(k)}, u_2^{(k)}, u_3^{(k)})$, $\phi^{(0)} = 0$ on $\Delta_k \forall j \neq k$, and $\phi^{(k)} = [\eta_1, \eta_2, \eta_3]$. The $\phi^{(0)}$ are called shape functions.
Over a typical element $\Delta_x$, the field variable is defined as the $k$th term in equation (18)

\[
u^{(k)} = \mathbf{u}_i^{(k)} \phi_i^{(k)}
\]

\[
= (u_1^{(k)}, u_2^{(k)}, u_3^{(k)} \left( \eta_1, \eta_2, \eta_3 \right)^{(k)T}
\]

\[
= u_1 \eta_1 + u_2 \eta_2 + u_3 \eta_3
\]

Three Node Linear Element

Figure 4

The "natural" triangular coordinates of a point $(x, y)$ inside the element are interpolation polynomials (see Fig.4) defined by

\[
\eta_1 = a_1 x + b_1 y + c_1
\]

\[
\eta_2 = a_2 x + b_2 y + c_2
\]

\[
\eta_3 = a_3 x + b_3 y + c_3
\]
where \( a_1 = \frac{y_2 - y_3}{2A_T} \), \( a_2 = \frac{y_2 - y_1}{2A_T} \), \( a_3 = \frac{y_1 - y_2}{2A_T} \)

\[ b_1 = \frac{x_3 - x_2}{2A_T}, \quad b_2 = \frac{x_1 - x_3}{2A_T}, \quad b_3 = \frac{x_2 - x_1}{2A_T} \]

\[ c_1 = \frac{x_2 y_3 - x_3 y_2}{2A_T}, \quad c_2 = \frac{x_3 y_1 - x_1 y_3}{2A_T}, \quad c_3 = \frac{x_1 y_2 - x_2 y_1}{2A_T} \] (21)

and where \( A_T = \frac{[y_1(x_3 - x_2) + y_2(x_1 - x_3) + y_3(x_2 - x_1)]}{2} \) which is the area of the triangular finite element. Eqs.(20) and (21) arise from the following:

\[
0 \leq \eta_k \leq 1 \quad \text{k=1,2,3 and } \eta_1 + \eta_2 + \eta_3 = 1
\]

\[
x = \eta_1 x_1 + \eta_2 x_2 + \eta_3 x_3 \] (22a)
\[
y = \eta_1 y_1 + \eta_2 y_2 + \eta_3 y_3
\]

Note also that

\[
\eta_1 = 0 \text{ on side 2-3}
\]
\[
\eta_2 = 0 \text{ on side 1-3}
\]
\[
\eta_3 = 0 \text{ on side 1-2} \] (22b)

\[
\eta_1 = 1, \quad \eta_2 = 0, \quad \eta_3 = 0 \text{ at node 1}
\]
\[
\eta_1 = 0, \quad \eta_2 = 1, \quad \eta_3 = 0 \text{ at node 2}
\]
\[
\eta_1 = 0, \quad \eta_2 = 0, \quad \eta_3 = 1 \text{ at node 3}
\]
Note that at node 1, \( u^{(1)} = (u_1, 0, 0) \), \( x = x_1 \), and \( y = y_1 \), and similarly for nodes 2 and 3. Note also that Equations (20) are the inverse of Equations (22). The element just described is a linear three node triangular element. Its interpolation shape function is a linear polynomial. A more complex, but more flexible six node quadratic element could also be used.

The finite element expansion of Eq. (19a) is substituted into the energy terms of Eq. (6) for each element. This leads to the formulation of matrices for kinetic energy, elastic potential energy, and the work due to the external forcing function for each element. The energy terms are summed for all elements to get the total energy over \( \tau_n \). This procedure is known as element assembly. During the assembly procedure, nodes that share the same location from adjacent elements are constrained to have the same field variable value. For a simple example of element assembly, see Appendix B. We now implement the above procedure.

Substitution of Eq. (19a) into Eq. (6) gives
To obtain the total energy of the region $\tau_n$, the energies of all the elements are added together. Thus for a region containing $N$ finite elements the total energy would be

$$J_{II} = \sum_{k=1}^{N} J_{II}^{(k)}$$

### 3.2.1 Mass Matrix Formulation

The integrand of the first integral in Eq. (23) represents the mass matrix for an element $\Delta_k$ in $\tau_{II}$ which is expressed as

$$M_{II}^{(k)} = \int_{\Delta_k} \phi^{(k)} \phi^{(k)T} \, dx \, dy$$

Substituting Eq. (19b) in Eq. (25) gives
\[ M_{II}^{(k)} = \int_{\Delta_k} \begin{bmatrix} \eta_1^{(k)} \\ \eta_2^{(k)} \\ \eta_3^{(k)} \end{bmatrix} \begin{bmatrix} \eta_1 & \eta_2 & \eta_3 \end{bmatrix} \, dA \]

\[ = \int_{\Delta_k} \begin{bmatrix} \eta_1^2 & \eta_1 \eta_2 & \eta_1 \eta_3 \\ \eta_2 \eta_1 & \eta_2^2 & \eta_2 \eta_3 \\ \eta_3 \eta_1 & \eta_3 \eta_2 & \eta_3^2 \end{bmatrix} \, dA = \frac{A_\tau^{(k)}}{3} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \quad (26) \]

where \( A_\tau^{(k)} \) is the area of element \( \Delta_k \), and use was made of the formula

\[ \int \eta_1^\alpha \eta_2^\beta \eta_3^\gamma \, dA = \frac{\alpha! \beta! \gamma!}{(\alpha + \beta + \gamma)!} A_\tau \]

### 3.2.2 Stiffness Matrix Formulation

The second integral in Eq. (23), or the "stiffness matrix", is discretized as follows. This matrix is expressed as

\[ K_{II}^{(k)} = \left[ \int \nabla \phi^{(k)} \cdot \nabla \phi^{(k)^T} \, dx \, dy \right] \quad (27) \]

where
\[ \nabla \phi^{(k)} = \begin{bmatrix} \nabla \eta_1^{(k)} \\ \nabla \eta_2^{(k)} \\ \nabla \eta_3^{(k)} \end{bmatrix} = \begin{bmatrix} \nabla \eta_1^{(k)} \\ \nabla \eta_2^{(k)} \\ \nabla \eta_3^{(k)} \end{bmatrix} = \begin{bmatrix} \frac{\partial \eta_1}{\partial x} + \frac{\partial \eta_1}{\partial j} \\ \frac{\partial \eta_2}{\partial x} + \frac{\partial \eta_2}{\partial j} \\ \frac{\partial \eta_3}{\partial x} + \frac{\partial \eta_3}{\partial j} \end{bmatrix} \tag{28} \]

Substituting Eq. (28) into (27) gives

\[ K_{\Pi}^{(k)} = \int_{A_k} \nabla \phi^{(k)T} \cdot \nabla \phi^{(k)} \, dA = \]

\[ = \int_{A_k} \begin{bmatrix} \left( \frac{\partial \eta_1}{\partial x} \right)^2 + \left( \frac{\partial \eta_1}{\partial y} \right)^2 & \frac{\partial \eta_1}{\partial x} \frac{\partial \eta_2}{\partial x} + \frac{\partial \eta_1}{\partial y} \frac{\partial \eta_2}{\partial y} & \frac{\partial \eta_1}{\partial x} \frac{\partial \eta_3}{\partial x} + \frac{\partial \eta_1}{\partial y} \frac{\partial \eta_3}{\partial y} \\ \frac{\partial \eta_2}{\partial x} \frac{\partial \eta_1}{\partial x} + \frac{\partial \eta_2}{\partial y} \frac{\partial \eta_1}{\partial y} & \left( \frac{\partial \eta_2}{\partial x} \right)^2 + \left( \frac{\partial \eta_2}{\partial y} \right)^2 & \frac{\partial \eta_2}{\partial x} \frac{\partial \eta_3}{\partial x} + \frac{\partial \eta_2}{\partial y} \frac{\partial \eta_3}{\partial y} \\ \frac{\partial \eta_3}{\partial x} \frac{\partial \eta_1}{\partial x} + \frac{\partial \eta_3}{\partial y} \frac{\partial \eta_1}{\partial y} & \frac{\partial \eta_3}{\partial x} \frac{\partial \eta_2}{\partial x} + \frac{\partial \eta_3}{\partial y} \frac{\partial \eta_2}{\partial y} & \left( \frac{\partial \eta_3}{\partial x} \right)^2 + \left( \frac{\partial \eta_3}{\partial y} \right)^2 \end{bmatrix} \, dx\,dy \]

Eq. (29)

where we see that each individual matrix element is given by

\[ K_{\Pi}^{(k)} \left[ \begin{array}{c} i, j \end{array} \right] = \frac{\partial \eta_i}{\partial x} \frac{\partial \eta_j}{\partial x} + \frac{\partial \eta_i}{\partial y} \frac{\partial \eta_j}{\partial y}, \quad i=1, 2, 3 \quad \& \quad j=1, 2, 3 \]
In order to get an explicit expression for the matrix elements of Eq.(29) it is necessary to differentiate Equations (20) with respect to x and y.

\[
\begin{align*}
\frac{\partial \eta_1}{\partial x} &= a_1, & \frac{\partial \eta_2}{\partial x} &= a_2, & \frac{\partial \eta_3}{\partial x} &= a_3, \\
\frac{\partial \eta_1}{\partial y} &= b_1, & \frac{\partial \eta_2}{\partial y} &= b_2, & \frac{\partial \eta_3}{\partial y} &= b_3,
\end{align*}
\]

Substituting these expressions into Eq.(29) finally gives

\[
\begin{bmatrix}
2 + b_1^2 & a_1a_2 + b_1b_2 & a_1a_3 + b_1b_3 \\
a_2a_1 + b_2b_1 & a_2^2 + b_2^2 & a_2a_3 + b_2b_3 \\
a_3a_1 + b_3b_1 & a_3a_2 + b_3b_2 & a_3^2 + b_3^2
\end{bmatrix}^\text{(k)}
\]

\[ (k) \]

3.2.3 Forcing Matrix Formulation

The forcing function for this thesis resides only in the finite element region and will be applied only to a single node, the selection of which will be determined later. The discretized force matrix for a typical element is obtained from substituting (19b) into the third integral.
of the RHS of (6)

\[ f^{(k)} = \int f(x,y,t) \phi^{(k)} \, dx \, dy = \int f(x,y,t) \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{bmatrix} \, dx \, dy \]  

(31)

Substituting Eq (4) into (31) gives

\[ f^{(k)} = F_0 \int_{A_i} \delta(x-\xi,y-\zeta) e^{-at} \begin{bmatrix} \eta_1^{(k)} \\ \eta_2 \\ \eta_3 \end{bmatrix} \, dx \, dy \]  

(32)

Next substituting (20) into (32), and utilizing the definition of the Dirac delta function, yields

\[ f^{(k)} = F_0 e^{-at} \int_{A_i} \begin{bmatrix} a_1 \xi + b_1 \zeta + c_1^{(k)} \\ a_2 \xi + b_2 \zeta + c_2 \\ a_3 \xi + b_3 \zeta + c_3 \end{bmatrix} \, dx \, dy \]  

(33)

Where \( \xi \) and \( \zeta \) are the point of application of the force in the Cartesian plane.

3.3 Incorporation of Field Variable Expansions into \( J \).

There are two sets of matrix formulations that are derived in this section. The first involves the derivation of the compatibility matrices for the penalty function
method by the substitution of the field variable definitions into the integrals that define the penalty function method. Following this will be the derivation of compatibility matrices for the Ritz-Galerkin method.

### 3.3.1 Penalty Function Matrix Formulation

The energy functional that establishes the interface compatibility between regions $\tau_i$ and $\tau_n$ for the penalty function method is given by equation (7) and is restated here as

$$J_r = \frac{1}{2} \mu \int_{t_i}^{t_n} \{ \int_r (u - v)^2 \, d\Gamma \} \, dt \quad (34)$$

This integral is a contour integral performed along the interface $\Gamma$, which is shown in Fig. (3). First, the integrand in (34) is expanded, i.e.,

$$J_r = \frac{1}{2} \mu \int_{t_i}^{t_n} \int_r (u^2 - uv - vu + v^2) \, d\Gamma$$

$$= \frac{1}{2} \mu \int_{t_i}^{t_n} \left[ u^2 \, d\Gamma - \mu \int_r (vu + uv) \, d\Gamma + \frac{1}{2} \mu \int_r v^2 \, d\Gamma \right] \, dt \quad (35)$$

For convenience, we introduce the definitions
\[ \Pi_1 = \frac{1}{2} \int_{t_1}^{t_2} \left[ \mu \int_{\Gamma} v^2 d\Gamma \right] dt \] 
(36a)

\[ \Pi_2 = -\frac{1}{2} \int_{t_1}^{t_2} \left[ \mu \int_{\Gamma} (vu + uv) d\Gamma \right] dt \] 
(36b)

\[ \Pi_3 = \frac{1}{2} \int_{t_1}^{t_2} \left[ \mu \int_{\Gamma} u^2 d\Gamma \right] dt \] 
(36c)

So that

\[ J_r = \Pi_1 + \Pi_2 + \Pi_3 \]

The integral in (36a) involves only the classical solution. The integrals in (36b) and (36c) involve finite elements so that these matrices will be developed on the element level. Substituting eq.(9) into (36a) gives

\[ \Pi_1 = \int_0^L A^T \left[ \mu \int_{\Gamma} \psi \psi^T d\Gamma \right] A \] 
(37)

Here we define

\[ K_1 = \left[ \mu \int_{\Gamma} \psi \psi^T d\Gamma \right] \] 
(38)

After substituting Eq.(11c) into (38) we get
Where as before the indices \(i\) and \(j\) are related to the \(n, m, p, q\) indices by

\[ i = m_{\text{max}}(n-1)+m, \quad j = q_{\text{max}}(p-1)+q, \quad \text{and} \quad m_{\text{max}} = q_{\text{max}} \]

Next, the expansions of Eqs (9) and (19a) are substituted into (36b) to get for an element \(k\) that lies on the interface \(\Gamma\)

\[
\Pi_{2}^{(k)} = -\frac{1}{2} \int_{\Gamma} \{ \mu \mathbf{A}^T [ \int_{\Gamma} \psi \phi^{(k)T} d\Gamma ] \mathbf{u}^{(k)} + \mu \mathbf{u}^{(k)T} [ \int_{\Gamma} \phi^{(k)} \psi^T d\Gamma ] \mathbf{A} \} dt
\]

From which we obtain two matrices for a boundary element \(k\)

\[
K_{2}^{(k)} = -\frac{1}{2} \mu \int_{\Gamma} \psi \phi^{(k)T} d\Gamma \quad (40)
\]

and

\[
K_{3}^{(k)} = -\frac{1}{2} \mu \int_{\Gamma} \phi^{(k)} \psi^T d\Gamma \quad (41)
\]

After substituting the specific expansions of Eq.(11b) and (19b) we obtain
where, for an individual matrix element we get

\[ K_{2_{ij}}^{(k)} = -\frac{1}{2}\mu \int_{\Gamma} \psi_i \eta_j d\Gamma = -\frac{1}{2}\mu \int_{\Gamma} \left( \sin \frac{n\pi x}{a} \sin \frac{m\pi y}{b} \right) \eta_j d\Gamma \]

where \( i = m_{\text{max}} (n-1) + m \) and \( i = 1, \ldots, Q \), \( j = 1, 2, 3 \) and \( Q = m_{\text{max}} n_{\text{max}} \)

Note that in Figure 5 below, the element nodes 1 and 2 lie on the interface. In this case the natural coordinate \( \eta_i \) on \( \Gamma \) is equal to zero. If it were nodes 2 and 3 that lay on the interface, then \( \eta_i \) would equal zero, etc.

A typical element \( k \) that lies on the interface

Figure 5
Since $n_j = a_j^{(k)} x + b_j^{(k)} y + c_j^{(k)}$ for some kth element on $\Gamma$ we can then write

$$K_{ij}^{(k)} = \frac{1}{2} \mu \left( a_j^{(k)} \int_{\Gamma} x \sin \frac{n \pi x}{a} \sin \frac{n \pi y}{b} d\Gamma + b_j^{(k)} \int_{\Gamma} y \sin \frac{n \pi x}{a} \sin \frac{n \pi y}{b} d\Gamma + c_j^{(k)} \int_{\Gamma} \sin \frac{n \pi x}{a} \sin \frac{n \pi y}{b} d\Gamma \right)$$

The factors $a_j^{(k)}, b_j^{(k)},$ and $c_j^{(k)}$ are defined in equation (21) and $x$ and $y$ can be expresses as

$$x = x_1^{(k)} + (x_2 - x_1)^{(k)} \gamma \quad \text{and} \quad y = y_1^{(k)} + (y_2 - y_1)^{(k)} \gamma \quad (41')$$

as referenced to Figure 5. Also, $0 \leq \gamma \leq 1$ over the element edge and is the variable of integration along the interface. By inspection we observe that $K_3^{(k)} = K_2^{(k)} \tau$. Finally, substituting Eq(19a) into (36c) gives us

$$\Pi_3^{(k)} = \frac{1}{2} u^{(k)\tau} \left[ \mu \int_{\Gamma} \phi^{(k)} \phi'^{(k)} \tau d\Gamma \right] u^{(k)}$$

from which we get the matrix
\[ k_4 = \frac{1}{2} \mu \int_\Gamma \phi^{(1)} \phi^{(1)*} \, d\Gamma \]  \hspace{1cm} (42)

Inserting (19b) into (42) yields

\[ k_4 = \frac{1}{2} \mu \int_\Gamma \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{bmatrix} \begin{bmatrix} \eta_1 & \eta_2 & \eta_3 \end{bmatrix} \, d\Gamma \]

An individual element is thus determined from

\[ K^{(k)}_{4ij} = \frac{1}{2} \mu \int_\Gamma \eta_i \eta_j \, d\Gamma = \frac{1}{2} \mu \begin{cases} \frac{\Delta L^{(k)}}{6} & i \neq j \\ \frac{\Delta L^{(k)}}{3} & i = j \end{cases} \]  \hspace{1cm} (43)

where use is made of the formula

\[ \int_0^1 \eta_i \eta_j \, d\Gamma = \Delta L \frac{\alpha! \beta!}{(\alpha + \beta + 1)!}, \quad \text{and} \]

\[ \Delta L^{(k)} = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \]

which is the length of the edge of element k that lies on \( \Gamma \), and \( p \) and \( q \) are the nodes of \( k \) that lie on \( \Gamma \).

We can now determine the entire matrix explicitly.
This particular case applies to Figure (5) since nodes 1 and 2 lie on the interface and therefore natural coordinate \( \eta_1 = 0 \) along \( \Gamma \). If the elements nodes 2 and 3 were to lie on the boundary, then \( \eta_1 = 0 \) along \( \Gamma \) and the matrix would take the form

\[
K_4^{(k)} = \frac{1}{2\mu} \frac{\Delta \tau^{(k)}}{6} \begin{bmatrix}
2 & 1 & 0 \\
1 & 2 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]  

(44a)

and so forth. This completes the definition of the interface compatibility matrices for the penalty function method.

3.3.2 Ritz-Galerkin Matrix Formulation

The integral term that establishes compatibility along the interface that separates the regions \( \tau_1 \) and \( \tau_2 \) is given by equation (8) and is rewritten below
\[ J_r = \int_{t_1}^{t_2} \left\{ - \int_r (u-v) \left( \frac{\partial u}{\partial n} - \frac{\partial v}{\partial n} \right) d\Gamma \right\} dt \quad (45) \]

Before proceeding further, we will at this point define the normal derivatives with respect to \( u \) and \( v \) that appear in equation (45). These derivatives are in the direction outwardly normal to the edge of the element that lies on the interface. Figure 6 illustrates an outward normal vector to an element edge. It makes an angle \( \theta \) to the \( x \) axis as shown in the figure. The \( u_i \) represents the field variable value at node \( i \) and the \((x_i, y_i)\) represents the coordinates of the node \( i \).

![Outward Normal Vector on an edge](image)  
For a typical element \( k \)  

Figure 6
3.3.2.1 Determination of Normal Derivatives

We begin with the chain rule

\[
\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x} \frac{x}{\partial x} + \frac{\partial u}{\partial y} \frac{y}{\partial n}\quad \text{(46)}
\]

where in reference to figure (6) we have that

\[
\frac{x}{\partial n} = \cos \theta \quad \text{and} \quad \frac{y}{\partial n} = \sin \theta\quad \text{(47)}
\]

Recalling (19c) we can differentiate it to obtain

\[
\frac{\partial u}{\partial x} = \frac{\partial \eta_1}{\partial x} u_1 + \frac{\partial \eta_2}{\partial x} u_2 + \frac{\partial \eta_3}{\partial x} u_3\quad \text{(48a)}
\]

and

\[
\frac{\partial u}{\partial y} = \frac{\partial \eta_1}{\partial y} u_1 + \frac{\partial \eta_2}{\partial y} u_2 + \frac{\partial \eta_3}{\partial y} u_3\quad \text{(48b)}
\]

Differentiating Eqs.(20) and substituting into Eqs.(48) we arrive at

\[
\frac{\partial u}{\partial x} = a_1 u_1 + a_2 u_2 + a_3 u_3\quad \text{(49a)}
\]

\[
\frac{\partial u}{\partial y} = b_1 u_1 + b_2 u_2 + b_3 u_3\quad \text{(49b)}
\]

Substituting (47) and (49) into (48) finally yields
\[
\frac{\partial u}{\partial n} = (a_1 \cos \theta + b_1 \sin \theta) u_1 + (a_2 \cos \theta + b_2 \sin \theta) u_2 + (a_3 \cos \theta + b_3 \sin \theta) u_3
\]

\[
= \begin{bmatrix}
(a_1 \cos \theta + b_1 \sin \theta),
(a_2 \cos \theta + b_2 \sin \theta),
(a_3 \cos \theta + b_3 \sin \theta)
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3
\end{bmatrix}
\]

Comparing (50) to the differentiation (19b) with respect to the normal vector, i.e.,

\[
\frac{\partial u}{\partial n} = \begin{bmatrix}
\frac{\partial \eta_1}{\partial n},
\frac{\partial \eta_2}{\partial n},
\frac{\partial \eta_3}{\partial n}
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3
\end{bmatrix}
\]

we note that

\[
\frac{\partial \eta_1}{\partial n} = a_1 \cos \theta + b_1 \sin \theta \quad (51a)
\]

\[
\frac{\partial \eta_2}{\partial n} = a_2 \cos \theta + b_2 \sin \theta \quad (51b)
\]

\[
\frac{\partial \eta_3}{\partial n} = a_3 \cos \theta + b_3 \sin \theta \quad (51c)
\]

We further note, for example, in the Fig.6 on side 1-2,

\[
\cos \theta = \frac{y_2 - y_1}{L_{1-2}} \quad \text{and} \quad \sin \theta = \frac{x_1 - x_2}{L_{1-2}} \quad (51d)
\]

where

\[
L_{1-2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.
\]

These quantities are calculated for each element.
Next we determine the normal derivative with respect to the classical solution \( v(x,y,t) \). Again we begin with the chain rule for a typical term in the expansion

\[
\frac{\partial \psi_{mn}}{\partial n} = \frac{\partial \psi_{mn}}{\partial x} \frac{\partial x}{\partial n} + \frac{\partial \psi_{mn}}{\partial y} \frac{\partial y}{\partial n} = \frac{\partial \psi_{mn}}{\partial x} \cos \theta + \frac{\partial \psi_{mn}}{\partial y} \sin \theta
\]  

(52)

Substituting a specific term, i.e., Eq.(11c) gives

\[
\frac{\partial \psi_{i}}{\partial n} = \frac{\partial \psi_{mn}}{\partial n} = \frac{\pi}{a} \left( \cos \frac{n \pi x}{a} \sin \frac{m \pi y}{b} \right) \cos \theta + \frac{m \pi}{b} \left( \sin \frac{n \pi x}{a} \cos \frac{m \pi y}{b} \right) \sin \theta
\]  

(53)

where as before, we define \( i = m_{\text{max}}(n-1)+m \)

3.3.2.2 Determination of Interface Compatibility Matrices for Ritz-Galerkin

Multiplying out the terms of the integrand of (45), it may be rewritten as

\[
J_r = - \int_{n} \{ \int_{r} \frac{\partial u}{\partial n} d\Gamma - \int_{r} v \frac{\partial u}{\partial n} - \int_{r} u \frac{\partial v}{\partial n} + \int_{r} v \frac{\partial v}{\partial n} \} dt
\]  

(54)

The first three integrals within the brackets of Eq.(54) involve finite element expansions and will be derived at the element level. The fourth integral involves only the
classical solution expansion. Substituting (19a) into the first of these integrals in (54) leads to the matrix for some element k on the interface

\[ K^{(k)} = \int \phi^{(k)} \frac{\partial \phi^{(k)\top}}{\partial n} \, d\Gamma \]  

Substituting the explicit expressions for the finite element expansion, (19b) and (51), into (55) yields, for a particular element of the matrix

\[ K_{ij}^{(k)} = \int \eta_i \frac{\partial \eta_j}{\partial n} \, d\Gamma \]

\[ = \int \eta_i \left( a_j^{(k)} \sin \theta + b_j^{(k)} \cos \theta \right) \, d\Gamma \]

\[ = \frac{\Delta L^{(k)}}{2} \left( a_j^{(k)} \sin \theta + b_j^{(k)} \cos \theta \right) \]  

Substitution of (11b) and (19a) into the second bracketed integral of (54) yields the following matrix

\[ K''^{(k)} = \int \psi \frac{\partial \phi^{(k)\top}}{\partial n} \, d\Gamma \]  

Inserting the specific expressions for \( \Psi \) and \( \phi^{(k)} \) yields the following matrix for an element k on the interface
\[ K_{ij}^{(k)} = \int_{\Gamma} \hat{\rho}_{ij} \frac{\partial \hat{\eta}_{ij}}{\partial \hat{n}} d\Gamma \]

\[ = \int_{\Gamma} \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right) (a_j \cos \theta + b_j \cos \theta) d\Gamma \]

\[ = (a_j \cos \theta + b_j \cos \theta) \int_{\Gamma} \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right) d\Gamma \quad (58) \]

where \( i = m_{\text{max}}(n-1) + m \), and \( j = 1, 2, 3 \). Substitution of (11b) and (19a) into the third bracketed integral of (54) yields the following matrix for an element \( k \)

\[ K'^{(k)} = \int_{\Gamma} \phi^{(k)} \frac{\partial \psi^T}{\partial \hat{n}} d\Gamma \quad (59) \]

Inserting explicit expressions for \( \phi^{(k)} \) and \( \Psi \), from (11c) and (19b) respectively, we obtain the following

\[ K_{ij}^{(k)} = \int_{\Gamma} \hat{\eta}_{ij} \frac{\partial \hat{\eta}_{ij}}{\partial \hat{n}} d\Gamma \]

\[ = \int_{\Gamma} \hat{\eta}_{ij} \left[ \left( \frac{n\pi}{a} \cos \frac{n\pi x}{a} \sin \frac{m\pi y}{b} \right) \cos \theta + \left( \frac{m\pi}{b} \sin \frac{n\pi x}{a} \cos \frac{m\pi y}{b} \right) \sin \theta \right] d\Gamma \quad (60) \]

where the \( \eta_{ij} \) are defined in (21) and (22), \( x \) and \( y \) can be written in terms of \( \Gamma \) as shown in (41'), and \( j = m_{\text{max}}(n-1) + m \), with \( i = 1, 2, 3 \).
Finally, we insert (11b) into the last of the bracketed integrals of Eq(45) obtaining

\[ K^{(iv)} = \int \frac{\partial \psi}{\partial n} \psi^T d\Gamma \]  \hspace{1cm} (61)

By inserting the explicit expressions of (11c) into (61), and making use of (53), we arrive at

\[ K^{(iv)}_{ij} = \int \psi_i \frac{\partial \psi_j}{\partial n} d\Gamma = \int \psi_i \left( \frac{\partial \psi_j}{\partial x} \cos \theta + \frac{\partial \psi_j}{\partial y} \sin \theta \right) d\Gamma \]

\[ = \int \left[ \frac{p\pi}{a} \left( \sin \frac{n\pi x}{a} \cos \frac{p\pi x}{a} \sin \frac{m\pi y}{b} \sin \frac{q\pi y}{b} \right) \cos \theta + \frac{q\pi}{a} \left( \sin \frac{n\pi x}{a} \sin \frac{p\pi x}{a} \cos \frac{m\pi y}{b} \cos \frac{q\pi y}{b} \right) \sin \theta \right] d\Gamma \]  \hspace{1cm} (62)

where again \( i = m_{\text{max}}(n-1)+m, \ j = q_{\text{max}}(p-1)+q \) and \( m_{\text{max}} = q_{\text{max}} \).

Also \( x \) and \( y \) can be defined in terms of the integration variable \( \Gamma \) as shown in (41') for any straight line contour, and \( \cos \theta \), and \( \sin \theta \) are defined in (51d) for the same contour. The matrix elements of (62) are determined numerically.
3.4 Summary of Discretization

All matrices which define the energy functionals of equations (5), (6), (7), and (8) have been defined above. These matrices involving finite elements must be assembled into global matrices that account for all the degrees of freedom represented by all finite element nodes. Part of this procedure involves the summation of all the energy functionals formulated for each finite element. Adjacent elements share nodes, and so the field variables at the common nodes must also share the same values. This makes it necessary to impose inter-element constraints during the assembly process. The resulting assembled matrices will be defined here as global matrices.

The classical solution case does not present an assembly problem. Each term of the classical solution expansion resides throughout the entire region, in this case $\tau$, thus no assembly is necessary. This is unlike the finite element case where expansion terms reside only within their proper finite element thus making assembly necessary. The assembly process as defined below will be simply illustrated as a summation of the energy functionals that are developed at the finite element level without
reference to the inter-element constraints. A simple example that illustrates the entire assembly procedure is presented in Appendix B. The energy functionals, defined in equations (5), (6), (7), and (8), are presented below in matrix form for each compatibility case.

Classical case, region $\tau_1$

$$J_1 = \int_{\tau_1} \left\{ -\frac{1}{2} A^T K_1 A + \frac{1}{2} A^T M_1 A \right\} dt$$  \hspace{1cm} (63)

Finite element case, $N$ elements in $\tau_{II}$

$$J_{II} = \sum_{k=1}^{N} \int_{\tau_k} \left\{ -\frac{1}{2} u^T K_{II} u + \frac{1}{2} \ddot{u}^T M_{II} \ddot{u} + u^T f \right\}^{(k)} dt$$  \hspace{1cm} (64)

Compatibility functional, $\mu$ method with $R$ elements residing on the interface

$$J_\mu = \frac{1}{2} \mu \int_{\tau_1} A^T K_1 A dt - \sum_{k=1}^{R} \frac{1}{2} \mu \int_{\tau_k} \left\{ A^T K_2 u + u^T K_3 A - u^T K_4 u \right\}^{(k)} dt$$  \hspace{1cm} (65)

Compatibility functional, Ritz-Galerkin, with $R$ elements residing on the interface

$$J_R = \sum_{k=1}^{R} \int_{\tau_k} \left\{ u^T K' u - A^T K' u - u^T K'' A \right\}^{(k)} dt + \int_{\tau_1} \left\{ A^T K^{(iv)} A \right\} dt$$  \hspace{1cm} (66)
From (63)-(66), the definition of the complete matrices are as follows

\[ K_1 = \int_{\Gamma_1} \nabla \psi \cdot \nabla \psi^T \, dx \, dy \]  
\[ K_{II} = \sum_{k=1}^{N} \int_{\Gamma_{II}} \nabla \phi^{(k)} \cdot \nabla \phi^{(k)^T} \, dx \, dy = \sum_{k=1}^{N} K_{II}^{(k)} \]  
\[ K_1 = \frac{1}{2} \int_{\Gamma} \psi \psi^T \, d\Gamma \]  
\[ K_2 = \sum_{k=1}^{R} \frac{1}{2} \int_{\Gamma} \phi^{(k)} \phi^{(k)^T} \, d\Gamma = \sum_{k=1}^{R} K_2^{(k)} \]  
\[ K_3 = \frac{1}{2} \sum_{k=1}^{R} \int_{\Gamma} \phi^{(k)} \psi^T \, d\Gamma = \sum_{k=1}^{R} K_3^{(k)} \]  
\[ K_4 = \sum_{k=1}^{R} \frac{1}{2} \int_{\Gamma} \phi^{(k)} \phi^{(k)^T} \, d\Gamma = \sum_{k=1}^{R} K_4^{(k)} \]  
\[ K' = \sum_{k=1}^{R} \int_{\Gamma} \frac{\partial \phi^{(k)}}{\partial n} \phi^{(k)^T} \, d\Gamma = \sum_{k=1}^{R} K'^{(k)} \]  
\[ K'' = \sum_{i=1}^{R} \int_{\Gamma} \frac{\partial \psi}{\partial n} \phi^{(k)^T} \, d\Gamma = \sum_{k=1}^{R} K''^{(k)} \]  
\[ K''' = \sum_{i=1}^{R} \int_{\Gamma} \frac{\partial \phi^{(i)}}{\partial n} \psi^T \, d\Gamma = \sum_{k=1}^{R} K'''^{(k)} \]  
\[ K^{(iv)} = \int_{\Gamma} \frac{\partial \psi}{\partial n} \psi^T \, d\Gamma \]
Note that in the equations where there are sums involved, finite elements are also involved. The summations are not literal sums of the finite element matrices, but are symbolic representations of the element assembly process as was mentioned earlier. With the assembled finite element matrices now defined, we can write equations (63) - (66) without the summation signs. We will go even further, that is, we will now combine all energy functional terms and express them in a compact matrix form showing the matrix representations of both the finite element node vectors, \( \mathbf{u} \), and coefficient vectors, \( \mathbf{A} \), of classical degrees of freedom. For example, \( J_I + J_{II} \) takes the form

\[
J_I + J_{II} = \int \left\{ [\mathbf{A}^T \mathbf{u}] \begin{bmatrix} \mathbf{M}_I & 0 \\ 0 & \mathbf{M}_{II} \end{bmatrix} [\mathbf{u}] - [\mathbf{A}^T \mathbf{u}] \begin{bmatrix} \mathbf{K}_I & 0 \\ 0 & \mathbf{K}_{II} \end{bmatrix} [\mathbf{u}] + 2[\mathbf{A}^T \mathbf{u}] [\mathbf{f}] \right\} dt
\]  

and \( J_f \) takes the form
\[ J_\Gamma = \frac{1}{2} \mu \int \left\{ [A^T \ u_\Gamma] \begin{bmatrix} K_1 & -K_2 \\ -K_3 & K_4 \end{bmatrix} \begin{bmatrix} A \\ u_\Gamma \end{bmatrix} \right\} dt \]  \hspace{1cm} \mu \text{ method} \hspace{1cm} (81)

\[ J_\Gamma = \int \left\{ [A^T \ u_\Gamma] \begin{bmatrix} -K^{(iv)} & K'' \\ K' & -K' \end{bmatrix} \begin{bmatrix} A \\ u_\Gamma \end{bmatrix} \right\} dt \]  \hspace{1cm} \text{Ritz-Galerkin} \hspace{1cm} (82)

where \( u_\Gamma \) represents only the finite element nodes that lie on the interface. In each case the total energy is expressed as

\[ J = J_1 + J_{II} + J_\Gamma \]  \hspace{1cm} (82')

Henceforth, the following definitions are established

\[ K = \begin{bmatrix} K_I & 0 \\ 0 & K_{II} \end{bmatrix} \]  \hspace{1cm} (83)

\[ M = \begin{bmatrix} M_I & 0 \\ 0 & M_{II} \end{bmatrix} \]  \hspace{1cm} (84)

\[ K_\Gamma = \begin{bmatrix} K_1 & -K_2 \\ -K_3 & K_4 \end{bmatrix} \]  \hspace{1cm} (85)

\[ K'_\Gamma = \begin{bmatrix} -K^{(iv)} & K'' \\ K''' & -K' \end{bmatrix} \]  \hspace{1cm} (86)
4.0 Imposition of constraints for Collocation and Least Square Method

The collocation and least square methods do not alter the form of the total energy formulation in the way the penalty function and the Ritz Galerkin methods do. Instead, constraints are enforced at the interface $\Gamma$. The procedures are described below.

4.1 Collocation

The goal here is to establish a constraint matrix relationship between the classical region $\tau_1$ and the finite element region $\tau_\Pi$ via the collocation method. This method involves equating the field variable expansions of adjacent sub-domains at their common interface. That is, the values at each finite element field variable node that lies on the interface $\Gamma$ is set equal to the value of the classical solution when it is evaluated at the same point. The idea is illustrated in Figure 7. The constraint matrix will then be inserted into the discretized energy functional of $J_1 + J_\Pi$.

Evaluating the classical solution at a point $p$, from (11a) gives
\[ v(x_p, y_p, t) = \sum_{n=1}^{n_{\text{max}}} \sum_{m=1}^{m_{\text{max}}} A_{nm}(t) \sin\left(\frac{n\pi x_p}{a}\right) \sin\left(\frac{m\pi y_p}{b}\right) \] (87)

where \( p \) ranges from 1 to \( R \).

Collocation of Points Between \( \tau_1 \) and \( \tau_\Pi \)

Figure 7

At the nodal point \( p \), the field variable has a value of \( u_p \). This is set equal to the classical solution value at the same location, i.e., \( u_p = v(x_p, y_p, t) \) on \( \Gamma \). Applying (87) we get

\[ u_p = \sum_{n=1}^{n_{\text{max}}} \sum_{m=1}^{m_{\text{max}}} A_{nm}(t) \sin\left(\frac{n\pi x_p}{a}\right) \sin\left(\frac{m\pi y_p}{b}\right) \]
To make the formulation more compact we apply Eq. (11b) to get

\[ u_p = \Psi'(x_p, y_p)A(t) \quad (88) \]

We see that a row matrix is formed from (88), namely \( \Psi'(x_p, y_p) \) for the node p. Utilizing (11c), a particular matrix element can be found from

\[ \psi_{mn}(x_p, y_p) = \sin \frac{n\pi x_p}{a} \sin \frac{m\pi y_p}{b} \]

For the node p-1 another row matrix is formed and so forth. Eq. (88) is repeated for each node on the interface, thus forming a matrix equation

\[
\mathbf{u}_r = \begin{bmatrix} : \\ u_{p-1} \\ u_p \\ u_{p+1} \\ : \end{bmatrix} = \begin{bmatrix} : \\ \Psi^T(x_{p+1}, y_{p+1})A(t) \\ \Psi^T(x_p, y_p)A(t) \\ \Psi^T(x_{p+1}, y_{p+1})A(t) \\ : \end{bmatrix} = \mathbf{C} \mathbf{A}(t) \quad (89)
\]

Which in specific terms this becomes, utilizing (11a)
where \( \mathbf{u}_r \) is the finite element degrees of freedom on \( \Gamma \), and applying (11b')

\[
\mathbf{C} = \begin{bmatrix}
\psi_T(x_{p+1}, y_{p+1}) \\
\psi_T(x_p, y_p) \\
\psi_T(x_{p+1}, y_{p+1}) \\
\vdots 
\end{bmatrix} = \begin{bmatrix}
\psi_1(x_{p-1}, y_{p-1}), \psi_2(x_{p-1}, y_{p-1}), \ldots, \psi_0(x_{p-1}, y_{p-1}) \\
\psi_1(x_p, y_p), \psi_2(x_p, y_p), \ldots, \psi_0(x_p, y_p) \\
\psi_1(x_{p+1}, y_{p+1}), \psi_2(x_{p+1}, y_{p+1}), \ldots, \psi_0(x_{p+1}, y_{p+1}) \\
\vdots 
\end{bmatrix}
\]

Utilizing Eq. (11c), we get for a typical matrix element in \( \mathbf{C} \)

\[
C_{ij} = \sin\left(\frac{n\pi x_i}{a}\right)\sin\left(\frac{m\pi y_i}{b}\right)
\]

where for each \( mn \) pair we have \( j = m_{\text{max}}(n-1) + m \)
The constraint equation, relating the coefficients $A$ with the interface finite element nodes, becomes

$$u_r = CA$$

The finite element interface vector, $u_r$, is replaced by $CA$ in the energy formulation. If we let $u_a$ be the vector of finite element nodes that do not lie on the interface, the total finite element vector can be expressed as

$$u = \begin{bmatrix} u_r \\ u_a \end{bmatrix} = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A \\ u_a \end{bmatrix}$$

The complete solution vector, that includes all degrees of freedom, now becomes

$$w = \begin{bmatrix} A \\ u_r \\ u_a \end{bmatrix} = \begin{bmatrix} I & 0 \\ C & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A \\ u_a \end{bmatrix} = C^*w_R \quad (89a)$$

where

$$C^* = \begin{bmatrix} I & 0 \\ C & 0 \\ 0 & I \end{bmatrix}, \quad \text{and} \quad w_R = \begin{bmatrix} A \\ u_a \end{bmatrix} \quad (89b)$$
The vector $w$ represents the total number of unconstrained degrees of freedom, and $w_r$ is the constrained solution vector.

Finally substituting (89a) into (81) we get

$$J = \frac{1}{2} \int \left\{ w_r^T C^* \begin{bmatrix} M & 0 \\ 0 & M_{\text{II}} \end{bmatrix} C^* w_r - w_r^T C^* \begin{bmatrix} K & 0 \\ 0 & K_{\text{II}} \end{bmatrix} C^* w_r - 2w_r^T C^* f \right\} dt$$

$$\Rightarrow J = \frac{1}{2} \int \left\{ \dot{w}_r^T M^* \dot{w}_r - w_r^T K^* w_r - 2w_r^T f^* \right\} dt$$

where we set

$$M^* = C^* \begin{bmatrix} M & 0 \\ 0 & M_{\text{II}} \end{bmatrix} C^*, \quad K^* = C^* \begin{bmatrix} K & 0 \\ 0 & K_{\text{II}} \end{bmatrix} C^*, \quad f^* = C^* f$$

The reduced solution vector $w_r$ has $(Q + m)$ number of degrees of freedom $Q + m$, and $C^*$ is a $(Q + L) \times (Q + m)$ matrix with $m$ being the number of finite element degrees of freedom not on $\Gamma$, and $L$ being the total number of finite element degrees of freedom. $Q$ was defined previously to equal the number of degrees of freedom contributed by the classical solution. Note: the length of $u_r$ is equal to $L-m$. 

50
4.2 Least Square Method

The least square error method is formulated by integrating the square of the difference of the field variables of the adjacent regions on the adjoining interface $\Gamma$

$$S^2 = \int_{I_h} \left\{ \int_{\Gamma} (u - v)^2 d\Gamma \right\} dt$$

$$= \int_{I_h} \left\{ A^T K_1 A - A K_2 u_r - u_r^T K_3 A + u_r^T K_4 u_r \right\} dt$$

Two alternatives are possible here. We may either let $S^2$ vary with respect to $u_r$, while holding $A$ fixed, and then solve for $u_r$ in terms of $A$, or we may vary $A$ with while holding $u_r$ fixed and then solve for $A$ in terms of $u_r$. Summarizing the two alternatives:

1) Differentiate with respect to $u_r$, hold $A$ constant, solve for $u_r$ in terms of $A$ thus constraining $u_r$ out of the formulation.

2) Differentiate with respect to $A$, holding $u_r$ constant, solving for $A$ in terms of $u_r$ thus constraining $A$ out of the formulation.
Alternative 1:

Variation of $S^2$ with respect to $u_r$ gives

$$\delta u_r^*(K_4 + K_2^r) u_r - \delta u_r^* K_1 A - \delta u_r^* K_2 A = 0$$

leading to

$$u_r = (K_4 + K_2^r)^{-1} (K_1 + K_3) A = C_1 A$$

where $C_1 = (K_4 + K_2^r)^{-1} (K_1 + K_3)$, and the unconstrained degree of freedom vector is given from

$$w = \begin{bmatrix} A \\ u_r \\ u_m \end{bmatrix} = \begin{bmatrix} A \\ C_1 0 \\ 0 I \end{bmatrix} \{w_r \} = C' w_r \quad (89c)$$

Substitution of (89c) into (81) gives us

$$J = \int \left\{ \frac{1}{2} w'^T C' \begin{bmatrix} M_I & 0 \\ 0 & M_{II} \end{bmatrix} C' w_r - \frac{1}{2} w'^T C' \begin{bmatrix} K_I & 0 \\ 0 & K_{II} \end{bmatrix} C' w_r - w'^T C^r f' \right\} dt$$

$$\Rightarrow J = \frac{1}{2} \int \left\{ \dot{w}_r^T M' \dot{w}_r - \dot{w}_r^T K' \dot{w}_r - w_r^T f' \right\} dt$$

where $w_r$ is defined in (89b), and

$$M' = C'^T \begin{bmatrix} M_I & 0 \\ 0 & M_{II} \end{bmatrix} C' \quad , \quad K' = C'^T \begin{bmatrix} K_I & 0 \\ 0 & K_{II} \end{bmatrix} C', \quad f' = C'^T f$$

52
Alternative 2:

Variation of $S^2$ with respect to $A$ gives

$$
\delta u_r^T K^z A + \delta u_r^T K^v A - \delta u_r^T (K^z + K^v) u_r = 0
$$

giving

$$
A = (K^z + K^v)^{-1} (K^z + K^v) u_r = C_2 u_r
$$

Where $C_2 = (K^z + K^v)^{-1} (K^z + K^v)$. The unconstrained degree of freedom vector now becomes

$$
w = \begin{bmatrix} \mathbf{A} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{u}_r \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{u}_s \end{bmatrix} = \mathbf{C}'' \mathbf{w}_r
$$

(89d)

Substitution of (89d) into (81) yields

$$
J = \int \left\{ \frac{1}{2} \mathbf{w}_r^T \mathbf{C}''^T \begin{bmatrix} \mathbf{M}^I & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^II \end{bmatrix} \mathbf{C}'' \mathbf{w}_r - \frac{1}{2} \mathbf{w}_r^T \mathbf{C}''^T \begin{bmatrix} \mathbf{K}^I & \mathbf{0} \\ \mathbf{0} & \mathbf{K}^II \end{bmatrix} \mathbf{C}'' \mathbf{w}_r - \mathbf{w}_r^T \mathbf{C}''^T \mathbf{f} \right\} \, dt
$$

$$
\Rightarrow J = \frac{1}{2} \int \left\{ \mathbf{w}_r^T \mathbf{M}'' \mathbf{w}_r - \mathbf{w}_r^T \mathbf{K}'' \mathbf{w}_r - \mathbf{w}_r^T \mathbf{f}'' \right\} \, dt
$$

where $\mathbf{w} = \mathbf{C}'' \mathbf{w}_r$ and $\mathbf{w}_r$ is defined in (89b). Further

$$
\mathbf{M}'' = \mathbf{C}''^T \begin{bmatrix} \mathbf{M}^I & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^II \end{bmatrix} \mathbf{C}'', \quad \mathbf{K}'' = \mathbf{C}''^T \begin{bmatrix} \mathbf{K}^I & \mathbf{0} \\ \mathbf{0} & \mathbf{K}^II \end{bmatrix} \mathbf{C}'', \quad \mathbf{f}'' = \mathbf{C}'' \mathbf{f}
$$
Alternative 1 will be chosen for the purposes of this thesis. The inverse of the matrix $K_i + K_i^2$, needed to obtain $C_2$, is sometimes singular.
5. Discretized System of Differential Equations

The discretized energy of the entire system takes the form, for the different methods:

\[
J = \begin{cases} 
\frac{1}{2} \int_{t_1}^{t_2} \{\dot{w}^T M \dot{w} - w^T K w + 2w^T f\} dt & \text{Collocation} \\
\frac{1}{2} \int_{t_1}^{t_2} \{\dot{w}^T M \dot{w} - w^T K w + 2w^T f\} dt & \text{Least Squares (alt 1)} \\
\frac{1}{2} \int_{t_1}^{t_2} \{\dot{w}^T M \dot{w} - w^T K w + 2w^T f - w^T K_r w\} dt & \text{Ritz Galerkin} \\
\frac{1}{2} \int_{t_1}^{t_2} \{\dot{w}^T M \dot{w} - w^T K w + 2w^T f - \mu(w^T K_r w)\} dt & \mu \text{ method}
\end{cases}
\]

where \( M = \begin{bmatrix} M_I & 0 \\ 0 & M_{II} \end{bmatrix} \), \( K = \begin{bmatrix} K_I & 0 \\ 0 & K_{II} \end{bmatrix} \), \( f = \begin{bmatrix} 0^{0x1} \\ f^{1x1} \end{bmatrix} \)

and \( K_r \) in its unexpanded form is defined by Eq.(85). When expanded to global dimensions its dimensions becomes \((Q+L) x (Q+L)\). The expanded form of \( K_r \) appears as

\[
K_r = \begin{bmatrix} K_1 & -K_2 & 0 \\ -K_3 & K_4 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

The zeros are placeholders for the finite element nodes that do not lie on \( \Gamma \). Similarly for the Ritz-Galerkin method.
Performing $\delta J = 0$ yields the following systems of linear second order differential equations for each of the four methods discussed.

\[
\mathbf{K}_r' = \begin{bmatrix}
-K^{(iv)} & K^* & 0 \\
K^{**} & -K' & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\mathbf{M}'\ddot{\mathbf{w}} + \mathbf{K}'\dot{\mathbf{w}} = \mathbf{f}(t) \quad \text{(Collocation)}
\]

\[
\mathbf{M}'\ddot{\mathbf{w}} + \mathbf{K}'\dot{\mathbf{w}} = \mathbf{f}(t) \quad \text{(Least Square)}
\]

\[
\mathbf{M}\dot{\mathbf{w}} + (\mathbf{K} + \mathbf{K}_r')\mathbf{w} = \mathbf{f}(t) \quad \text{(Ritz - Galerkin)}
\]

\[
\mathbf{M}\dot{\mathbf{w}} + (\mathbf{K} + \mu\mathbf{K}_r)\mathbf{w} = \mathbf{f}(t) \quad \text{(\(\mu\)-method)}
\]

It is a necessary condition that to avoid a trivial solution when applying the \(\mu\)-method, \(\mathbf{K}_r\) must be singular.

For the formulation above this is guaranteed since \(\mathbf{K}_r\) will contain rows and columns with all zeros. This is a consequence of the fact that only a few of the elements in \(\mathbf{\tau}_H\) will lie on \(\Gamma\). Should \(\mathbf{K}_r\) be non-singular, letting \(\mu \to \infty\) would cause \(\mu\mathbf{K}_r\) to dominate the system and thus lead to a trivial solution.
5.1 Modal Analysis Formulation

For discussion purposes in this section we will refer to any of the systems of equations above generically as

\[ M\ddot{w} + Kw = f(t) \]  

A modal analysis will be performed on this system to decouple the equations. Each of the decoupled equations are then solved. Quantified descriptions are given below.

The eigenvalue for the above system is formed by assuming a solution \( w = se^{-\omega t} \) and substituting this into the above equation, after setting \( f = 0 \), to get

\[-\omega^2Mse^{-\omega t} + Kse^{-\omega t} = 0\]

After factoring out the exponential term we get

\[(K - \lambda M)s = 0\]

where we let \( \lambda = \omega^2 \). The next step is to perform

\[ \text{det}(K - \lambda M) = 0 \]

yielding a polynomial equation in \( \lambda \). The order of the polynomial equation is determined by the order of the matrices \( K \) and \( M \). The solution to the equation above
determines the natural frequencies of vibration of the system. Each eigenvalue solution \( \lambda_i \) to the polynomial equation has an associated eigenvector. The eigenvectors in this case determine the mode shapes of their respective frequencies, i.e.,

\[
(K - \lambda_i M) s_i = 0
\]

where \( s_i \) is the eigenvector, or mode shape associated with the \( \lambda_i \) th eigenvalue, from which the associated natural frequency is determined by setting

\[
\omega_i = \sqrt{\lambda_i} \text{ radians / sec or } \nu_i = \frac{\omega_i}{2\pi} \text{ in cycles / sec or Hertz}
\]

### 5.2 Solutions

The eigenvectors play a very important role in facilitating the solution of the system of differential equations. Each eigenvector is first normalized with respect to the mass matrix

\[
s'_i = \frac{s_i}{\sqrt{s^T_i Ms_i}}
\]

The modal matrix is then formed from the columns of the normalized eigenvectors
\[ S = [s_1 | s_2 | \cdots | s_n] \]

Through the modal matrix we are able to relate the solution vector \( w \) with a set of generalized coordinates \( \delta \)

\[ w = S \delta \quad (91) \]

Eq. (91) is substituted into (90) and pre-multiplied by \( S^T \) to give

\[ S^T MS\ddot{\delta} + S^T KS\dot{\delta} = S^T f \quad (92) \]

The matrices \( S^T M S \) and \( S^T K S \) are diagonal matrices which produce a system of uncoupled differential linear equations which take the form

\[ \ddot{\delta}_i + \omega_i^2 \delta_i = f_i(t) \]

each of which gives a solution of the form, assuming homogeneous initial conditions

\[ \delta(t) = a_i \int_0^t f_i'(t) \frac{\sin \omega_i(t-\tau)}{\omega_i} d\tau \quad (93) \]

where the \( a_i \) are determined from \( f_i' \) which are calculated and the integral on the right hand side of (93) is known as
Duhamel's convolution integral. The known solutions from (93) are substituted back into eq. (91) to give \( w(t) \)

\[
w(t) = S\delta(t)
\]

In this study, the first eleven mode shapes are considered in the solution, that is, the modal matrix \( S \) will have dimensions of 11 by the number of degrees of freedom. This is done so as not to force the finite element grid fit a high resolution. The linear three-node triangle is a "stiff" element, which would yield spurious mode shapes if too many eigenvalues were included. One possible approach to alleviating this problem would be to use a six-node quadratic triangle. This element is considerably more flexible than the latter. Fewer such elements, and even degrees of freedom, would be necessary before including more mode shapes into the solution.
6. L2 Success Criteria

The method used to described the effectiveness of each method is discussed here. The discretized form of the L2 method is employed here. Test cases are run for which an exact solution exists. A point grid over the solution area (e.g., $\tau_I + \tau_{II}$ in this case) is created. The difference between exact and approximate solutions are computed for each point on the grid and then squared. Each squared difference is then multiplied by a weighted area. These quantities are then summed. The resulting sum is divided by the area of the solution and the square root is taken of the result

$$
\varepsilon(t) = \sqrt{\sum_{q=1}^{Q} [V(x_q, y_q, t) - V^*(x_q, y_q, t)]^2 \Delta_q + \sum_{p=1}^{P} [u_p(t) - V^*(x_p, y_p, t)]^2 \Delta_p}
\sqrt{(\text{Area of } \tau_I + \tau_{II})}
$$

(94)

where the weighted areas are

$$\Delta_q = \mathbf{w}[(x_{q-1} - x_q)(y_{q-1} - y_q)]$$

$$\Delta_p = \mathbf{w}[(x_{p-1} - x_p)(y_{p-1} - y_p)]$$

$V^*(x, y, t)$ is the "exact" solution, or truth model (defined below), $Q$ is the number of classical solution degrees of freedom, and $P$ is the number finite element degrees of freedom.
The error $\varepsilon(t)$ is numerically integrated over the time interval that is equal to the inverse of the frequency of the lowest vibration mode

$$\Pi = \sum \varepsilon(t_i) \Delta t_i / T , \text{ with } \Delta t_i = \frac{P_n}{10P_0} \text{ and } T = t_2 - t_1$$

and where the period of the lowest natural frequency, $P_0 = 1/v_o$, the period of the highest natural frequency from the discrete system, $P_n = 1/v_n$, and $t_{i+1} = t_i + \Delta t_i$. 
7.0 Truth Model

The truth model used in this analysis is the eigenvalue solution for the entire unit square with a point load source located in the $\tau_{ii}$ region as before. Without going into the details of deriving the truth model the result is stated here

$$u_{\text{TRUTH}} = \sum_{i=1}^{\infty} Y_i(x,y)q_i(t)$$

$$Y_i(x,y) = \frac{4p_{mn}}{ab} \sin \frac{\pi mx}{a} \sin \frac{\pi ny}{b}$$

$$p_{mn} = \sin \frac{\pi n \xi}{a} \sin \frac{\pi n \zeta}{b}$$

where $(\xi, \zeta)$ is the point of application of the force, and

$$q_j(t) = \frac{1}{\omega_j} \int_0^t \bar{f}(\tau) \sin \omega_j(t-\tau) d\tau$$

$$\bar{f}(\tau) = \int_0^\tau \tau e^{-a \tau}, \text{ where we use } a = .3$$
From the truth model we may also obtain the exact values of the eigenfrequencies. These are determined from

$$\omega_i = \pi \sqrt{n^2 + m^2}$$  \hspace{1cm} (95)

where again the relationship between the subscript $i$ and the $m,n$ pairs is given by eq.(13).
8. Analysis Model

The region for which results are obtained in this study are based on a unit square that is shown in Fig. 8 below.

The expansions of equation (12) satisfy exactly the boundary conditions of the unit square shown above. Region $\tau_{II}$ is covered with linear triangular finite elements. The finite element nodes on the boundary $\sigma_2$ are set to zero so as to satisfy the homogeneous boundary conditions. The unit force acting on the unit square is applied to the point $(.75,.75)$ which places it in $\tau_{II}$, the finite element
region. Since the force is applied to only a single point, the energy associated with it equals the force magnitude multiplied by the field variable quantity of the point at which the force is applied. The model is simply

\[ u_q f(x, y, t) = u_q F_0 \delta(x-\xi, y-\zeta) t e^{-at} \]

where the nodal value \( u_q \) is located at node \( q \) which occupies the Cartesian coordinate \((\xi, \zeta)\), the point of application of the force, \( \delta(x-\xi, y-\zeta) \) is the Dirac delta function, and \( F_0 \) is the maximum magnitude of the force. Putting the above expression in terms of a finite element matrix would yield a vector matrix with all elements equal to zero except for the entry corresponding to node \( q \)

\[ u_q f(\xi, \zeta, t) = [0, 0, \ldots, 0, u_q, 0, \ldots, 0, 0]^T t e^{-at} \]

\[ = u_q [0, 0, \ldots, 0, 1, 0, \ldots, 0, 0]^T t e^{-at} \]

where the time varying term was chosen to simulate a "smooth pulse".

Before going to the next section, the term "resolution", often used in this study, is to be defined its appropriate context. It has to do with how refined an approximate solution we obtain. For example, "high"
resolution regarding finite elements means that the region has been modeled with a "large" number of elements as opposed to the region being modeled with a "small" number of elements as would be in the case of "low" resolution. In this study, part of the definition of resolution also relates to the number of terms used in the summation series of the classical solution, i.e., how many terms do we keep before truncating the series. The term "resolution" is defined in detail in Appendix C. The interface $\Gamma$ in Figure 8 is always taken to be parallel to the $y$-axis, and divides the region into two equal areas except when otherwise noted.
9. Results and Discussion

Comparison is made among the four methods using the L2 criterion as pointed out in section 6. The results are plotted in Figures 9, 10, and 11 for .1 to 10 seconds, each for different resolutions, and in Figure 12 for .1 to 20 seconds for resolution 16:8:5. All four plots show a comparable L2 error for all four methods, Ritz-Galerkin (RG), the penalty function (PF) method, collocation (Coll), and least square (LS). Figure 12a shows that L2 errors steadily converge after they peak at about three seconds, then begin to "scatter" after 15 seconds. An explanation for the "scatter" is not forthcoming at this time.

It must be pointed out the RG and PF methods retain all degrees of freedom, whereas in the LS and Coll methods the finite element nodes along the interface were constrained out. As a result of this one would expect a more accurate solution when applying RG or PF, however this is not apparent. As the resolution increases, the L2 error decreases for all four methods as is illustrated in Figures 13 - 16. This is normally expected.

In certain optimization problems, in order to satisfy constraints, a method known as the penalty function method
is applied. This involves multiplying a constraint equation by a penalty parameter, say $\mu$. The constraint equation, if ideally met, would be equal to zero, i.e., $c(x) = 0$. If the product between $\mu$ and $c(x)$ is always finite, then as $\mu$ is increased, the better the constraint $c(x) = 0$, is met. Thus one may simply expect to increase the penalty parameter without bound to better meet the constraint. This often works, however it does not always. As illustrated in Figures 17 and 18, increasing the parameter will yield a lower L2 error up to a certain extent. Beyond a certain penalty value the solution begins to deteriorate. For the 8:4:5 resolution, from Figure 17 it can seen that as $\mu$ is increased from 0 to 1000 the L2 error steadily decreases. The L2 error then begins to steadily increase as $\mu$ continues to increase beyond 1000 as Figure 18 illustrates. An explanation of this may lie with the fact that errors inherent with finite element discretization, and the classical solution truncation, are being magnified by their product with $\mu$ as it is being increased. This behavior is further evident from the results tabulated in Table I for a 10:5:5 resolution. For a penalty value of 100, the PF method yields the most
accurate values for the first four eigenfrequencies of the unit square when compared to the other four methods. For low penalty values (0 to 10), the connection between regions $\tau_i$ and $\tau_{ii}$ is very loose or does not exist and thus inaccurate estimates of the eigenfrequencies result. The errors for low penalty values tend to yield lower frequencies, for the second and fourth eigenfrequencies, than the actual values. This is expected, as with a low penalty value, the interface connection tends to disappear and the two regions just, so to speak, "flap in the breeze". It is evident that an exact solution for the second and fourth eigenfrequencies lies between penalty values of 10 and 100. From Table I it can be seen that estimates for the first and third eigenfrequencies are quite accurate. For the first frequency this can be explained by the fact that its mode shape does not change when $\mu$ is given a small value since the interface divides the two regions exactly in half. In fact, in this case, if there was no connection between the two regions at all ($\mu = 0$), the fundamental eigenfrequency (lowest frequency) would be the same for each subdomain, which is equal to the fundamental eigenfrequency for the entire domain. The
explanation for the why the third eigenfrequency estimate is also accurate is similar. In this case the interface cuts directly across a nodal line for this particular frequency’s mode shape. If the penalty is low or even zero, the interface becomes of little significance and the resulting calculation converges the second eigenfrequency of the “flapping” regions, which equals the second eigenfrequency of the entire region \((\tau_1 + \tau_2)\).

Continuing to refer to Table I, as expected based on our previous discussion, that as \(\mu\) gets significantly larger, the error in the eigenfrequency estimates tends in turn to get significantly larger. What is of interest here is that convergence is still achieved, however not to the desired solution. Note that the first eigenfrequency approaches the value of the second eigenfrequency as \(\mu\) gets very large, 1,000,000 and beyond. This was explained by the fact that the inherent discretization errors were being magnified as \(\mu\) increased without bound. When this occurs, the some coefficients of the classical solution, and all the finite element nodes on the interface individually tend to zero in order to maintain the constraint. The result we get is an interface that collapses to zero which results in
two constrained regions side by side. Thus the resulting
first two eigenfrequencies listed at the bottom left of
Table I are actually the first eigenfrequencies of each of
the constrained regions, that is \( \tau_1 \) & \( \tau_{II} \) become two
individually separated fixed regions each half the size in
area as when the two were combined. Note that while the
two are close in value, 49.1149 vs. 51.7531, the first one
is obtained from the classical expansion while the second
one, slightly higher, is obtained purely from finite
elements, hence a "stiffer" region. Referring to the
remainder of Table I results, we see that the collocation,
least square, and Ritz-Galerkin methods yield reasonably
accurate and consistent results for the estimation of the
first four eigenfrequencies.

The remaining illustrations, Figures 19 through 27
show the time history of the field variable response from
the force model that was defined in the previous section at
selected points for each method. Figures 19-21 show the
time history at the point (.75, .25), which is in \( \tau_{II} \), for
resolutions 8:4:5, 12:6:5, and 16:8:5 respectively. The
figures show that as the resolution increases the time
plots become more consistent, that is, the more they all
converge toward the truth model. Figures 22 and 23 show the time histories at the center of the unit square, point (.5, .5) for resolutions 8:4:5 and 16:8:5 respectively. As in the previous case, the time histories more closely resemble the truth model as the resolution increases. Figures 24 and 25 show time histories at a point in \( \tau_i \), coordinate (.25, .75), for resolutions 8:4:5 and 16:8:5. As before, higher resolution improves the accuracy of our trial solutions. In Figures 19 through 25 we used \( \mu = 10000 \).

Figure 26 demonstrates a point that was made earlier. For a low resolution, which means a higher discretization error, a lowering of the penalty function parameter, in this case from 10000 to 100, for resolution of 8:4:5, could enhance the accuracy of the penalty function method. Note that in this figure that the time history plot at point (.25, .75) for the penalty function method is considerably closer to the truth model than for \( \mu = 10000 \) which is shown in Figure 24.

Figure 27 shows the time history at the point where the forcing function was directly applied. All four methods show remarkable consistency with each other. The truth model, although shaped very similarly to our trial
solutions, is considerably more displaced. This is due to
the singular nature of the solution at the point of
application. The solution at this point diverges while it
converges at all other points as is evident from the
previous time history plots discussed earlier. The error
that results at this coordinate contributes significantly
to the L2 error. If the point of application L2 error were
removed from the L2 calculation, the L2 errors in the
vicinity of the 3 second point would be reduced by
approximately by 50% as indicated when Figure 28 is
compared with Figures 9-12. Figure 29 shows a direct
comparison when the L2 error of the point load is removed
for the μ method case.

Significant points that are drawn from this study are as
follows:

1. All methods yield comparable L2 results with Ritz-
   Galerkin slightly lagging.

2. Optimal solution with penalty function method is achieved
   with a finite value of the penalty function parameter.
3. Coarse finite element grid limits the effectiveness of the penalty function method.

4. Point of force application contributes disproportionately to L2 error.

5. L2 error converges with time then eventually begins to "scatter".
Table I: Comparisons of the First Four Eigenvalues

<table>
<thead>
<tr>
<th>Resolution 10:5:5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eigenvalue #</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual value</td>
<td>19.739209</td>
<td>49.348022</td>
<td>49.348022</td>
<td>78.956835</td>
</tr>
<tr>
<td>collocation</td>
<td>19.977865</td>
<td>50.37758</td>
<td>50.8647</td>
<td>82.613221</td>
</tr>
<tr>
<td>% error</td>
<td>1.209045408</td>
<td>2.086320704</td>
<td>3.073432204</td>
<td>4.630866979</td>
</tr>
<tr>
<td>Galerkin</td>
<td>19.9885966</td>
<td>51.006616</td>
<td>51.851084</td>
<td>82.280147</td>
</tr>
<tr>
<td>% error</td>
<td>1.26528373</td>
<td>3.361014146</td>
<td>5.072264092</td>
<td>4.209023829</td>
</tr>
<tr>
<td>least square</td>
<td>19.9797</td>
<td>50.407799</td>
<td>50.871945</td>
<td>82.492596</td>
</tr>
<tr>
<td>% error</td>
<td>1.218341627</td>
<td>2.147557201</td>
<td>3.088113643</td>
<td>4.478093632</td>
</tr>
<tr>
<td><strong>Penalty Function</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mu = 0$</td>
<td>19.739206</td>
<td>20.003921</td>
<td>49.34592</td>
<td>52.316578</td>
</tr>
<tr>
<td>% error</td>
<td>-1.51982E-05</td>
<td>-59.46358093</td>
<td>-0.004259543</td>
<td>-33.74027974</td>
</tr>
<tr>
<td>$\mu = 1$</td>
<td>19.97168</td>
<td>26.881608</td>
<td>50.361089</td>
<td>57.673786</td>
</tr>
<tr>
<td>% error</td>
<td>1.177711832</td>
<td>-45.93175791</td>
<td>2.052902951</td>
<td>-26.95529652</td>
</tr>
<tr>
<td>$\mu = 10$</td>
<td>19.97998</td>
<td>43.563494</td>
<td>50.605007</td>
<td>75.022694</td>
</tr>
<tr>
<td>% error</td>
<td>1.219760123</td>
<td>-11.72190448</td>
<td>2.549210584</td>
<td>-4.982647797</td>
</tr>
<tr>
<td>$\mu = 100$</td>
<td>19.983254</td>
<td>50.051553</td>
<td>50.335739</td>
<td>81.618661</td>
</tr>
<tr>
<td>% error</td>
<td>1.236346401</td>
<td>1.425651873</td>
<td>2.001533111</td>
<td>3.371242021</td>
</tr>
<tr>
<td>$\mu = 1000$</td>
<td>20.015452</td>
<td>50.622402</td>
<td>50.971517</td>
<td>82.41645</td>
</tr>
<tr>
<td>% error</td>
<td>1.399463373</td>
<td>2.5824338</td>
<td>3.208889701</td>
<td>4.381653596</td>
</tr>
<tr>
<td>$\mu = 10000$</td>
<td>20.331815</td>
<td>50.597557</td>
<td>55.084279</td>
<td>82.616862</td>
</tr>
<tr>
<td>% error</td>
<td>3.002177038</td>
<td>2.532087304</td>
<td>11.62408698</td>
<td>4.637783417</td>
</tr>
<tr>
<td>$\mu = 100000$</td>
<td>23.269343</td>
<td>50.670369</td>
<td>67.530608</td>
<td>83.565925</td>
</tr>
<tr>
<td>% error</td>
<td>17.8386759</td>
<td>2.67965265</td>
<td>36.84522171</td>
<td>5.837480694</td>
</tr>
<tr>
<td>$\mu = 1000000$</td>
<td>37.673439</td>
<td>50.748026</td>
<td>78.098154</td>
<td>86.048893</td>
</tr>
<tr>
<td>% error</td>
<td>90.85586966</td>
<td>2.837001248</td>
<td>58.25994809</td>
<td>8.982196411</td>
</tr>
<tr>
<td>$\mu = 10000000$</td>
<td>48.170717</td>
<td>51.365026</td>
<td>78.79103</td>
<td>127.85925</td>
</tr>
<tr>
<td>% error</td>
<td>144.0357007</td>
<td>4.087304654</td>
<td>59.86400842</td>
<td>61.9356323</td>
</tr>
<tr>
<td>$\mu = 50000000$</td>
<td>49.114949</td>
<td>51.753138</td>
<td>78.906669</td>
<td>128.28177</td>
</tr>
<tr>
<td>% error</td>
<td>148.8192359</td>
<td>4.873783999</td>
<td>59.8934202</td>
<td>62.47076013</td>
</tr>
<tr>
<td>Higher mode approx for</td>
<td>49.114949</td>
<td>51.753138</td>
<td>78.906669</td>
<td>128.28177</td>
</tr>
<tr>
<td>$\mu = 500000000$, ...</td>
<td>0.472304645</td>
<td>4.873783999</td>
<td>0.063535981</td>
<td>0.017994034</td>
</tr>
</tbody>
</table>
Figure 9: L2 Comparisons, Four Methods
Resolution 8:4:5, $\mu = 10000$
Figure 10: L2 Comparisons, Four Methods
Resolution 12:6:5, $\mu = 10000$

![Graph showing L2 comparisons for four methods: Ritz-Galerkin, Collocation, Least Squares, and Penalty Function. The graph plots L2 error against time in seconds, with data points indicating the performance of each method at different time intervals.](image-url)
Figure 11: L2 Comparisons, Four Methods
Resolution 16:8:5, $\mu = 10000$
Figure 12: L2 Comparisons, Four Methods
Resolution 16:8:5, $\mu = 10000$, (20 sec)
Fig 12a: L2 Comparison, Four Methods

Resolution 16:8:5, /\( \leq 1000 \text{ (50 sec)} \)

Penalty Function

Ritz-Galerkin

Coiiocation Least Squares

Time, Seconds

50 40 30 20 10 0

0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0.00

0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0.00

L2 Error

Fig 12a: L2 Comparison, Four Methods
Figure 13: L2 Error, Ritz-Galerkin for Various Resolutions

- Resolution 16:8:5
- Resolution 12:6:5
- Resolution 8:4:4
Figure 14: L2 Error, Penalty Function
For Various Resolution, $\mu = 10000$
Figure 15: L2 Error, Least Squares for Various Resolutions

Time, Seconds

L2 Error

Resolution 16:8:5 • Resolution 12:6:5 • Resolution 8:4:5
Figure 16: L2 Error, Collocation for Various Resolutions
Figure 18: L2 Comparisons for Various F, (Continued)

Resolution 8:4:5

Time, Seconds

0 1 6 8 7 9 5 4 3 2

Resolution 8:4:5

L2 Error

0 10'0 20'0 30'0 40'0 50'0 60'0 70'0 80'0 90'0 100'0
Figure 19: Field Variable vs. Time
Pt.(.75, .25), Res 8:4:5, $\mu = 10000$

Field Variable vs. Time
Ritz-Galerkin - a - Penalty Function Collocation
Least Squares Truth Model
Figure 20: Field Variable vs. Time

Pt(.75, .25), Res 12:6:5, $\mu = 10000$
Figure 21: Field Variable vs. Time
Pt(0.75, 0.25), Res 16:8:5, \( \mu = 10000 \)
Figure 22: Field Variable vs. Time
Pt (0.5, 0.5), Res 8:4:5, μ = 10000

Field Variable, w(0.5, 0.5, t)

Time, Seconds

Ritz-Galerkin  Penalty Function  Collocation
Least Squares  Truth Model
Figure 23: Field Variable vs. Time
Pt (.5, .5), Res 16:8:5, μ = 10000
Figure 24: Field Variable vs. Time
Pt (.25, .75), Res 8:4:5, μ = 10000
Figure 25: Field Variable vs. Time
Pt (25, .75), Res 16:8:5, $\mu = 10000$

Field Variable vs. Time

- Ritz-Galerkin
- Penalty Function
- Collocation
- Least Squares
- Truth Model
Figure 26: Field Variable vs. Time

Field Variable, $w(0.25, 0.75, t)$

- Ritz-Galerkin
- Least Squares
- Penalty Function
- Truth Model
- Collocation

Time, Seconds

0.00 0.01 0.02 0.03 0.04 0.05

-0.01 0.00 0.01 0.02 0.03 0.04 0.05
Figure 27: Time Variance of Load Point
Pt (.75, .75), Res 16:8:5, $\mu = 10000$
Figure 28: L2 Comparisons
Point Load Removed, $\mu = 1000$
Figure 29: L2 Error Comparison
With and Without Point Load, $\mu = 1000$

Time, Seconds

L2 Error

- With Point Load L2
- No Point Load L2
Appendix A: m & n pair mapping onto a single index

Example: Find single index terms for the expansion

\[ v(x,y,t) = \sum_{n=1}^{n_{\text{max}}} \sum_{m=1}^{m_{\text{max}}} \sin \frac{\pi nx}{a} \sin \frac{\pi my}{b} = A_{11}(t) \sin \frac{\pi x}{a} \sin \frac{\pi y}{b} + \]
\[ \quad A_{12}(t) \sin \frac{\pi x}{a} \sin \left( \frac{2\pi y}{b} \right) + A_{13}(t) \sin \frac{\pi x}{a} \sin \left( \frac{3\pi y}{b} \right) + A_{21}(t) \sin \left( \frac{2\pi x}{a} \right) \sin \frac{\pi y}{b} + \]
\[ \quad A_{22}(t) \sin \left( \frac{2\pi x}{a} \right) \sin \left( \frac{2\pi y}{b} \right) + A_{23}(t) \sin \left( \frac{2\pi x}{a} \right) \sin \left( \frac{3\pi y}{b} \right) + A_{31}(t) \sin \left( \frac{3\pi x}{a} \right) \sin \frac{\pi y}{b} + \]
\[ \quad A_{32}(t) \sin \left( \frac{3\pi x}{a} \right) \sin \left( \frac{2\pi y}{b} \right) + A_{33}(t) \sin \left( \frac{3\pi x}{a} \right) \sin \left( \frac{3\pi y}{b} \right) \quad \text{(B1)} \]

Here we see that \( m_{\text{max}} = 3 \), so that

\[ k = 3(n-1) + m \quad \text{(B2)} \]

For the first three terms in the expansion, \( i = 1, 2, 3 \) for \( n = 1, m = 1, 2, 3 \)

\[(m,n) \Rightarrow (1,1) \rightarrow k=1, \quad (1,2) \rightarrow k=2, \quad (1,3) \rightarrow k=3, \]
\[(2,1) \rightarrow k=4, \quad (2,2) \rightarrow k=5, \quad (2,3) \rightarrow k=6, \]
\[(3,1) \rightarrow k=7, \quad (3,2) \rightarrow k=8, \quad (3,3) \rightarrow k=9 \]

For this example \( A_{13} = A_6, \ A_{31} = A_7, \ A_{11} = A_3, \ A_{22} = A_5, \ A_{32} = A_8, \) etc.

Also, for example,

\[ \sin \left( \frac{3\pi x}{a} \right) \sin \left( \frac{3\pi y}{b} \right) = \psi_9(x,y), \quad \sin \left( \frac{\pi x}{a} \right) \sin \left( \frac{2\pi y}{b} \right) = \psi_5(x,y), \]
\[ \sin \left( \frac{3\pi x}{a} \right) \sin \left( \frac{\pi y}{b} \right) = \psi_7(x,y), \quad \sin \left( \frac{3\pi x}{a} \right) \sin \left( \frac{2\pi y}{b} \right) = \psi_8(x,y), \] etc.
The mapping of \( m,n \) pairs to a particular \( k \) is unique. This can be described in the following way. The maximum value of \( m \) is fixed. As the first inner summation in (B1) takes place, the index \( n \) of the outer summation begins with the number one. The index \( m \) is then varied from one to its maximum value, in this case, three. The number \( k \) varies from to three accordingly. As the next inner summation sweep takes place, \( n \) is given the value of two. The first term on the right side of (B2) has a value of three. When \( m \) varies from one to three \( k \) takes on the values 4, 5 and 6 which of course is distinct from 1, 2, and 3. This procedure is repeated until the summation is complete.

With each inner sum sweep we go through distinct \( m,n \) pairs. Likewise we respectively produce distinct values of \( k \). The illustration below demonstrates the point.

Solid lines are plots of \( k = m_{\text{max}}(n-1) + m \) where \( m_{\text{max}} \) is equal to 3 in this case. For every constant value of \( n \), \( m \) varies from 1 to 3.
Appendix B: Finite Element Assemblage

The following example illustrates the method of finite element assembly.

Let the following trapezoidal region be divided into three finite elements

The numbers in parentheses represent the element number, the interior numbers at the nodes represent the local element nodal number, and the exterior numbers represent the global node numbers. Each element has its own energy representation, i.e.,

\[
J^{(1)} = \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_3^{(1)} \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{22}^{(1)} & a_{23}^{(1)} & a_{23}^{(1)} \\ \text{sym} & a_{33}^{(1)} & \text{sym} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_3^{(1)} \end{bmatrix}
\]

\[
J^{(2)} = \begin{bmatrix} u_1^{(2)} \\ u_2^{(2)} \\ u_3^{(2)} \end{bmatrix} \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{22}^{(2)} & a_{23}^{(2)} & a_{23}^{(2)} \\ \text{sym} & a_{33}^{(2)} & \text{sym} \end{bmatrix} \begin{bmatrix} u_1^{(2)} \\ u_2^{(2)} \\ u_3^{(2)} \end{bmatrix}
\]

\[
J^{(3)} = \begin{bmatrix} u_1^{(3)} \\ u_2^{(3)} \\ u_3^{(3)} \end{bmatrix} \begin{bmatrix} a_{11}^{(3)} & a_{12}^{(3)} & a_{13}^{(3)} \\ a_{22}^{(3)} & a_{23}^{(3)} & a_{23}^{(3)} \\ \text{sym} & a_{33}^{(3)} & \text{sym} \end{bmatrix} \begin{bmatrix} u_1^{(3)} \\ u_2^{(3)} \\ u_3^{(3)} \end{bmatrix}
\]
We have assumed symmetry for simplicity. At this point could we add all three energy terms to get the total energy

\[ J = J^{(1)} + J^{(2)} + J^{(3)} \]

However this formulation is meaningless unless inter-element constraints are imposed, namely the values of element nodes that share a common location, e.g., the node 2 of element one shares the same node as node 1 of element 2, and likewise node 1 of element 3. All three have the same nodal value, which is equal to the global node 3, i.e.,

\[ u_3 = u_2^{(1)} = u_1^{(2)} = u_1^{(3)} \]

Constraints for the remainder of the nodes are

\[ u_1 = u_1^{(1)} \]
\[ u_2 = u_3^{(1)} = u_3^{(2)} \]
\[ u_4 = u_2^{(2)} = u_3^{(3)} \]
\[ u_5 = u_2^{(3)} \]

After incorporating the constraints, the energy term for element (1), in terms of the global nodes, takes the form

\[ J^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ \text{sym} & a_{32}^{(1)} & a_{33}^{(1)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \\ u_2 \end{bmatrix} \]
Expressing $J^{(1)}$ in terms of the remaining global nodes gives, after appropriate rearrangement matrix elements, now take the form

$$J^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{13}^{(1)} & a_{12}^{(1)} & 0 & 0 \\ a_{33}^{(1)} & a_{13}^{(1)} & a_{23}^{(1)} & 0 & 0 \\ a_{11}^{(1)} & a_{12}^{(1)} & 0 & 0 & 0 \\ \text{sym} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_1 \\ 0 & 0 & 0 & 0 & u_2 \\ 0 & 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & 0 & u_4 \\ 0 & 0 & 0 & 0 & u_5 \end{bmatrix}$$

Similarly, for $J^{(2)}$ and $J^{(3)}$ we have

$$J^{(2)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{33}^{(2)} & a_{13}^{(2)} & a_{23}^{(2)} & 0 & 0 \\ a_{11}^{(2)} & a_{12}^{(2)} & 0 & 0 & 0 \\ \text{sym} & a_{22}^{(2)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_1 \\ 0 & 0 & 0 & 0 & u_2 \\ 0 & 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & 0 & u_4 \\ 0 & 0 & 0 & 0 & u_5 \end{bmatrix}$$

$$J^{(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ a_{11}^{(3)} & a_{13}^{(3)} & a_{12}^{(3)} & 0 & 0 \\ \text{sym} & a_{33}^{(3)} & a_{23}^{(3)} & a_{22}^{(3)} & 0 \\ 0 & 0 & 0 & 0 & u_1 \\ 0 & 0 & 0 & 0 & u_2 \\ 0 & 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & 0 & u_4 \\ 0 & 0 & 0 & 0 & u_5 \end{bmatrix}$$

Note: For nodes that do not lie on elements their corresponding rows and columns will be made up entirely of zeros. For example, nodes 1 and 2 do not lie on element (3), thus the corresponding rows and columns are made up entirely of zeros.
At this stage we are ready for assembly. We simply add the three 5x5 matrices together to get for the entire system matrix

$$K = \begin{bmatrix}
    a_{11}^{(1)} & a_{13}^{(1)} & a_{12}^{(1)} & 0 & 0 \\
    a_{33}^{(1)} + a_{33}^{(2)} & a_{23}^{(1)} + a_{13}^{(2)} & a_{23}^{(2)} & a_{22}^{(1)} + a_{21}^{(2)} + a_{11}^{(3)} & a_{12}^{(2)} + a_{13}^{(3)} + a_{12}^{(3)} \\
    a_{22}^{(1)} + a_{11}^{(2)} + a_{11}^{(3)} & a_{12}^{(2)} + a_{13}^{(3)} & a_{12}^{(3)} & a_{22}^{(2)} + a_{33}^{(3)} & a_{23}^{(3)} \\
    \text{sym} & a_{22}^{(2)} + a_{33}^{(3)} & a_{23}^{(3)} & a_{22}^{(3)} & a_{22}^{(3)}
\end{bmatrix}$$
Appendix C: Definition of Resolution

Resolution as defined in this study is an indication of how many finite elements are utilized and to what extent the summation limits of the classical solution. An example is given for the figure below. Resolution is expressed in the form N1:N2:N3 where N1 represents the number of finite element divisions in the y direction in the region τi as shown in the figure. Similarly, N2 represents the number of finite element divisions in the x direction. N3 is the limit of each the summations in the classical expansion expression.

Example:

\[ v(x, y, t) = \sum_{m=1}^{5} \sum_{n=1}^{5} A_{mn} \sin(m\pi y)\sin(n\pi y), \text{ note that limit of each sum is 5} \]

Thus resolution for this case is 8:4:5
Appendix D: Computer Codes

The computer codes for this project are presented here. In the first three sections code lines are listed for the following calling programs:

1) collocation/least squares, Appendix D1
2) Ritz-Galerkin, Appendix D2
3) Penalty Function Method, Appendix D3

The next code section listed is for the utility programs. Included in this section are subfunctions and subroutines that the calling programs listed above utilize. These include the stiffness and mass matrix formulations for finite element and classical expansions, numerical integration, and matrix multiplication and inversion routines. This section is enumerated as Appendix D4.

The next section, Appendix D5, is where the modal analysis is done. This is where the code lines for the calculation of the eigenvalues and eigenvectors are listed.

The final section, Appendix D6, lists all the routines that calculate the time response solution, the truth model, and the L2 error estimates. The diagram below illustrates the flow of operations.
Appendix D1: Collocation-Least Square Methods Calling Code

program main
  dimension xfe(800,3),yfe(800,3)
  dimension at(800),rl(800,3),the(800,3),usol(400)
  dimension ae(800,3),be(800),ce(800,3),cm(20,50),cls(20,50)
  dimension ainv_test(3,3),ainv(3,3), defl_cns(400,400)
  dimension coef_inf(500,500),defl(500,500),cst(500,400)
  dimension cstr(400,500),c_str(500,400),cf_infl(400,400)
  dimension k2(20,50,3),k2p(20,50,3),k3(20,3,50),k3p(20,3,50)
  dimension kl(20,3,3),kl(20,3,3),k3(20,3,50),k3p(20,3,50)
  real k2(20,50,3),k2p(20,50,3),k3(20,3,50),k3p(20,3,50)
  real ki(50,50),mi(50,50),kg(500,500),mg(500,500)
  real kiv(50,50),ki(50,50)
  real kiij, kivij,normal, angle, mu
  real kii(800,3,3),mii(800,3,3)
  real kgm(500,500),kgamg(500,500)
  integer offset,p,q,dof,iinal_dof,ei(800,3)
  open(8,file='drivers.dat')
  open(IO,file='evp.daf')
  c open(12,file='static_sol.dat')
  nlim = 5
  mlim = nlim
  c number of grid spaces in the x direction for gauss integration
  nxd = 2
  c number of grid spaces in the y direction for gauss integration
  nxd = 4
  c whole region (square) x limit
  a = 1.
  c whole region (square) y limit
  b = 1.
  c number of integration points on gamma
  ns = n
  c calculate number of degrees of freedom for the classical solution
  offset = nlim*mlim
  c set penalty function parameter
  c For future reference we should let nxdiv lie between 0 and nbe. The driver
  c which would be the number of grid lines on the square in each direction.
  c Then set nydiv = nbe.
  c number of grid spaces in the x direction for gauss integration
  nxdiv = 8
  c number of grid spaces in the y direction for gauss integration
  nydiv = nbe
  c grid space limit for integrating classical solution via Gauss
  nxdiv = 8
  c coordinates of first gamma integration point
  x0 = a
  y0 = 0.
  c coordinates of last gamma integration point
  xn = a
  yn = b
  c the dof parameter below is the total number of degrees of freedom
  c for the unconstrained system.
  dof = offset + nodes
  c final number of degrees of freedom for penalty and Galerkin methods
  nfdof = offset + nxdiv*nydiv
  c final number of dof for collocation and least square (constrained)
  nfdofc = nfdof - (nbe - 1)
  c Begin the general assembly
  write(*,*) nlim,nlim,offset,nxdiv,50,50
  call fe_sparsity(xfe,be,ce,xfy,ie,thet,1)
  call coupled_matrix_creation(nlim,nlim,offset,nxdiv,50,50)
  call fe_4_m matrices(xfe,be,ce,xfy,ie,thet,1)
  call uncoupled_matrix creation(xfe,ce,xfe,xfy,ie,thet,1)
  call coupled_mat_asm(xfe,be,ce,k2,k2p,k3,k3p,k3g,kgam,kgamg)
  call assem_uncoupled(xfe,be,ce,k2,k2p,k3,k3p,k3g,kgam,kgamg,offset,kg,mg)
  c the next two routines represent the penalty function method and the Galerkin
  c method
  c mu is set to zero when the least square problem is formulated
  c mu = 0.
  call mu_method(kg,kgam,kgamg,offset,kg,mg)
  c
  c stiffness matrix is inverted here as the first step to calculating K inverse
  c The following routines are the solutions to the collocation and least square
  c problems
  c call coupled_matrix_creation(nlim,nlim,offset,nxdiv,50,50)
  c call coupled_matrix_creation(nlim,nlim,offset,nxdiv,50,50)
  c call coupled_matrix_creation(nlim,nlim,offset,nxdiv,50,50)
  c The following routines are to be used in the matrix iteration scheme
  c onto unit 10, which then goes into Hornung's matrix deflation algorithm
  c write(10*) nfdof
  do 200 i=1,nfdof
    write(10*) (defl_cns(ij),j=1,nfdof)
  200 continue
  do 201 i=1,nfdof
    write(10*) (cstr(ij),j=1,nfdof)
  201 continue
end
Appendix D2: Ritz-Galerkin Calling Code

```
program main

dimension x&(800,3), yfe(800,3)
dimension at(800), rl(800,3), thet(800,3)
dimension ae(800,3), b^800, 3), ce(800,3)
dimension ainv_tes^3, 3), ainv(3,3)
dimension coef_inf(S00,500), defl(S00,500)
real k2(20,50,3), k2p(20,50,3), k3(20,3,50), k3p(20,3,50)
dimension xg(20), yg(20)
real ki(50,50), mi(50,50), kg(500,500), mg(500,500)
real klg8[500,500], mgfl[500,500)
real kiv(50,50), kl(50,50)
real kii(800,3,3), mii(800,3,3)
real kgam(500,500), l%amg(500,500)
integer ofrset, p, q, dof, iinal_dof, ei(800)
open(8, file='drivers.dat')
open(10, file='*evp.dat')

nlim=5

mlim=nlim

cnumberofgrid spacesintheXdirectionforgaussintegration
nxd=2

cnumberofgrid spacesintheydirection forgaussintegration
nyd=4

c whole region (square) x limit
a = 1.
c whole region (square) y limit
b = 1.
calculate number of degrees of freedom for the classical solution
offset = nlim*nlim
c For future reference we should let nxdiv lie between 0 and nbe. The driver
should be nbe
which would be the number of grid lines on the square in each direction.
Then set nxd = nbe.
c
nbe = 16
nxdiv = 8
nydiv = nbe
c
nydiv = 2*nxdiv
nel = 2*nxdiv*nydiv
c
nbe = nxdiv
c
m = nxdiv + 1
c
m = (nxdiv-1)/xdiv + 1
c
grid space limit for integrating classical solution via Gauss
mbe = nbe
mxdiv = nxdiv
agrid = 1.0
r = nxdiv/mbe
c
coordinates of first gamma integration point
x = agrid
y = 0.
c
coordinates of last gamma integration point
xn = agrid
yn = 1.
c
Number of unconstrained nodes
c
n = nxdiv*(nxdiv+1)/2

c The dof parameter below is the total number of degrees of freedom
c for the unconstrained system.
dof = offset + nsnodes
c
final number of degrees of freedom for penalty and Galerkin methods
nsdf = offset + nxdiv*(nxdiv+1)
c
final number of dof for collocation and least square (constrained)
c
write(10,*) snodes
close(10)
c
Begin the general assembly
call el_ind_midpt(x, nydiv, nxdiv)
call fe_constants(x, ae, be, ce, xfe, yfe, rl, thet,
set, rle, xdiv, nydiv, nxdiv, nbe, a, b)
call coupled_matrix_creation(xlim, ylim, xdiv, ydiv, nxdiv, nydiv, nbe, a, b,
xdiv, ydiv, ae, be, ce, xfe, yfe, rl, thet, x, y, rl)
call fe_on_gam(x, y, be, ce, xfe, yfe, rl, thet, a, a)
call fe_k_m_matrices(ae, ce, be, ainv, xfe, yfe, rl, thet, a, a)
call fe_assembly(kii, mi, x, set, offset, kg, mg)
call uncoupled_matrix_creation(xlim, ylim, xdiv, ydiv, nxdiv, nydiv, nbe, a, b,
set, rle, xdiv, nydiv, nxdiv, 50, a, b, xdiv, ydiv, ae, be, ce, xfe, yfe, rl, thet, x, y, rl)
call coupled_mat_assem(ki, k2, k2p, k3, k3p, k4, kgam, kgamg,
1 snbe, set, ml lim, xdiv, offset, ei, nlim, nlim)
call assen_uncoupled(ki, k1, kiv, kgam, kgamg, offset, kg, mg)
c
The next two routines represent the penalty function method and the Galerkin
method
mu is set to zero when the least square problem is formulated
c
call galerkin(kg, kgamg, dof)
do 110 i=1, dof
110 continue
do 111 i=1, dof
111 continue
c
calculate stiffness and mass matrices below are formed to take into account
c the fact that the finite element nodes on the boundary are zero.
c
calculate matricel(sfinal(kg, mg, nxdiv, nydiv, dof, offset, kg, mg)
c
c stiffness matrix is inverted here as the first step to calculating K inverse
c
calculate_matrix2(kg, coef_inf, 500, nfdof)
c
c Here, the deflation matrix that is used in finding eigenvalues via the matrix
iteration
c method for either the Galerkin or penalty function method, is calculated below,
calculate matrix_k3(kg, coef_inf, 500, 500, 500, 1
nfdof, nfdof, nfdof, defl)
c
write(10,*) nfdof
do 92 i=1, nfdof
write(10,*) defl(i, j)= defl(j, i)
92 continue
do 97 i=1, nfdof
write(10,*) mg(i, j)= mg(j, i)
97 continue
c
close(10)
c
continuation
```
Appendix D3: Penalty Function Method Calling Code

program main
  dimension xfe(800,3), yfe(800,3)
dimension at(800), rl(800,3), theta(800,3), usol(400)
dimension ae(800,3), be(800), ce(800,3), cm(20,50), cls(20,50)
dimension amv_tes(3,3), amv(3,3), defl_cas(400,400)
dimension coef_inf(500,500), defl(500,500), cst(500,400)
dimension cst_tr(400,500), c_tem(500,400), cf_infl(400,400)
dimension k2(20,3,3), k2p(20,3,3), k3(20,3,50), k3p(20,3,50)
dimension xg(20), yg(20)
  real k2(20,50,3), k2p(20,50,3), k3(20,3,50), k3p(20,3,50)
  real klpp(3,3), k4(20,3,3), kc8tr(400,400)
  c dimension xg(20), yg(20)
  real kK50,50, m(50,50), kg(500,500), mg(500,500)
  real kgf(500,500), gf(500,500), mcstr(400,400)
  real kiv(50,50), kl(50,50)
  real kii(800,3,3), mii(800,3,3)
  real lqg(500,500), lkgam(500,500)
  integer ofrset, p, q, dof, final_dof, ei(800,3)
  open(8, file='driver5.dat')
  open(10, file='evp.dat')
  c open(12, file='static_sol.dat')
nlim = 5
mlim = nlim
  c number of grid spaces in the x direction for gauss integration
  nxdiv = 2
  c number of grid spaces in the y direction for gauss integration
  nydiv = 4
  c whole region (square) x limit
  a = 1.
  c whole region (square) y limit
  b = 1.
  c number of integration points on gamma
  nna = 4
  c calculate number of degrees of freedom for the classical solution
  offset = nlim*nlin
  c For future reference we should let nxdiv lie between 0 and nbe. The driver should be "nbe"
  c which would be the number of grid lines on the square in each direction.
  c Then set nydiv = nbe.
  c nbe = 16
  c nxdiv = 8
  c nydiv = 8
  c nydiv = 2*nxdiv
  c m = nxdiv + 1
  c m = (nbe-nxdiv) + 1
  c grid space limit in x for integrating classical solution via Gauss
  agrid = 1 - nxdiv/txdiv
  c grid space limit in y for integrating classical solution via Gauss
  bgrid = 1.0
  c coordinates of first gamma integration point
  x0 = agrid
  y0 = 0
  c coordinates of last gamma integration point
  x = agrid
  y = 1
  c total number of finite element nodes including the boundary
  nodes = (nxdiv+1)*(nydiv+1)
  c The dof parameter below is the total number of degrees of freedom
  c for the unconstrained system.
  dof = offset + nodes
  c final number of degrees of freedom for penalty and Galerkin methods
  nfdof = offset + nxdiv*(nydiv-1)
  c final number of degrees of freedom for penalty and Galerkin methods (constrained)
  nfdof = nfdof - (nbe - 1)
  c write the three driving parameters to a file
  write(K.*) nlim, nbe, nxdiv
  close(K)
  c Begin the general assembly
  call el_ind_midp(x, y, dxdiv, nxdiv)
  call fe_constants(xa, be, co, xk, yk, ftheta,
  1  nfdof, nxdiv, nydiv, nbe, a, b)
  call coupled_matrix_creation(nlin, nlim, nbe, nfdof, 50, a, b, 1
  x, y, dxdiv, nbe, xe, be, co, xk, yk, ftheta,
  1  call fe_on_gam(x, y, dxdiv, nxdiv, nbe, a, b, k4)
  call fe_k_m_matrix(x, y, dxdiv, nxdiv, nbe, a, b, m)
  call fe_assembly(ki, mi, e, offset, kg, mg)
  call uncoupled_matrix_creation(x0, y0, xi, yi, 1000,
  1  nlim, nlin, a, x, y, dxdiv, agrid, bgrid, ki, mi, kv, k1)
  c do 27 i = 1, 4
  c write(6, *) (kiv(i,j), j = 1, 4)
  c do 27 continue
  c coupled_matrix_assm(k2, k2p, k3, k3p, k4, kgm, kgam,
  1  nbe, nfdof, offset, e, llim, lim)
  c assm_uncoupled(ki, mi, k1, kv, kg, kgam, offset, kg, mg)
  c
  c The next two routines represent the penalty function method and the Galerkin method
  c mu is set to zero when the least square problem is formulated
  c mu = 1000.
  call mu_method(kg, kgam, mg, dof)
  c call galerkin(kg, kgam, mg, dof)
  do 110 j = 1, dof
  do 110 i = 1, dof
  kg(i,j) = kg(i,j) + kg(j,i)
  do 110 continue
  do 111 j = 1, dof
  kg(j,j) = kg(j,j) + kg(j,i)
  do 111 continue
  c the final stiffness and mass matrices below are formed to take into account
  c the fact that the finite element nodes on the boundary are zero.
  call matrices_final(kg, mg, nxdiv, nydiv, dof, offset, kgf, mgf)
  c
  c stiffness matrix is inverted here as the first step to calculating K inverse
  call inverse_matrix(kg, coef_inf, 500, nfdof)
  c
  c Here, the deflation matrix that is used in finding eigenvalues via the matrix iteration
  c method for either the Galerkin or penalty function method, is calculated below.
  c call matrix_mult(coef_inf, mgf, 500, 500, 500, defl)
  call matrix_mult3(coef_inf, mgf, 500, 500, 500, nfdof, nfdof, nfdof, defl)
  write(10, *) nfdof
  do 92 j = 1, nfdof
  write(10, *) (defl(j,j), j = 1, nfdof)
  92 continue
  do 97 j = 1, nfdof
  write(10, *) (mgf(j,j), j = 1, nfdof)
  97 continue
  c close(10)
  stop
end
Appendix D4: Utility Routines

function psi(n,ni,x,y,a,b)
pi = acos(-1.)
argvx = n*pi*x/a
argvy = m*pi*y/b
psi = sin(argvx)*sin(argvy)*cos(theta)/a
return
end

function dpсидn(n,m,x,y,a,b,theta)
pi = acos(-1.)
argvx = n*pi*x/a
argvy = m*pi*y/b
dpsид = m*pi*cos(argvx)*sin(argvy)*cos(theta)/a
1 + m*pi*sin(argvx)*cos(argvy)*sin(theta)/b
return
end

function xpsi(n,m,x,y,a,b)
xpsi = x*psi(n,m,x,y,a,b)
return
end

function ypsi(n,m,x,y,a,b)
ypsi = y*psi(n,m,x,y,a,b)
return
end

function kiij(n,m,p,q,x,y,a,b)
kiij = dpsид(n,m,x,y,a,b)*dpsид(n,m,x,y,a,b)
1 + dpsид(n,m,x,y,a,b)*dpsид(n,m,x,y,a,b)
return
end

function xpsи(n,m,x,y,a,b)
xpsи = x*dpsи(n,m,x,y,a,b)
return
end

function yпsи(n,m,x,y,a,b)
ypsi = y*dpsи(n,m,x,y,a,b)
return
end

function xпsi(n,m,x,y,a,b)
xпsi = x*psi(n,m,x,y,a,b)
return
end

function yпsi(n,m,x,y,a,b)
ypsi = y*psi(n,m,x,y,a,b)
return
end

realtiinction kivij(n,m,p,q,x,y,a,b,theta)
kivij = .5*(term1 + term2)
return
end

function pipj(n,m,p,q,x,y,a,b)
pipj = psi(n,m,x,y,a,b)*psi(n,m,x,y,a,b)
return
end

subroutine seg_pars(x0,y0,xn,yn,ns)
common/params/ddx, dely, ddgam, theta
debc = (xn-x0)/ns
defy = (yn-y0)/ns
gam = sqrt((xn-x0)^2 + (yn-y0)^2)
delgam = gam/ns
theta = acos((ym-y0)/gam)
call seg_pars(x0,y0,xn,yn)
sum = 0.
do 20 i = 0, ns-1
xh = x0 + delx*(i + .5)
yh = y0 + dely*(i + .5)
20 continue
sum = sum + xpsi(n,m,xh,yh,a,b)
return
end

...
\[(y(k,2) - y(k,3))r(y(U) - y(k,3))
\]
\[dl = (x(k,3) - x(M))y_{rl}(k,1)
\]
\[dy = (y(k,3) - y(U))y_{rl}(k,1)
\]
\[theta(k,1) = \text{angle}(dx, dy)
\]
\[dx_2 = (x(k,1) - x(k,3))y_{rl}(k,2)
\]
\[dy_2 = (y(k,1) - y(k,3))y_{rl}(k,2)
\]
\[theta(k,2) = \text{angle}(dx_2, dy_2)
\]
\[dx_3 = (x(M) - x(k,1))y_{rl}(k,3)
\]
\[dy_3 = (y(M) - y(k,1))y_{rl}(k,3)
\]
\[theta(k,3) = \text{angle}(dx_3, dy_3)
\]
\[10 \text{ continue}
\]
\[return
\]
\[end
\]
\[subroutine \text{fe}_k_m_niatiice^a,b,c,at,nel,kii,mii)
\]
\[dimension a(800,3), b(800,3), c(800,3), \text{at}(800)
\]
\[real mii(800,3,3), kii(800,3,3)
\]
\[do10 k = 1, nel
\]
\[a_{4d} = \text{at}(k,4)
\]
\[a_{12} = \text{at}(k,12)
\]
\[do20 i = 1, 3
\]
\[\text{do20 j} = 1, 3
\]
\[kii(k,ij) = a_{ij} \times (a_{ki} \times a_{kj} + b_{ki} \times b_{kj})
\]
\[mii(k,ij) = \text{at}_{12}
\]
\[mii(k,1,1) = 2 \times \text{at}_{12}
\]
\[mii(k,2,2) = mii(k,1,1)
\]
\[mii(k,3,3) = mii(k,1,1)
\]
\[10 \text{ continue}
\]
\[return
\]
\[end
\]
\[subroutine \text{fe}_assembly(kii^ii, ei, nel, offset, l(8*mg)
\]
\[real kg(500,500), mg(500,500), kii(800,3,3), ii(800,3,3)
\]
\[integer of, \text{ei}(800,3)
\]
\[do10 k = 1, nel
\]
\[\text{do20 i} = 1, 3
\]
\[\text{do20 j} = 1, 3
\]
\[i_p = \text{ei}(k, i) + \text{offset}
\]
\[j_p = \text{ei}(k, j) + \text{offset}
\]
\[k_{B}(i_p, j_p) = kg(i_p, j_p) + \text{at}_{4d}
\]
\[m_{B}(i_p, j_p) = mg(i_p, j_p) + mii(k,ij)
\]
\[20 \text{ continue}
\]
\[10 \text{ continue}
\]
\[return
\]
\[end
\]
\[real function normal_angle(dx, dy)
\]
\[pi = \text{acos}(-1.)
\]
\[\text{if}(dx \leq 0. \text{ and } dy \geq 0.) \text{ normal_angle} = \text{asin}(-dx)
\]
\[\text{if}(dx \leq 0. \text{ and } dy \leq 0.) \text{ normal_angle} = \text{pi} + \text{asin}(dx)
\]
\[\text{if}(dx \geq 0. \text{ and } dy \leq 0.) \text{ normal_angle} = \text{pi} + \text{asin}(dx)
\]
\[\text{if}(dx \geq 0. \text{ and } dy \geq 0.) \text{ normal_angle} = 2 \times \text{pi} - \text{asin}(dx)
\]
\[return
\]
\[end
\]
\[subroutine \text{gauss}_{\text{rect}}(n, m, p, q, nxdiv, nydiv, agrid, bgrid, cgrid)
\]
\[1, xg(800,3), yg(800,3)
\]
\[dimension xg(800,3), yg(800,3)
\]
\[integer do, offset
\]
\[do10 i = 1, offset
\]
\[do10 j = 1, offset
\]
\[kg(i, j) = \text{ki}(i, j)
\]
\[mg(i, j) = \text{mi}(i, j)
\]
\[kgam(i, j) = \text{kl}(i, j)
\]
\[kgamg(i, j) = \text{kiv}(i, j)
\]
\[10 \text{ continue}
\]
\[return
\]
\[end
\]
\[subroutine \text{coupled}_{\text{matrix}}(nl, ml, nbe, nei, nxdiv, nydiv, a, b, c, k2, k2p, k3, k3p)
\]
\[dimension a(800,3), b(800,3), c(800,3)
\]
\[integer do, offset
\]
\[do10 i = 1, offset
\]
\[do10 j = 1, offset
\]
\[k2(i, j) = k2(i, j)
\]
\[k3(i, j) = k3(i, j)
\]
\[k2p(i, j) = k2p(i, j)
\]
\[k3p(i, j) = k3p(i, j)
\]
\[10 \text{ continue}
\]
\[return
\]
\[end
\]
\[subroutine \text{mu}_{\text{method}}(kg, kgam, kgamg, offset, kg, mg)
\]
\[dimension kg(800,3), kgam(800,3), kgamg(800,3)
\]
\[integer do, offset
\]
\[do10 i = 1, offset
\]
\[do10 j = 1, offset
\]
\[kg(i, j) = kg(i, j)
\]
\[kgam(i, j) = kgam(i, j)
\]
\[kgamg(i, j) = kgamg(i, j)
\]
\[10 \text{ continue}
\]
\[return
\]
\[end
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
\[C
\]
do 30 m = 1, ml
  j = ml*(i-1) + m
  x0 = x(k,3)
  y0 = y(k,3)
  call line_int_psi(x0,y0,xn,yn,ns,sum1,n,m,aa,bb)
  call line_int_xpsi(x0,y0,xn,yn,ns,sum2,n,m,aa,bb)
  call line_int_ypsi(x0,y0,xn,yn,ns,sum3,n,m,aa,bb)
  call line_int_npsi(x0,y0,xn,yn,ns,sum4,n,m,aa,bb)
  do 50 j = 1,3
    k2(kn,l,j) = (a(k,i)*sum3+b(k,i)*sum2+c(k,i)*sum1)
    k3(kn,ij) = a(k,i)*sum6+b(k,i)*sum4+c(k,i)*sum2
  50 continue
  do 51 i = 1,3
    k2p(kn,l,i) = a(k,i)*sum3+b(k,i)*sum2+c(k,i)*sum1
    k4p(kn,l,i) = a(k,i)*sum6+b(k,i)*sum4+c(k,i)*sum2
  51 continue
  do 40 continue
  return
end

subroutine f_o_omega(i1,theta,a,b,nbe,klp,k4)
dimension i1(800,3),theta(800,3),a(800,3),b(800,3)
real klp(20,3,3),k4(20,3,3)
clear the registers
  do 30 kn = 1,nbe
    do 25 i = 1,3
      do 25 j = 1,3
        klp(kn,i,j) = 0.
        k4(kn,i,j) = 0.
      25 continue
    30 continue
  do 40 kn = 1,nbe
    k = (2*nxdiv)*(kn-1)+1
    k4(kn,2,2) = r(k,iy3.
    k4(kn,3,3) = k4(kn,2,2)
    k4(kn,2,3) = r(k,iy6.
    k4(kn,3,2) = k4(kn,2,3)
    do 35 i = 2,3
      do 35 j = 1,3
        klp(kn,i,j) = (rl(k,iy2.)*(a(kj)*sin(theta(k,l))
                       +b(k,l)*cos(theta(k,l)))
      35 continue
    40 continue
  return
end

subroutine uncoupled_matrix_creation(xO,yO,xn,yn,ns,nlim,mlim,aa,bb,nxd,nyd,agrid,tgrid,ki,fni,kiv,kl)
real ki(500,500),mi(500,500),kiv(500,500),kl(500,500)
dimension node_check(800)
calculate the registers
  do 10 n = 1,nlim
    do 10 m = 1,mlim
      i = (n-1)*mlim+m
      do 10 q = 1,mlim
        j = (q-1)*mlim+q
        call line_int_kin(x0,y0,xn,yn,ns,sum1,n,m,p,q,aa,bb)
        call line_int_kin(x0,y0,xn,yn,ns,sum2,n,m,p,q,aa,bb)
        call gauss_rect_kin(n,m,p,q,aa,bb)
        call gauss_rect_kin(n,m,p,q,aa,bb)
        call line_int_kin(x0,y0,xn,yn,ns,sum1,n,m,p,q,aa,bb)
        call line_int_kin(x0,y0,xn,yn,ns,sum2,n,m,p,q,aa,bb)
        call line_int_kin(x0,y0,xn,yn,ns,sum3,n,m,p,q,aa,bb)
        call line_int_kin(x0,y0,xn,yn,ns,sum4,n,m,p,q,aa,bb)
      10 continue
  return
end

subroutine create_inverse_matrix(kg,mg,nxdiv,nydiv,dof,offset,kgf,mgf)
real kg(500,500),mg(500,500),kgf(500,500),mgf(500,500)
dimension node_check(800)
calculate the registers
  do 51 i = 1,nodes
    do 40 j = 1,offset
      kgf(i,j) = kg(i,j)
      mgf(i,j) = mg(i,j)
    40 continue
  do 30 continue
  return
end

subroutine matrix_mult3(a,b,n,m,ir,nl,ml,irl,c)
dimension a(n,m),b(m,ir),c(n,ir)
calculate the registers
  do 20 i = 1,nl
    do 20 j = 1,irl
      sl = 0.
      do 20 k = 1,ml
        sl = sl + a(i,k)*b(k,j)
      20 continue
      c(i,j) = sl
  10 continue
  return
end
do 100 i = 1, n
100 l(i,l) = a(i,l)
dol01j = 2, n
101 u(lj) = a(lj)/l(l,l)
dol02k = l, n
102 u(k,k) = l.
do 30 j = 2, n-l
30 jml = j-l
dol0i = j, n
50 s = 0.
dol0k = l, jml
10 sl = sl + l(i,k)*u(kj)
20 l(ij) = a(ij) - sl
do 40 i = 2, n
40 ipl = i + l
40 iinl = i - l
dol0j = ipl, n
20 s2 = 0.
dol0k = l, iinl
60 s2 = s2 + u(i,k)*l(kj)
62 uO,k) = a(i,k) - s2
30 continue
s3 = 0.
dol0k = l, n-l
62 s3 = s3 + l(n,k)*u(kn)
64 l(n,k) = a(n,k) - s3
do 70 j = 1, n
70 do 70 j = 1, n
70 b(ij) = 0.
do 71 j = 1, n
71 b(ij) = 1.
do 80 i = 1, n
80 d(ij) = b(ij)*l(i,j)
do 81 i = 1, n
81 s = 0.
do 82 k = 1, l
82 s2 = s2 + d(kj)*l(k)
81 d(ij) = b(ij) - s2
80 continue
do 90 j = 1, p
90 sinv(ij) = d(ij)
do 91 i = 1, l, -1
91 s = 0.
do 92 k = 1, l
92 s2 = s2 + u(kj)*sinv(kj)
91 sinv(ij) = d(ij) - s2
90 continue
return
end
kl=(k-1)/(nxdiv-1)

{i-1,1} = i+kl+1

20 \text{ cst}(i.j)=1.

\text{do } 40 i=1,\text{ offset } + \text{ ndoff}
\text{do } 40 j=1,\text{ offset } + \text{ ndoff } - (nydiv-1)

40 \text{ cst}_{\text{tr}(i.j)}=\text{ cst}(i.j)

\text{return}
\text{end}
Appendix D5: Modal program

program main
   c implicit double precision (a-h,o-z)
   dimension ak(500,500),am(500,500),u(500)
   open(14,file='egvhot.dat')
   open(15,file='egvout.dat')
   continue
   read(9,*) lim
   do 40 j=1,lim
      40 continue
   open(9, file='egvhot.dat')
   open(18, file='egvout.dat')
   continue
   return
   ndim=5
   end
   open(9, file='egvhot.dat')
   subroutine egv_hotte(a,am,u,rlam,lim,ndim)
   c implicit double precision (a-h,o-z)
   dimension a(ndim,ndim), u(ndim)
   do 40 j=1,lim
      40 continue
   u(0)=1
   call egv_iterate(ak,riam,ndim,lim,1.0,.000001)
   do 10 i=1,lim
      10 continue
   return
   c The following eigenvectors are normalized wrt the mass matrix
   subroutine sort_largest(u,lim,ndim,n)
   c implicit double precision (a-h,o-z)
   dimension u(ndim)
   do 61 i=1,lim
      61 continue
   if(i.eq.1) goto 20
   if(i.eq.2) goto 75
   return
   continue
   goto 81
   continue
   write(6,'(*)') 'solution achieved in', kount, 'iterations'
   return
   end
   subroutine egv_hotte(a,am,u,rlam,lim,ndim)
   c implicit double precision (a-h,o-z)
   dimension a(ndim,ndim), am(ndim,ndim), u(ndim)
   dimension utemp(500,500), s(500)
   do 10 i=1,lim
      s(i) = 0
      do 10 k=1,lim
         10 continue
      s2 = 0
      d
      do 40 i=1,lim
         40 continue
      sl(i) = abs(s(i))
      do 40 i=1,lim
         40 continue
      umag = sqrt(s2)
      do 20 i=1,lim
         20 continue
      do 40 i=1,lim
         40 continue
   return
   c The following eigenvectors are normalized wrt the mass matrix
   subroutine sort_largest(u,lim,ndim,n)
   c implicit double precision (a-h,o-z)
   dimension u(ndim)
   do 61 i=1,lim
      61 continue
   if(i.eq.1) goto 20
   if(i.eq.2) goto 75
   return
   continue
   goto 81
   continue
   write(6,'(*)') 'solution achieved in', kount, 'iterations'
   return
   end
   subroutine sort_largest(u,lim,ndim,n)
   c implicit double precision (a-h,o-z)
   dimension u(ndim)
   do 61 i=1,lim
      61 continue
   if(i.eq.1) goto 20
   if(i.eq.2) goto 75
   return
   continue
   write(6,'(*)') 'number of iterations exceeded'
   return
   end
Appendix D6: Transient Solution Code

```fortran
program main
  integer offset
  open(17, file='drivers.dat')
  do 80 i = 1, 100
    t = t + delt
    read(17, *) nlim, nbe, nxdiv
    open(10, file='nlim.dat')
    read(10, *) offset
    call nonn_eigval(eigvect_norm, nfdof, sm, smt)
    continue
  continue
  open(19, file='eps.dat')
  open(20, file='trans.dat')
  do 81 i = 1, 1000
    lim = lim + delt
    write(6, *) nlim, nbe, nxdiv
    nfdof = offset + nxdiv*(nydiv-1)
    call force_coef(30, 30, a, b, 1.0, 1.0, pf)
    t = t + delt
  continue
  close(19)
  close(20)
end
```

The steps below give a profile for 20 points
at a particular instant in time

```fortran
do 82 i = 1, 20
  usol = usol + offset, nxdiv, sm, smt, (x, y, t)
  write(6, *) usol
end
```

Calculations of L2 Error are made here

```fortran
d = 1
  t = 0.
  do 91 i = 1, 50
    call 12_convergence(nbe, pf, offset, nxdiv, nydiv, nfdof, 1.0, pf, eps, eps
    t = t + delt
  continue
  close(15)
end
```

Call transient response on the finite element grid

```fortran
if(ls.eq.0) then
  subroutine grid_gen_lscol(offset, nydiv, nxdiv, a, b, xc, yc, x, y)
  call grid_gen_igpen(offset, nydiv, nxdiv, xc, yc, 1.0, 1.0, x, y)
  dimension x(300), y(300), xc(300), yc(300)
  endif
if(ls.eq.1) then
  m = nydiv - nxdiv + 1
  call grid_gen_lscol(offset, nydiv, nxdiv, 1.0, 1.0, xc, yc, x, y)
endif
mxdiv = nxdiv
mydiv = nydiv
```

```fortran
subroutine grid_gen_lscol(offset, nydiv, nxdiv, a, b, xc, yc, x, y)
dimension x(300), y(300), xc(300), yc(300)
integer offset
  m = nydiv - nxdiv + 1
  mxdiv = nxdiv
  mydiv = nydiv
  dx = x(nxdiv)
  x = x(nxdiv)
  x(nxdiv) = x(nxdiv)
end
```

```
```
```fortran
c & x & y coordinates in the in the classical solution region
c do 12 j = 1, nydiv - 1
  y(j) = y(j) + dely
end
```

```fortran
c c & x & y coordinates in the finite element region
c do 10 j = 1, nydiv - 1
  x(j) = x(j) + delx
end
```

```fortran
```
```fortran
```
10 y(which) = y(del)
return
end

subroutine grid_gsr_open(nxdiv,nydiv,xx,xya,xyb,xyx,xyy,xyz)
dimension x(nxdiv),y(nydiv),z(nxdiv)
integer offset
mxdiv = nxdiv
mydiv = nydiv
delx = x(nxdiv)
dely = y(nydiv)
m = mxdiv - nxdiv + 1
xm = delx / (m - 1)
c
x & y coordinates in the in the classical solution region
do 12 j=1,mydiv - 1
do 12 i=1,mxdiv - 1
12 continue

y(which) = y(del)
c
x & y coordinates in the in the finite element region
do 10 j=1,mydiv
do 10 i=1,mxdiv
10 continue

function x sol(nx,ny,offset,nxdiv,nydiv,sm,smx,smn,amp,1
freq,ndof,kmodes,ixwhich,iywhich)
dimension sm(500,20,500),sm(500,500),sm(500,20),sm(500,500)
integer offset
if (ix.which.eq.0) corface = 2
if (iy.which.eq.0) corface = 3
ns = nxdiv
moddiv = (nxdiv - 1) / ns
ir = offset + (ns * moddiv + ns * 2) / 2 - moddiv
if (i > 10) kmodes
fr = freq(k)
eta = (k,ir)*force_alt2(3,fr,x)
endif
10 continue

ww = 0
ww = ww + sm(which,k)*eta(k)
ww = w_sol(nx,ny,offset,nxdiv,nydiv,sm,smx,smn,amp,freq,ndof,1
kmodes,ixwhich,iywhich)
return
end

subroutine nom_signal(nxdiv,nydiv,smx,smn,ampl)
dimension eig_vec(500,20,20,20,500)
integer offset
if (i > 10) pdoff
30 continue

sm(j) = eig_vec(j,ir,fr,ampl)
30 continue

sm(j) = sm(j)
do 40 j=1,nom
40 continue

return
end

subroutine 1d_convergence(nx,ny,offset,nydiv,mydiv,ampl,amp,ndof,1
kmodes,nmix,nlim,ka,xb,xy,xy,smx,smn,freq,eps)
dimension x(500),y(500),xy(500),xy(500),xy(500),xy(500)
dimension freq(20),p(100,100)
integer offset
pi = acos(-1)
c
vxy = x(y WHICH)
return
end

subroutine force_coef(nmix,nlim,ka,xb,xy,smx,smn,amp,p)
dimension p(100,100)
p(i,j) = 0
ar = p(i,j)
sr = p(i,j)
do 10 j=1,nlim
10 continue

pi = (p(xy,xb)) * sin(1*2*pi) * sin(2*2*pi)
return
end

subroutine mesh

function v_star(nmix,nlim,amp,p(x,xy,smx,smn)
dimension p(100,100)
p(i,j) = 0
sum = 0
ar = p(i,j)
c
ar = pop
c p(m,n) = (4.*amp/(a*b))*sin(argx*m)*sin(argy*n)
c 10 continue
c fxyn = force_dist(l_1.x_st_y_st,100,100,75,75,1,p)
do 100 m=1,mslim
argx = m*pi*x/a
argy = m*pi*y/a
do 100 n=1,nslim
argy = n*pi*y/b
argx = n*pi*x/a
p(m,n) = (4.*amp/(a*b))*sin(argx*m)*sin(argy*n)
q freq = pi*sqrt((in/a)*(ni/a)+(n/n))
c write(6,*) freq
c cc = (2.*amp/argx(a*b))*sin(argx)*sin(argy)
c write(6,*) cc
c Dirac delta impulse force input
c sum = sum + p(m,n)*sin(argx)*sin(argy)*sin(freq*t)/(freq)
c write(6,*) cc
v_star = sum
return
end

function force_alt2(exp_const,freq,t)
  f1 = 1.*(exp_const**2 + freq**2)
  f2 = 2.*exp_const*freq
  f3 = (exp_const**2 - freq**2)*f1/freq
  f4 = 1./(freq**2)
  f5 = 1./f4
  force_alt2 = f5*sin(freq*t)
  return
end