This is to certify that the
thesis entitled

MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT MODELS OF COMPONENTS

presented by

Zhen Ren

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Mechanical Engineering

Clark Radcliffe
Major Professor's Signature

April 4, 2008
Date

MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT MODELS OF COMPONENTS

By

Zhen Ren

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Mechanical Engineering

2008

ABSTRACT

MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT MODELS OF COMPONENTS

By

Zhen Ren

Analytical engineering design is a global activity requiring efficient global distribution of analytical models of dynamic physical systems through computer networks. Finite Element Method (FEM) models are used globally to analyze the response of physical systems assembled from physical components. FEM models from different physical component suppliers often have geometrically incompatible meshes. This geometric incompatibility of mesh node placement typically requires reformulation of component models to allow those models to be assembled into a model of a system of those components. The modular model assembly introduced in this paper does not require such component model reformulation. It assembles incompatible finite element component models fast and with accuracy comparable to traditional reformulation. The proprietary geometry and material component details are not revealed during the assembly. Modular model assembly can be used to assemble distributed component models through the internet in global engineering design. Dynamic examples are provided.

A paper discussing this work written by Zhen Ren and Clark Radcliffe has been submitted to 2008 Dynamic Systems and Control Conference of ASME.

ACKNOWLEDGEMENTS

My thanks are due to Dr. Clark Radcliffe for advising my Master program, sharing enthusiasm and guidance.

My thanks are also due to Dr. Jon Sticklen and Dr. Ronald Rosenberg for serving as my Master of Science program committee members.

Thanks to many faculty members in Mechanical Engineering, Michigan State University for suggestions that are very useful for my research work.

Finally, thanks to my family and my friends for encouraging me for many years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

The Finite Element Method (FEM) is widely used in industry today. It is a method to discretize a continuous problem in structural dynamics, fluid mechanics, heat transfer, and other field problems. The FEM yields accurate, detailed response for individual system components with local FEM mesh discretization. Its solution approaches the true continuum solution as the number of discrete variables increase [1]. The traditional FEM method requires geometrically compatible FEM models at any assembly interface where the nodes of component meshes have identical positions. Guaranteed compatibility of the nodal structure in finite element models from different suppliers is often not possible. To assemble a system FEM model from different suppliers often requires complete reformulation and rederivation of the component models in the assembly. Reformulating components is time consuming. For commercial components, model reformulation also requires proprietary component internal material and geometry design details.

The Modular Modeling Method (MMM) uses an energy based fixed input/output model structure [2] to assemble internet-distributed component models into an assembly model using a standardized process. The MMM uses external ports only and hides internal information. The MMM has shown the ability to assemble FEM models without remeshing of the component models [3] when using local linear interpolation between adjacent nodes. The size of the assembly model can be reduced and the model can be distributed without revealing component details. The locally linear interpolated MMM

assembly model can also provide accuracy comparable to traditional FEM models with comparable mesh resolution [3-4].

This paper introduces a standardized Internet-based model assembly method using Lagrange interpolation. It is a global interpolation method for assembly based on MMM. This approach is demonstrated using an FEM model of a mechanical system assembly. Two FEM models of two-dimensional plates with incompatible meshing are assembled using the MMM approach. The results of both static and dynamic simulation results are then compared with the traditional results from commercial software (ANSYS). When component models are geometrically compatible, the static and dynamic simulation results generated by MMM assembly are the same as traditional FEM assembly. When the component models are geometrically incompatible, the static and dynamic simulation results generated by MMM assembly usually lay between the result generated by traditional FEM assembly of compatible lower resolution meshes and the result generated by traditional FEM assembly of compatible higher resolution meshes.

Analytical test results successfully show that FEM models assembled using MMM provides engineering analysis accuracy comparable to traditional FEM reformulation. It also assembles incompatible component models without reformulation. The MMM assembly is fast and does not require revealing proprietary component design detail information. These features will enable the MMM to be used to assemble component models distributed through the internet for global engineering design and system analysis.

Modular Modeling Method in FEM Analysis

2. 1 Definition of model inputs and outputs

Modular modeling is a model assembly method designed to eliminate model equation reformulation and enable model experimental performance verification [2]. Energy based input-output ports are used for assembling modular models. A set of standardized input-output pairs are selected. The output variable is selected as the variable physically constrained at the assembly port during the assembly of components. The outputs at the assembly interface are equal with compatible ports. The assembly of physical systems also requires energy conservation. Input variables are selected as the variables required for computing units of energy when it is multiplied by their respective output [5].

The concept of the input and output pair selection can be represented mathematically [5]. In an assembly, the connected component nodal outputs $\mathbf{Y}_c$ are expressed as a linear combination of the assembly nodal outputs $\mathbf{Y}_a$.

$$\mathbf{Y}_c = \begin{bmatrix} \mathbf{Y}_{c_1} \\ \mathbf{Y}_{c2} \\ \vdots \\ \mathbf{Y}_{cn} \end{bmatrix} = \mathbf{S}\mathbf{Y}_a \qquad (1)$$

where $\mathbf{S}$ is a linear constraint matrix consisting of constants and constrains component output vector $\mathbf{Y}_c$ and assembly output vector $\mathbf{Y}_a$. The physical constrain requires energy flow into an assembly port must equal the energy flow into the connected components ports. The energy/power conservation is defined as:

$$\mathbf{U}_c^{\mathrm{T}}\mathbf{Y}_c = \mathbf{U}_a^{\mathrm{T}}\mathbf{Y}_a \tag{2}$$

where $\mathbf{U}_a$ is the assembly port input vector and the $\mathbf{U}_c$ is component port input vector. Using assembly constraints (1) in the energy conservation equation (2), yields a complementary relationship (3) between assembly port inputs and component port inputs [5].

$$\mathbf{S}^{\mathrm{T}}\mathbf{U}_c = \mathbf{U}_a \tag{3}$$

Energy Domain	Output (y)	Input (u)
Electrical	Potential	Charge
Mechanical (Translation)	Displacement	Force
Mechanical (Rotation)	Angle	Torque
Hydraulic	Pressure	Volume
Acoustic	Sound pressure	Volume
Heat Transfer	Temperature	Heat Flux

Table 1 Input and Outputs for Different Energy Domains [5]

Input and output pairs are defined in different engineering domains (Tab.1). For instance, assembly of wires at a port constrains the potential output for all component and assembly wires at that port to be the same. The assembly constraint (1) sets electrical potential as an output. The corresponding amount of input electrical charge going from an external wire into the assembly port must equal the sum of the charges that goes from the port into the component wires (3).

2.2 Modular Finite Element Model assembly

An unassembled finite element component $\mathbf{P}_i$ in an energy domain with m degrees of freedom (DOF) can be expressed as

$$\mathbf{P}_{ci}\,\mathbf{Y}_{ci} = \mathbf{U}_{ci} \qquad or$$

$$\begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1m} \\ P_{21} & P_{22} & \ddots & P_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mm} \end{bmatrix}_{Ci} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix}_{Ci} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_m \end{bmatrix}_{Ci} \qquad (4)$$

where $\mathbf{U}_{ci}$ and $\mathbf{Y}_{ci}$ are nodal force (input) vector and nodal displacement (output) vectors for the component i. A system with n unassembled components can be represented as a superposition (2.5) of the component models. The subscripts 1 to n represent components 1 to n. The component input vector (nodal force) and component output vector (nodal displacement) are represented by $\mathbf{U}_{ci}$ and $\mathbf{Y}_{ci}$, respectively.

$$\mathbf{P}_c\,\mathbf{Y}_c = \mathbf{U}_c \qquad or$$

$$\mathbf{P}_c\,\mathbf{Y}_c = \begin{bmatrix} \mathbf{P}_{c1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{c2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P}_{cn} \end{bmatrix} \begin{bmatrix} \mathbf{Y}_{c1} \\ \mathbf{Y}_{c2} \\ \vdots \\ \mathbf{Y}_{cn} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{c1} \\ \mathbf{U}_{c2} \\ \vdots \\ \mathbf{U}_{cn} \end{bmatrix} = \mathbf{U}_c \qquad (5)$$

Using assembly constraints (1) in (5)

$$\mathbf{P}_c\mathbf{S}\mathbf{Y}_a = \mathbf{U}_c \qquad (6)$$

Multiplying energy conservation equation (6) by $\mathbf{S}^{\mathbf{T}}$ and substituting (3), the assembly model contains only assembly node variables

$$\mathbf{P}_a\,\mathbf{Y}_a = \mathbf{U}_a \qquad (7)$$

where the assembly system matrix

$$\mathbf{P}_a = \mathbf{S}^{\mathbf{T}}\mathbf{P}_c\mathbf{S}$$

The assembled model (7) has the same standard form as the component model (4). After the assembled model is generated, the assembly model itself can also be used as a component model for higher level assembly until a final product model is assembled.

The challenge for assembly of a discretized field problem using FEM is to define a global output constraint. In many cases, the nodal structures of the FEM component models along the assembly interfaces are geometrically incompatible. An interpolation is required for constraint matrix $\mathbf{s}$ to constrain the component nodal outputs and the assembly nodal outputs. Previously, the constraint was constructed based on local linear interpolation [3-4]. The local linear interpolation is complex to operate and there was no standard form. A constraint based on Lagrange Interpolation introduced in this paper is much more convenient to implement.

2.3 Linear constraint matrix from Lagrange interpolation

Lagrange interpolation is used in the linear constraint matrix to constrain the component nodal outputs along the assembly interface. Lagrange interpolation has a standard form. Using Lagrange interpolation, the mandatory remeshing of incompatible component models in traditional FEM is no longer required.

Lagrange interpolation is frequently used in finite element analysis as an easy and systematic way of generating a shape function of any order [1]. An $n-1$ order Lagrange interpolation generates a unique curve on the $s-y$ plane connecting n points $((y_1,s_1)\cdots(y_i,s_i)\cdots(y_n,s_n))$. The Lagrange interpolation polynomial $L_i(s)$ is selected such that the $L_i(s)$ equals one at point i, and zero at all the other points.

$$L_i(s)=\frac{(s-s_1)}{(s_i-s_1)}\frac{(s-s_2)}{(s_i-s_2)}\cdots\frac{(s-s_{i-1})}{(s_i-s_{i-1})}\frac{(s-s_{i+1})}{(s_i-s_{i+1})}\cdots\frac{(s-s_n)}{(s_i-s_n)} \qquad (8)$$

The Lagrange interpolation function:

$$y(s)=\sum_{i=1}^{n} y_i L_i(s) \qquad (9)$$

allows a continues output $y(s)$ to be computed as a function of n assembly node outputs y_i along the assembly interface. At any position along the assembly interface, the output can be written in vector form as:

$$y(s)=\begin{bmatrix}L_1(s) & L_2(s) & \cdots & L_n(s)\end{bmatrix}\begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_n\end{bmatrix}_a \qquad (10)$$

To implement a Lagrange interpolation assembly constraint, let $y_{c_i}(s_i)=y(s_i)$ and constrain component nodes along the assembly interface. Using the component nodal interpolation (10) in (1), a general form of the Lagrange linear constraint can be obtained (11), where n is the number of assembly nodes and m is the number of component nodes. s_l is the geometry position of the node l on the component. From (11), any

component nodal output is now constrained to be a function of assembly nodal outputs based on its geometrical position. All component nodal output values are always a linear function of the assembly nodal outputs along the field interface interpolation.

$$\begin{bmatrix} y_{c_1}(s_1) \\ y_{c_2}(s_2) \\ \vdots \\ \vdots \\ y_{c_m}(s_m) \end{bmatrix} = \begin{bmatrix} L_1(s_1) & L_2(s_1) & \cdots & \cdots & L_n(s_1) \\ L_1(s_2) & L_2(s_2) & \cdots & \cdots & L_n(s_2) \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ L_1(s_m) & L_2(s_m) & \cdots & \cdots & L_n(s_m) \end{bmatrix} \begin{bmatrix} y_{a_1} \\ y_{a_2} \\ \vdots \\ \vdots \\ y_{a_n} \end{bmatrix} \qquad (11)$$

and $$\mathbf{S}_{ij} = \left[L_j(s_i)\right]$$

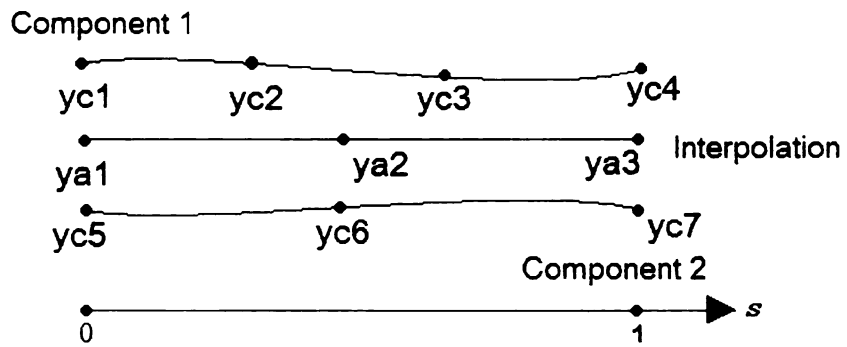

Figure 1 Interpolation Interface Used to Assemble Two Components

As a special case, when the components have compatible components along the assembly interface, the assembly nodes can be chosen at the same positions as the component nodes. By doing that, the entries in the linear constraint matrix are 1 when correspond nodes position overlap and 0 at everywhere else. The output interpolation joins only nodes at the same geometrical position. The outputs of the component nodes at each assembly point are constrained to be equal. This is equivalent to one of the rules used in traditional FEM assembly: displacement compatibility [1], or continuity of the primary variable [6].

Following is an assembly interpolation interface that joins two 1-D components with incompatible meshing of 3 elements and 2 elements (Fig.1). The nodes on the components are numbered from 1 to 7. The unconstrained nodal outputs are y_{c_i}. The system has 3 assembly nodes y_{a_i}.

From the Lagrange interpolation polynomial (8), the Lagrange interpolation polynomials are

$$\begin{aligned} L_1(s) &= 2(s-0.5)(s-1) \\ L_2(s) &= -4(s)(s-1) \\ L_3(s) &= 2(s-0.5)(s) \end{aligned} \tag{12}$$

Each individual component nodal output can be represented by a linear combination of assembly nodal outputs. For instance, the component nodal output at node 3, Y_{c_3}, can be represented as:

$$Y_{c3} = y(2/3) = L_1(2/3)Y_{a_1} + L_2(2/3)Y_{a2} + L_3(2/3)Y_{a3} = -\frac{1}{9}Y_{a_1} + \frac{8}{9}Y_{a2} + \frac{2}{9}Y_{a3}$$

The Lagrange linear interpolation matrix is:

$$\mathbf{Y}_c = \begin{bmatrix} 1 & 0 & 0 \\ 2/9 & 8/9 & -1/9 \\ -1/9 & 8/9 & 2/9 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Ya1 \\ Ya2 \\ Ya3 \end{bmatrix} = \mathbf{S}\mathbf{Y}_a \tag{13}$$

MMM assembly examples

This example is used to validate the MMM assembly method on both geometrically compatible and incompatible FEM component models. The mechanical domain example represents general MMM finite element assembly. FEM components in other domains, which can be represented by (7), e.g.: heat transfer, can also be assembled using the same approach. In the following examples, the accuracy of MMM Lagrange interpolation assembly is tested in two stages. First, geometrically compatible component models are assembled using MMM and the results are compared with traditional FEM results. Second, geometrically incompatible component models are assembled using MMM. Because traditional FEM cannot assemble geometrically incompatible component models, the MMM assembly results are compared with different compatible FEM models at resolutions both lower and higher than the resulting MMM assembly.

Geometrically compatible MMM assembly results are compared with 2 models. The first is model constructed using commercial software ANSYS, which provides an industrially trusted system model $\mathbf{P}$ and output (displacement) solution. Unfortunately, ANSYS does not provide system model through the user interface. So a second model is assembled using traditional FEM methods in MATLAB. The second model generates both stiffness matrices and the displacement solution through the user interface. The results from the second model will be compared with ANSYS results, allowing validation of the component models.

The MMM assembly generates results comparable with the traditional FEM assembly. When the component models are geometrically compatible, the MMM generates the same assembly model as the traditional FEM. When component models are geometrically incompatible, the results from the MMM assembly lie between the results generated by a traditional FEM assembly at higher and lower resolution.

3.1 FEM Dynamic Analysis Using Modular Modeling Assembly

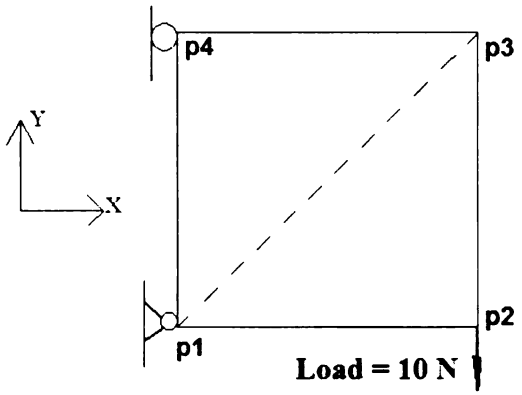

Figure 2 Example of assembling two 2-D plate models

The following example demonstrates the operation of the MMM assembly using Lagrange interpolation. The problem is defined as the following: a 2-D square plate (Fig.2) assembled from 2 identical triangular plates attached to a fixed joint at the lower left corner and its upper left corner restricted to move only vertically. The components are named "upper" and "lower" plates. A load of 10 Newtons pointing downward is applied to its lower right corner. Gravity and damping are ignored in the calculation. The system properties are listed in (Tab.2). The objective is to find the displacement of the points p2,

p3 and p4 at corners of the plate when the load is applied and the first three natural frequencies of the system without the load.

Different assembly cases are presented with different FEM component models, derived using the traditional FEM method. Three assembly cases have component models with geometrically compatible nodes. These models are assembled using both traditional FEM and MMM assembly. Two cases have component models with geometrically incompatible nodes. These models will only be assembled using MMM assembly.

Young's modulus	Poisson's ratio	Dimension	Density
$15\times10^{6}N/cm^{2}$	0.25	$6\times6\times0.1cm$	$2g/cm^{3}$

Table 2 Material Properties Of The Example

After the model is assembled, the static assembly can be solved using a traditional FEM approach. This approach requires the matrix $\mathbf{P}_a$ to be invertible after applying boundary conditions. The standard constraint matrix $\mathbf{S}$ and sufficient boundary conditions remove DOF from the unassembled model and makes $\mathbf{P}_a$ full rank, and therefore (7) can be solved. In system simulation, the output can be generated from (14) [7].

$$\mathbf{Y}_a(s) = \mathbf{G}_a(s)\mathbf{U}_a(s) \tag{14}$$

where

$$\mathbf{G}_a(s) = \mathbf{P}_a^{-1}(s)$$

For dynamics simulation, the following equation applies:

$$\mathbf{M}_a\ddot{\mathbf{Y}}_a+\mathbf{P}_a\mathbf{Y}_a=\mathbf{U}_a \tag{15}$$

where $\mathbf{Y}_a$ is the assembly output; $\mathbf{U}_a$ is the assembly input; $\mathbf{M}_a$ is the assembly mass matrix and $\mathbf{P}_a$ is the assembly stiffness matrix. In the examples, lumped mass matrices are used. The lumped mass matrix is traditional FEM mass matrices for thin plates. For free vibration, where $\mathbf{U}_a$ is zero vector, the natural frequencies ω are the solution of the following:

$$\mathbf{det}(\mathbf{P}_a-\omega^2\mathbf{M}_a)=\mathbf{0} \tag{16}$$

The procedure of system mass matrix assembly (17) is similar as system stiffness matrix assembly (7) in MMM. $\mathbf{M}_a$ can be used in (16) for solving the natural frequencies.

$$\mathbf{M}_a=\mathbf{S}^{\mathbf{T}}\begin{bmatrix}\mathbf{M}_{c1} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{M}_{c2} & \ddots & \vdots\\ \vdots & \ddots & \ddots & \mathbf{0}\\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{M}_{cn}\end{bmatrix}\mathbf{S} \tag{17}$$

3.2 Two components with geometrically compatible nodes

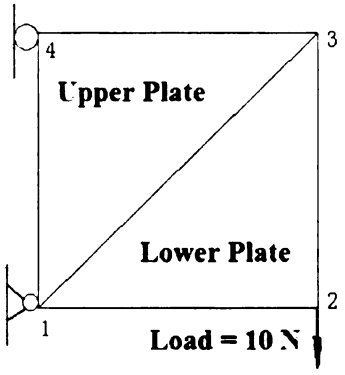

Figure 3 Assembly of Geometrically Compatible components (5 DOF)

Case one (Fig.3) is an assembly with two triangular components each using one linear triangular element. These two components have geometrically compatible nodes on the assembly interface. These two models can be assembled using both traditional FEM and MMM approaches

The component stiffness matrix for the upper plate (18) and lower plate (19) can be obtained using traditional FEM [1,8]:

$$\mathbf{P}_{C_1} = \begin{bmatrix} 8 & 0 & -8 & 2 & 0 & -2 \\ 0 & 3 & 3 & -3 & -3 & 0 \\ -8 & 3 & 11 & -5 & -3 & 2 \\ 2 & -3 & -5 & 11 & 3 & -8 \\ 0 & -3 & -3 & 3 & 3 & 0 \\ -2 & 0 & 2 & -8 & 0 & 8 \end{bmatrix} \times 10^5 (N/cm) \qquad (18)$$

$$\mathbf{P}_{C_2} = \begin{bmatrix} 3 & 0 & 0 & -3 & -3 & 3 \\ 0 & 8 & -2 & 0 & 2 & -8 \\ 0 & - & 8 & 0 & -8 & 2 \\ -3 & 0 & 0 & 3 & 3 & -3 \\ -3 & 2 & -8 & 3 & 11 & -5 \\ 3 & -8 & 2 & -3 & -5 & 11 \end{bmatrix} \times 10^5 (N/cm) \qquad (19)$$

Mass matrices for the upper and lower plates are the same:

$$\mathbf{M}_C = \begin{bmatrix} 1.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.2 \end{bmatrix} (g) \qquad (20)$$

Using the traditional FEM assembly, global stiffness matrix (21) and mass matrix (22) of the system can be constructed by direct superposition of component matrices.

$$\mathbf{P}_a = \begin{bmatrix} 11 & 0 & -8 & 2 & 0 & -5 & -3 & 3 \\ 0 & 11 & 3 & -3 & -5 & 0 & 2 & -8 \\ -8 & 3 & 11 & -5 & -3 & 2 & 0 & 0 \\ 2 & -3 & -5 & 11 & 3 & -8 & 0 & 0 \\ 0 & -5 & -3 & 3 & 11 & 0 & -8 & 2 \\ -5 & 0 & 2 & -8 & 0 & 11 & 3 & -3 \\ -3 & 2 & 0 & 0 & -8 & 3 & 11 & -5 \\ 3 & -8 & 0 & 0 & 2 & -3 & -5 & 11 \end{bmatrix} \times 10^5 (N/cm) \qquad (21)$$

$$\mathbf{M}_a = \begin{bmatrix} 2.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2 \end{bmatrix} (g) \qquad (22)$$

Using the MMM approach, the linear constraint matrix for this assembly can be constructed between the component nodes on assembly interface using (11).

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (23)$$

Using (18), (19) and (23) in (7) and using (22) and (23) in (17), the MMM assembly matrices are identical to the matrices assembled by the traditional FEM method (21) and (22). Because the assembly models are identical, the results from both assemblies are identical (Tab.3). MATLAB code for MMM assembly can be found in the appendix.

Displacement in x direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
Traditional 2 elements	-8.33	8.33	0
MMM 2 elements	-8.33	8.33	0
ANSYS 2 elements	-8.33	8.33	0
Displacement in y direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
Traditional 2 elements	-33.3	-25.0	-8.33
MMM 2 elements	-33.3	-25.0	-8.33
ANSYS 2 elements	-33.3	-25.0	-8.33
Modes			
Assembly	**First**	**Second**	**Third**
Traditional 2 elements	15.1kHz	31.6kHz	40.7kHz
MMM 2 elements	15.1kHz	31.6kHz	40.7kHz
ANSYS 2 elements	15.1kHz	31.6kHz	40.7kHz

TABLE 3 Simulation results with compatible Nodal meshing (1-1 element)

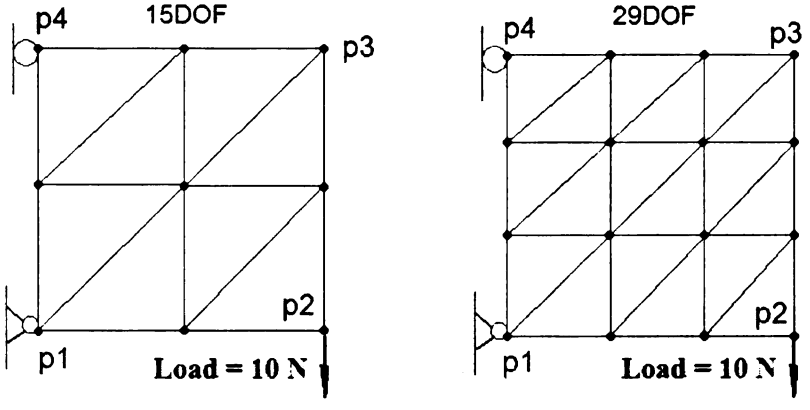

Figure 4 Aassemblies of Geometrically Compatible components (15 and 29 DOF)

Two additional compatible assembly models at higher resolution were constructed (Fig.4) at 15 and 29 DOF. Again, models assembled with MMM and traditional FEM were identical and produced identical displacement and natural frequency predictions (Tab.4). As the number of DOF increases, the solutions change to approach the "exact" continuous field solution [1]. MATLAB code for MMM assembly can be found in the appendix.

Displacement in x direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
All assembly 8 elements	-15.6	14.1	0
All assembly 18 elements	-20.4	18.4	0
Displacement in y direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
All assembly 8 elements	-51.4	-37.4	-15.6
All assembly 18 elements	-64.2	-45.8	-20.36
Modes			
Assembly	**First**	**Second**	**Third**
All assembly 8 elements	13.6kHz	25.4kHz	33.8kHz
All assembly 18 elements	12.4kHz	22.5kHz	30.5kHz

TABLE 4 Simulation Results With Compatible Nodal Meshing

3.3 Two components with geometrically incompatible nodes

An assembly with two triangular components with geometrically incompatible interface nodes is constructed (Fig.5). Node 4 on the lower plate does not have corresponding node at the same position from the upper plate. Upper plate has one linear triangular element and lower plate has four linear triangular elements. The stiffness and

mass matrices of individual components are derived using the traditional FEM method and have been validated in the geometrically compatible component model cases.

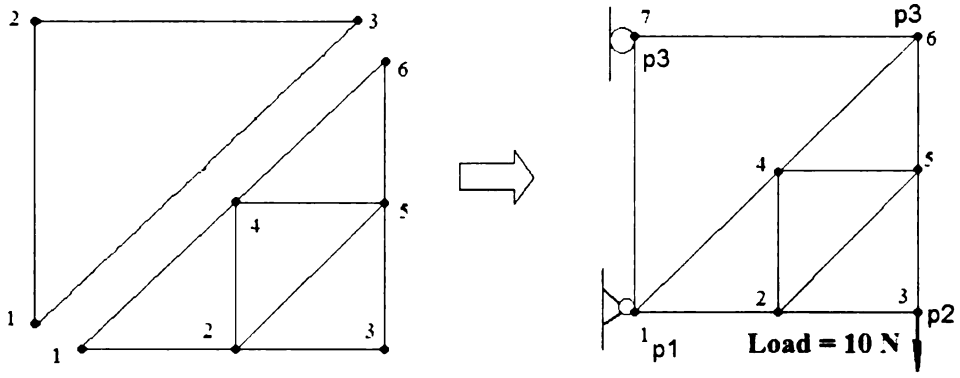

Figure 5 Assembly of Geometrically Compatible components (11 DOF)

Displacement in x direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
MMM assembly 13 elements	-16.6	20.5	0
Displacement in y direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
MMM assembly 13 elements	-58.9	-45.6	-20.2
Modes			
Assembly	**First**	**Second**	**Third**
MMM assembly 13 elements	13.4kHz	24.4kHz	32.4kHz

TABLE 5 Simulation results with incompatible Nodal meshing (1-4 element)

The component nodes from the lower plate on the assembly interface are selected as assembly nodes. According to the geometrical relationship between the nodes on the assembly interface of the two components, the global constraint matrix can be constructed. Using linear constraint matrix to assemble system stiffness matrix $\mathbf{P}_a$ and mass matrix $\mathbf{M}_a$, and applying boundary conditions and loads in equation (14) and (17),

the output of the nodes and the system natural frequencies can be computed (Tab.5). MATLAB code for the simulation can be found in the appendix.

An additional incompatible assembly with 23 DOF is constructed (Fig.6). In this case, 3 nodes on the component models do not have corresponding nodes on the other components. The component nodes from the upper plate on the assembly interface are selected as assembly nodes. According to the geometrical relationship between the nodes on the assembly interface of the two components, the outputs of component nodes and the assembly nodes on the assembly interface can be expressed as:

$$\begin{bmatrix} U_{1x} \\ U_{1y} \\ U_{4x} \\ U_{4y} \\ U_{6x} \\ U_{6y} \end{bmatrix}_L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{16} & 0 & \frac{27}{48} & 0 & \frac{27}{48} & 0 & -\frac{1}{16} & 0 \\ 0 & -\frac{1}{16} & 0 & \frac{27}{48} & 0 & \frac{27}{48} & 0 & -\frac{1}{16} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_{1x} \\ U_{1y} \\ U_{2x} \\ U_{2y} \\ U_{6x} \\ U_{6y} \\ U_{10x} \\ U_{10y} \end{bmatrix}_A \quad \textit{and}$$

$$\begin{bmatrix} U_{1x} \\ U_{1y} \\ U_{2x} \\ U_{2y} \\ U_{6x} \\ U_{6y} \\ U_{10x} \\ U_{10y} \end{bmatrix}_U = \mathbf{I} \begin{bmatrix} U_{1x} \\ U_{1y} \\ U_{2x} \\ U_{2y} \\ U_{6x} \\ U_{6y} \\ U_{10x} \\ U_{10y} \end{bmatrix}_A \tag{25}$$

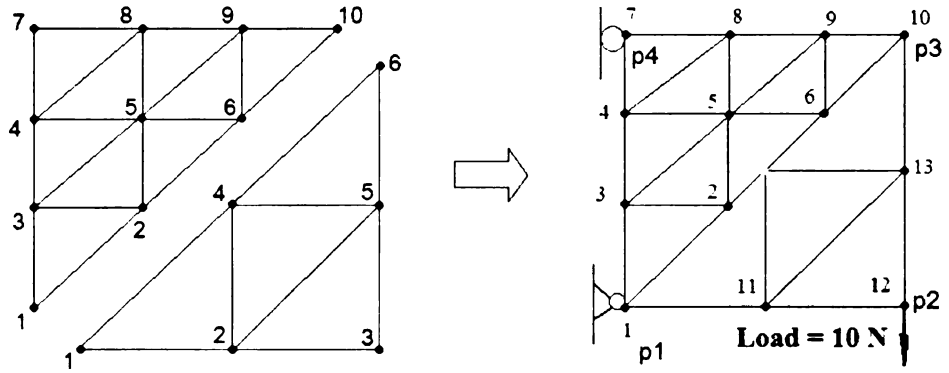

Figure 6 Assembly of Geometrically Compatible components (23 DOF)

The global constraint matrix can be constructed according to the nodal numbering and not shown in this paper due to its size. The assembly approach is similar as the previous part. The simulation results are shown in (Tab.6). MATLAB code for the simulation can be found in the appendix.

Displacement in x direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
MMM assembly 13 elements	-16.59	20.48	0
Displacement in y direction (10^{-6} cm)			
Assembly	**p2**	**p3**	**p4**
MMM assembly 13 elements	-58.88	-45.63	-20.16
Modes			
Assembly	**First**	**Second**	**Third**
MMM assembly 13 elements	13.4kHz	24.4kHz	32.4kHz

TABLE 6 Simulation results with incompatible Nodal meshing (9-4 element)

3.4 MMM assembly validation

The MMM successfully assembled geometrically compatible and incompatible component models without model reformulation. When the component models are

geometrically compatible, MMM generates the same assembly model and solutions as the traditional FEM method. When component models are geometrically incompatible, the model generated by MMM has no comparable FEM model. Solutions from the MMM model lie between the solutions generated by traditional FEM assembly at higher and lower model resolution.

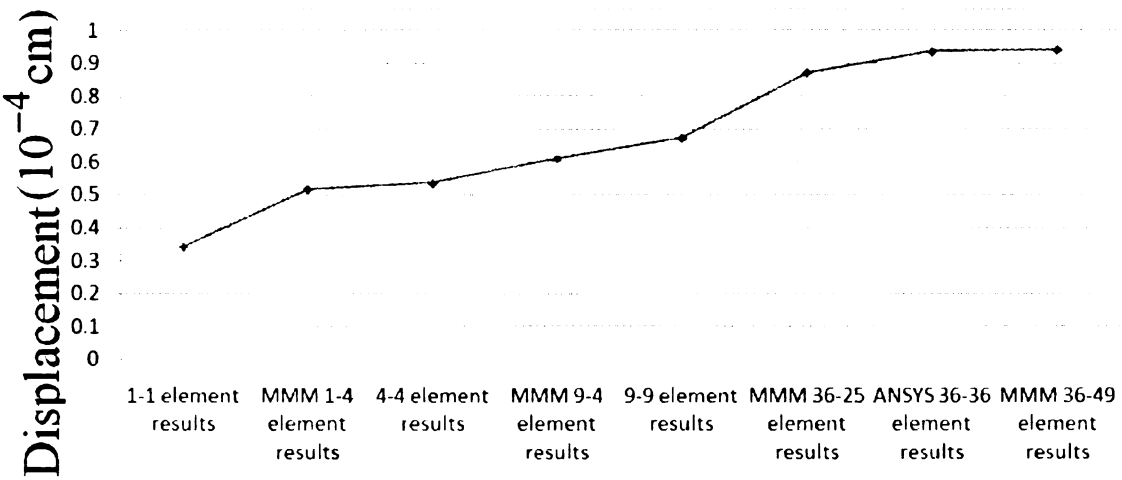

Figure 7 Displacement of p2 Using Component Models With Different Resolutions

The MMM generates better assembly solutions when the component models have higher resolution. This demonstrates the same behavior as the traditional FEM assembly method [1]. In the example models, displacement results converge to the solution from a model with the highest resolution monotonically as the meshing resolution increases. Figure 7 shows displacement of p2, where the load is applied and represents the assembly stiffness, converges when the assembly has higher resolution component models.

Displacement in x (10^{-6} cm)			
Assembly method	**p2**	**p3**	**p4**
1-1 elements results	-8.33	83.3	0
MMM 1-4 elements results	-9	7	0
4-4 elements results	-15.6	14.1	0
MMM 9-4 elements results	-16.6	20.5	0
9-9 elements results	-20.4	18.4	0
ANSYS 36-36 elements results	-29.22	26.89	0
Displacement in y (10^{-6} cm)			
Assembly method	**p2**	**p3**	**p4**
1-1 elements results	-33.3	-25	-8.33
MMM 1-4 elements results	-50.9	-30.6	-96.3
4-4 elements results	-51.4	-37.4	-15.6
MMM 9-4 elements results	-58.9	-45.6	-20.2
9-4 elements results	-64.2	-45.8	-20.4
ANSYS 36-36 elements results	-88.8	-61.9	-29.2
Displacement - magnitude (10^{-6} cm)			
Assembly method	**p2**	**p3**	**p4**
1-1 elements results	34.4	26.4	8.33
MMM 1-4 elements results	51.8	31.4	9.63
4-4 elements results	53.7	39.9	15.57
MMM 9-4 elements results	61.2	50.0	20.16
9-4 elements results	67.4	49.4	20.36
ANSYS 36-36 elements results	93.5	67.5	29.2

TABLE 7 Nodal Displacement Results from Different Assemblies

The nodal displacement values and natural frequency results of all 5 cases, together with the reference result from a refined traditional FEM assembly using 72 elements, are shown in (Tab.8) and (Tab.9). There is unexpected behavior of node p3. The nodal displacement solution at p3 does not converge monotonically when using MMM. This unexpected behavior could be the result of a low-resolution mesh in these models combined with the use of p3 in the interpolation on the assembly interface. There are also

a few non-monotonic convergent natural frequencies in a few of the MMM assemblies with geometrically incompatible component models. These behaviors will be investigated in future work.

Assembly method	1st Mode	2nd Mode	3rd Mode
1-1 elements results	15.1kHz	31.6kHz	40.7kHz
MMM 1-4 elements results	13.6k Hz	25.5kHz	32.9kHz
4-4 elements results	13.6kHz	25.4kHz	33.8kHz
MMM 9-4 elements results	13.4kHz	24.4kHz	32.4kHz
9-9 elements results	12.4kHz	22.5kHz	30.5kHz
ANSYS 36-36 elements results	10.6kHz	18.4kHz	25.4kHz

TABLE 8 Modal Analysis Results

Component models	Displacement results from different orders of Lagrange Interpolation(10^{-6} cm)				
	0th	1st	2nd	3rd	4th
1+1 elements	15.3	34.4	n/a	n/a	n/a
1+4 elements	20.1	43.0	51.7	n/a	n/a
4+4 elements	20.1	51.0	53.7	n/a	n/a
9+4 elements	20.1	56.0	60.0	61.2	93.6
9+9 elements	25.1	61.8	67.1	67.4	67.4

TABLE 9 Nodal Displacement at p2 Using Different Orders of Lagrange Interpolation

The order of the linear constraint interpolation on the assembly surface can be chosen independently because the assembly nodes can be defined arbitrarily. As a result, the order of the interpolation will affect the overall stiffness of the system. In this example, the static cases discussed previously are assembled using Lagrange interpolation with different orders. All the Lagrange interpolation use equally distributed nodes on the assembly interface. The displacement results for p2, where the load is applied and represents the assembly stiffness, are shown as (Fig.8) and (Tab.9).

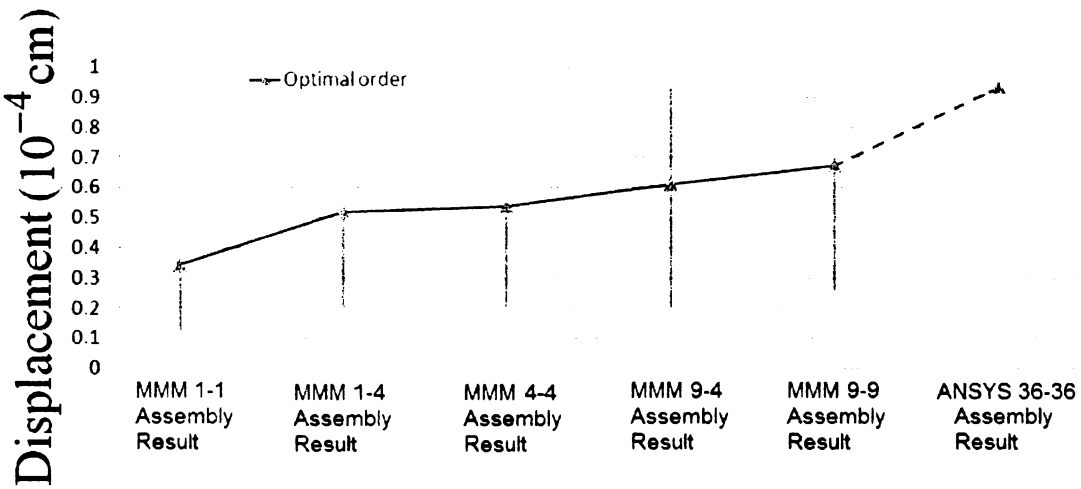

Figure 8 Nodal displacement magnitude of P2 using different methods

As the order of Lagrange interpolation increases, the overall stiffness of the system decreases. This behavior demonstrates a property similar to the traditional assembly. In traditional FEM, when the discretization resolution increases, the system stiffness decreases and the solution approaches the true continuum solution [1]. When the order of interpolation is higher than the order associated with the largest number of nodes on a single component's interface surface, the assembly will have too many degrees of freedom to yield a solution. To get the most accurate assembly solution with a given set of component models, the optimal order of linear interpolation along the assembly surface appears to be the order associated with the largest number of nodes on assembly surface on any single component.

Conclusion

The Modular Modeling Method can be used to assemble FEM models efficiently and easily even when they have incompatible node geometries. It does not require re-meshing of incompatible nodes. Using MMM assembly, the proprietary internal information from the components is not required when the FEM component models do not have compatible meshing at the assembly interface.

A standardized form of the linear constraint matrix has been developed. Lagrange interpolation is used to mathematically describe the output relationship between assembly nodes and component nodes. This method interpolates the component nodal output using only the assembly nodal output on the assembly interface. The assembly nodes can be selected as any combination of points on the assembly interface, so it provides more flexibility when compared to traditional FEM. For optimal results, the order of Lagrange interpolation should be selected as the order associated with the number of nodes of the single component with the most nodes on the assembly surface.

MMM assembly generates results comparable to the results generated by traditional FEM models with similar meshing. When the component models are geometrically compatible, the static and dynamics simulation results generated by MMM assembly are the same as the traditional FEM assembly. When the component models are geometrically incompatible, the static and dynamics simulation results generated by MMM assembly usually lie between the result generated by traditional FEM assembly of

compatible lower resolution meshes and the result generated by traditional FEM assembly of compatible higher resolution meshes. The trend shows that the accuracy of MMM becomes better when the components have higher resolution meshes. Convergence of MMM results has similar behavior to traditional FEM results. When combined with higher resolution mesh, MMM generates better accuracy and therefore can be used in engineering practice.

APPENDIX MATLAB Codes

List of MATLAB Codes

Name	Function
Plate11_MMM_static.m	Solves example 3.1.1 using MMM
Plate44_MMM_static.m	Solves example 3.1.2 using MMM
Plate99_MMM_static.m	Solves example 3.1.3 using MMM
Plate14_MMM_static.m	Solves example 3.1.4 using MMM
Plate49_MMM_static.m	Solves example 3.1.5 using MMM
Plate14_MMM_dynamics.m	Solves example 3.2 using MMM (1+4 elements, 11DOF)
Plate49_MMM_dynamics.m	Solves example 3.2 using MMM (4+9 elements, 23DOF)
LinearTriangleElementMassMatrix.m	Generates lumped mass matrix for a 2D triangle FEM element

Plate11_MMM_static.m for solving example 3.1.1 using Modular Modeling Method

```
%       plate11_MMM_static.m
%
%       MATLAB code for solving example 3.1.1 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%
%       Code by Zhen Ren
%       Sept 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003]
%       and LinearTriangleAssemble [P. I. Kattan, 2003] are called

clc;
clear;

%       generate stiffness matrix for P1
k1=zeros(6);
k1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 6, 0, 6, 6, 1);

%       generate stiffness matrix for P2
k2=zeros(6);
k2=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 6, 6, 0, 6, 1);

%       generate un-assembled system stiffness matrix
P=[k1, zeros(6,6); zeros(6,6), k2];

%       generate constraint matrix according to the geometric relationship between
%       the component nodes and assembly nodes
S=[1 0 0 0 0 0 0 0 ; 0 1 0 0 0 0 0 0; 0 0 1 0 0 0 0 0; 0 0 0 1 0 0 0 0; 0 0 0 0 1 0 0 0; 0 0 0 0 0 1 0 0;
    1 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0; 0 0 0 0 1 0 0 0; 0 0 0 0 0 1 0 0; 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 1];

%       Assemble global stiffness matrix using MMM
PA=S'*P*S;

%       Apply boundary condition for nodal force load
f=[0 0 0 -10 0 0 0 0];
f=f';
```

```
%          Apply boundary condition for displacement (fixed points)
for (i=1: 8)
     PA(1, i)=0;
     PA(i, 1)=0;
     PA(2, i)=0;
     PA(i, 2)=0;
     PA(7,i)=0;
     PA(i,7)=0;
end
PA(1, 1)=1;
PA(2, 2)=1;
PA(7,7)=1;

%          Solve for displacement results in x, and y direction
D=(inv(PA)*f)

%          Solve for displacement magnitude
for (i=1:4)
     Dx(i)=D(2*i-1);
     Dy(i)=D(2*i);
     DD(i)=((Dx(i))^2+(Dy(i))^2)^(1/2);
end

%          Display the displacement magnitude
DD
```

Plate44_MMM_static.m for solving example 3.1.2 using Modular Modeling Method

```
%       plate44_MMM_static.m
%
%       MATLAB code for solving example 3.1.2 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%
%       Code by Zhen Ren
%       Sept 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003]
%       and LinearTriangleAssemble [P. I. Kattan, 2003] are called

clc;
clear;

%       generate stiffness matrix "A" for lower plate
A=zeros(12);
k1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 3, 3, 3, 0, 1);
A=LinearTriangleAssemble(A, k1, 1, 4, 2);
k2=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 0, 6, 3, 6, 0, 1);
A=LinearTriangleAssemble(A, k2, 2, 5, 3);
k3=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 0, 3, 3, 6, 3, 1);
A=LinearTriangleAssemble(A, k3, 2, 4, 5);
k4=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 3, 6, 6, 6, 3, 1);
A=LinearTriangleAssemble(A, k4, 4, 6, 5);

%       generate stiffness matrix "B" for upper plate
B=zeros(12);
k5=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 0, 3, 3, 3, 1);
B=LinearTriangleAssemble(B, k5, 1, 2, 3);
k6=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 3, 0, 6, 3, 6, 1);
B=LinearTriangleAssemble(B, k6, 2, 4, 5);
k7=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 3, 3, 6, 3, 3, 1);
B=LinearTriangleAssemble(B, k7, 2, 5, 3);
k8=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 3, 3, 6, 6, 6, 1);
B=LinearTriangleAssemble(B, k8, 3, 5, 6);

%       generate un-assembled system stiffness matrix
```

```
P=[-A, zeros(12,12); zeros(12,12), -B];

%       generate constraint matrix according to the geometric relationship between
%       the component nodes and assembly nodes
S=[eye(12), zeros(12, 6); zeros(12, 18)];
S(13, 1)=1;
S(14, 2)=1;
S(15, 13)=1;
S(16, 14)=1;
S(17, 7)=1;
S(18, 8)=1;
S(19, 15)=1;
S(20, 16)=1;
S(21, 17)=1;
S(22, 18)=1;
S(23, 11)=1;
S(24, 12)=1;

%       Assemble global stiffness matrix using MMM
PA=S'*P*S;

%       Apply boundary condition for nodal force load
f=[0 0 0 0 0 -10 0 0 0 0 0 0 0 0 0 0 0 0];
f=f';

%       Apply boundary condition for displacement (fixed points)
for (i=1: 18)
    PA(1, i)=0;
    PA(i, 1)=0;
    PA(2, i)=0;
    PA(i, 2)=0;
    PA(15, i)=0;
    PA(i, 15)=0;
end
PA(1, 1)=1;
PA(2, 2)=1;
PA(15,15)=1;
PA

%       Solve for displacement results in x, and y direction
D=(inv(PA)*f)
```

```
%          Solve for displacement magnitude
for (i=1:9)
    Dx(i)=D(2*i-1);
    Dy(i)=D(2*i);
    DD(i)=((Dx(i))^2+(Dy(i))^2)^(1/2);
end

%          Display the displacement magnitude
DD
```

Plate99_MMM_static.m for solving example 3.1.3 using Modular Modeling Method

```
%       plate99_MMM_static.m
%
%       MATLAB code for solving example 3.1.3 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%
%       Code by Zhen Ren
%       Sept 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003]
%       and LinearTriangleAssemble [P. I. Kattan, 2003] are called

clc;
clear;

%       generate stiffness matrix "A" for upper plate
K=zeros(20);
k1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 0, 2, 2, 2, 1);
K=LinearTriangleAssemble(K, k1, 1, 3, 2);
k2=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 2, 2, 4, 2, 2, 1);
K=LinearTriangleAssemble(K, k2, 3, 5, 2);
k3=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 2, 2, 4, 4, 4, 1);
K=LinearTriangleAssemble(K, k3, 2, 5, 6);
k4=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 2, 0, 4, 2, 4, 1);
K=LinearTriangleAssemble(K, k4, 3, 4, 5);
k5=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 4, 0, 6, 2, 6, 1);
K=LinearTriangleAssemble(K, k5, 4, 7, 8);
k6=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 4, 2, 6, 2, 4, 1);
K=LinearTriangleAssemble(K, k6, 4, 8, 5);
k7=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 4, 2, 6, 4, 6, 1);
K=LinearTriangleAssemble(K, k7, 5, 8, 9);
k8=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 4, 4, 6, 4, 4, 1);
K=LinearTriangleAssemble(K, k8, 5, 9, 6);
k9=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 4, 4, 6, 6, 6, 1);
K=LinearTriangleAssemble(K, k9, 6, 9, 10);
A=K

%       generate stiffness matrix "B" for lower plate
```

```
K=zeros(20);
k10=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 2, 2, 2, 0, 1);
K=LinearTriangleAssemble(K, k10, 1, 5, 2);
k11=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 0, 2, 2, 4, 2, 1);
K=LinearTriangleAssemble(K, k11, 2, 5, 6);
k12=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 0, 4, 2, 4, 0, 1);
K=LinearTriangleAssemble(K, k12, 2, 6, 3);
k13=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 0, 4, 2, 6, 2, 1);
K=LinearTriangleAssemble(K, k13, 3, 6, 7);
k14=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 0, 6, 2, 6, 0, 1);
K=LinearTriangleAssemble(K, k14, 3, 7, 4);
k15=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 2, 4, 4, 4, 2, 1);
K=LinearTriangleAssemble(K, k15, 5, 8, 6);
k16=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 2, 4, 4, 6, 4, 1);
K=LinearTriangleAssemble(K, k16, 6, 8, 9);
k17=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 2, 6, 4, 6, 2, 1);
K=LinearTriangleAssemble(K, k17, 6, 9, 7);
k18=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 4, 6, 6, 6, 4, 1);
K=LinearTriangleAssemble(K, k18, 8, 10, 9);
B=K

%       generate un-assembled system stiffness matrix
P=[A, zeros(20); zeros(20), B];

%       generate constraint matrix according to the geometric relationship between
%       the component nodes and assembly nodes
S=[eye(20), zeros(20, 12); zeros(20,32)]
S(21,1)=1; S(22,2)=1;
S(23,21)=1; S(24,22)=1;
S(25,23)=1; S(26,24)=1;
S(27,25)=1; S(28,26)=1;
S(29, 3)=1; S(30,4)=1;
S(31, 27)=1; S(32, 28)=1;
S(33, 29)=1; S(34,30)=1;
S(35,11)=1; S(36,12)=1;
S(37,31)=1; S(38,32)=1;
S(39, 19)=1; S(40,20)=1

%       Assemble global stiffness matrix using MMM
PA=S'*P*S;
```

```
%          Apply boundary condition for nodal force load
f=[0 0 0 0 0, 0 0 0 0 0, 0 0 0 0 0, 0 0 0 0 0, 0 0 0 0 0, 10 0 0 0 0, 0 0];
f=f';

%          Apply boundary condition for displacement (fixed points)
for (i=1: 32)
      PA(1, i)=0;
      PA(i, 1)=0;
      PA(2, i)=0;
      PA(i, 2)=0;
      PA(13, i)=0;
      PA(i, 13)=0;
end
PA(1, 1)=1;
PA(2, 2)=1;
PA(13,13)=1;
PA

%          Solve for displacement results in x, and y direction
D=(inv(PA)*f)

%          Solve for displacement magnitude
for (i=1:16)
      Dx(i)=D(2*i-1);
      Dy(i)=D(2*i);
      DD(i)=((Dx(i))^2+(Dy(i))^2)^(1/2);
end

%         Display the displacement magnitude
DD
```

Plate14_MMM_static.m for solving example 3.1.4 using Modular Modeling Method

```
%       plate14_MMM_static.m
%
%       MATLAB code for solving example 3.1.4 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%
%       Code by Zhen Ren
%       Sept 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003]
%       and LinearTriangleAssemble [P. I. Kattan, 2003] are called

clc;
clear;

%       generate stiffness matrix "P1" for the upper plate
P1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 0, 6, 6, 6, 1);

%       generate stiffness matrix "P2" for the lower plate
P2=zeros(12);
k1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 3, 3, 3, 0, 1);
P2=LinearTriangleAssemble(P2, k1, 1, 4, 2);
k2=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 0, 3, 3, 6, 3, 1);
P2=LinearTriangleAssemble(P2, k2, 2, 4, 5);
k3=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 0, 6, 3, 6, 0, 1);
P2=LinearTriangleAssemble(P2, k3, 2, 5, 3);
k4=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 3, 6, 6, 6, 3, 1);
P2=LinearTriangleAssemble(P2, k4, 4, 6, 5);

%       generate un-assembled system stiffness matrix
P=[P1, zeros(6,12); zeros(12,6), P2]

%       generate constraint matrix according to the geometric relationship between
%       the component nodes and assembly nodes
S=[zeros(6, 14); eye(12), zeros(12,2)];
S(1,1)=1;
S(2,2)=1;
S(3, 13)=1;
```

```
S(4, 14)=1;
S(5, 11)=1;
S(6, 12)=1;

%          Assemble global stiffness matrix using MMM
PA=S'*P*S

%          Apply boundary condition for nodal force load
f=[0 0 0 0 0 10 0 0 0 0 0 0 0 0 ];
f=f';

%          Apply boundary condition for displacement (fixed points)
for (i=1: 14)
     PA(1, i)=0;
     PA(i, 1)=0;
     PA(2, i)=0;
     PA(i, 2)=0;
     PA(13,i)=0;
     PA(i,13)=0;
end
PA(1, 1)=1;
PA(2, 2)=1;
PA(13,13)=1;

%          Solve for displacement results in x, and y direction
D=(inv(PA)*f)

%          Solve for displacement magnitude
for (i=1:7)
     Dx(i)=D(2*i-1);
     Dy(i)=D(2*i);
     DD(i)=((Dx(i))^2+(Dy(i))^2)^(1/2);
end

%          Display the displacement magnitude
DD
```

Plate49_MMM_static.m for solving example 3.1.5 using Modular Modeling Method

```
%       plate49_MMM_static.m
%
%       MATLAB code for solving example 3.1.5 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%
%       Code by Zhen Ren
%       Sept 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003]
%       and LinearTriangleAssemble [P. I. Kattan, 2003] are called

clc;
clear;

%       generate stiffness matrix "P1" for the upper plate
K=zeros(20);
k1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 0, 2, 2, 2, 1);
K=LinearTriangleAssemble(K, k1, 1, 3, 2);
k2=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 2, 2, 4, 2, 2, 1);
K=LinearTriangleAssemble(K, k2, 3, 5, 2);
k3=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 2, 2, 4, 4, 4, 1);
K=LinearTriangleAssemble(K, k3, 2, 5, 6);
k4=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 2, 0, 4, 2, 4, 1);
K=LinearTriangleAssemble(K, k4, 3, 4, 5);
k5=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 4, 0, 6, 2, 6, 1);
K=LinearTriangleAssemble(K, k5, 4, 7, 8);
k6=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 4, 2, 6, 2, 4, 1);
K=LinearTriangleAssemble(K, k6, 4, 8, 5);
k7=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 4, 2, 6, 4, 6, 1);
K=LinearTriangleAssemble(K, k7, 5, 8, 9);
k8=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 2, 4, 4, 6, 4, 4, 1);
K=LinearTriangleAssemble(K, k8, 5, 9, 6);
k9=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 4, 4, 4, 6, 6, 6, 1);
K=LinearTriangleAssemble(K, k9, 6, 9, 10);
P1=K;

%       generate stiffness matrix "P2" for the lower plate
```

```
K=zeros(12);
k1=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 0, 0, 3, 3, 3, 0, 1);
K=LinearTriangleAssemble(K, k1, 1, 4, 2);
k2=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 0, 6, 3, 6, 0, 1);
K=LinearTriangleAssemble(K, k2, 2, 5, 3);
k3=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 0, 3, 3, 6, 3, 1);
K=LinearTriangleAssemble(K, k3, 2, 4, 5);
k4=LinearTriangleElementStiffness(15000000, 0.25, 0.1, 3, 3, 6, 6, 6, 3, 1);
K=LinearTriangleAssemble(K, k4, 4, 6, 5);
P2=K;

%       generate un-assembled system stiffness matrix
P=[P1, zeros(20,12); zeros(12,20), P2];

%       generate constraint matrix according to the geometric relationship between
%       the component nodes and assembly nodes
S=[eye(20),zeros(20,6); zeros(12,26)];
S(21,1)=1;
S(22,2)=1;
S(23,21)=1;
S(24,22)=1;
S(25,23)=1;
S(26,24)=1;
S(27,1)=-(1/16);
S(28,2)=-(1/16);
S(27,3)=(27/48);
S(28,4)=(27/48);
S(27,11)=(27/48);
S(28,12)=(27/48);
S(27,19)=-(1/16);
S(28,20)=-(1/16);
S(29,25)=1;
S(30,26)=1;
S(31,19)=1;
S(32,20)=1;

%       Assemble global stiffness matrix using MMM
PA=S'*P*S;

%       Apply boundary condition for nodal force load
f=[0 0 0 0 0, 0 0 0 0 0, 0 0 0 0 0, 0 0 0 0 0, 0 0 0 10 0, 0];
```

```
f=f';

%          Apply boundary condition for displacement (fixed points)
for (i=1: 26)
     PA(1, i)=0;
     PA(i, 1)=0;
     PA(2, i)=0;
     PA(i, 2)=0;
     PA(13,i)=0;
     PA(i,13)=0;
end
PA(1, 1)=1;
PA(2, 2)=1;
PA(13,13)=1;

%          Solve for displacement results in x, and y direction
D=(inv(PA)*f)

%          Solve for displacement magnitude
for (i=1:13)
     Dx(i)=D(2*i-1);
     Dy(i)=D(2*i);
     DD(i)=((Dx(i))^2+(Dy(i))^2)^(1/2);
end

%          Display the displacement magnitude
DD
```

Plate14_MMM_dynamics.m for solving example 3.2 using Modular Modeling Method (1+4 elements, 11 DOF)

```
%       plate14_MMM_dynamics.m
%
%       MATLAB code for solving example 3.2 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%       (1+4 elements, 11 DOF)
%
%       Code by Zhen Ren
%       Nov 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003],
%       LinearTriangleAssemble [P. I. Kattan, 2003] and
%       LinearTriangleElementMassMatrix are called
%

clc
clear

%       generate stiffness matrix "Ku"  and mass matrix "Mu" for the upper plate
Ku=zeros(6);
k1=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0, 0, 0.06, 0.06, 0.06, 1);
Ku=LinearTriangleAssemble(Ku, k1, 1, 2, 3);
Mu=zeros(6);
m1=LinearTriangleElementMassMatrix(2000, 0.001, 0, 0, 0, 0.06, 0.06, 0.06);
Mu=LinearTriangleAssemble(Mu, m1, 1, 2, 3);

%       generate stiffness matrix "Kl" and mass matrix "Ml" for the upper plate
Kl=zeros(12);
Ml=zeros(12);
k1=LinearTriangleElementStiffness(15e10, 0.25, 0.001, 0, 0, 0.03, 0.03, 0.03, 0, 1);
Kl=LinearTriangleAssemble(Kl, k1, 1, 4, 2);
m1=LinearTriangleElementMassMatrix(2000, 0.001, 0, 0, 0.03, 0.03, 0.03, 0);
Ml=LinearTriangleAssemble(Ml, m1, 1, 4, 2);
k2=LinearTriangleElementStiffness(15e10, 0.25, 0.001, 0.03, 0, 0.06, 0.03, 0.06, 0, 1);
Kl=LinearTriangleAssemble(Kl, k2, 2, 5, 3);
m2=LinearTriangleElementMassMatrix(2000, 0.001, 0.03, 0, 0.06, 0.03, 0.06, 0);
Ml=LinearTriangleAssemble(Ml, m2, 2, 5, 3);
```

```
k3=LinearTriangleElementStiffness(15e10, 0.25, 0.001, 0.03, 0, 0.03, 0.03, 0.06, 0.03, 1);
Kl=LinearTriangleAssemble(Kl, k3, 2, 4, 5);
m3=LinearTriangleElementMassMatrix(2000, 0.001, 0.03, 0, 0.03, 0.03, 0.06, 0.03);
Ml=LinearTriangleAssemble(Ml, m3, 2, 4, 5);
k4=LinearTriangleElementStiffness(15e10, 0.25, 0.001, 0.03, 0.03, 0.06, 0.06, 0.06, 0.03, 1);
Kl=LinearTriangleAssemble(Kl, k4, 4, 6, 5);
m4=LinearTriangleElementMassMatrix(2000, 0.001, 0.03, 0.03, 0.06, 0.06, 0.06, 0.03);
Ml=LinearTriangleAssemble(Ml, m4, 4, 6, 5);

%       generate stiffness matrix and mass matrix for the un-assembled system
PK=-[Ku, zeros(6,12); zeros(12,6), Kl];
PM=[Mu, zeros(6,12); zeros(12,6), Ml];

%       generate constraint matrix according to the geometric relationship between
%       the component nodes and assembly nodes
S=[zeros(6, 14); eye(12), zeros(12,2)];
S(1,1)=1;
S(2,2)=1;
S(3, 13)=1;
S(4, 14)=1;
S(5, 11)=1;
S(6, 12)=1;

%       Assemble global stiffness matrix and global mass matrix using MMM
PKA=S'*PK*S;
PMA=S'*PM*S;

%       Apply boundary condition for displacement (fixed points)
for i=1:10
    for j=1:10
        Mc(i,j)=PMA(i+2,j+2);
        Kc(i,j)=PKA(i+2,j+2);
    end
    Mc(11,i)=PMA(14,i+2);
    Mc(i,11)=PMA(i+2, 14);
    Kc(11,i)=PKA(14,i+2);
    Kc(i,11)=PKA(i+2, 14);
end
    Mc(11,11)=PMA(14, 14);
    Kc(11,11)=PKA(14,14);
```

```
%           solve for natl freq (in rad/s)
[V,D]=eig(Kc, Mc);
V;
D;

%           display natl freq results in Hz
for j=1:1:11
     natl_freq(j)=sqrt(D(j,j))/(2*pi);
end
natl_freq
```

Plate49_MMM_dynamics.m for solving example 3.2 using Modular Modeling Method (4+9 elements, 23 DOF)

```
%       plate49_MMM_dynamics.m
%
%       MATLAB code for solving example 3.2 in
%       MODULAR MODEL ASSEMBLY FROM FINITE ELEMENT
%       COMPONENT MODELS
%       using the Modular Modeling Method
%       (4+9 elements, 23 DOF)
%
%       Code by Zhen Ren
%       Nov 2007
%
%       *functions of LinearTriangleElementStiffness [P. I. Kattan, 2003],
%       LinearTriangleAssemble [P. I. Kattan, 2003] and
%       LinearTriangleElementMassMatrix are called
%

clc;
clear;

%       generate stiffness matrix "Ku"   and mass matrix "Mu" for the upper plate
Ku=zeros(20);
Mu=zeros(20);
k1=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0, 0, 0.02, 0.02, 0.02, 1);
Ku=LinearTriangleAssemble(Ku, k1, 1, 3, 2);
m1=LinearTriangleElementMassMatrix(2000, 0.001, 0, 0, 0, 0.02, 0.02, 0.02);
Mu=LinearTriangleAssemble(Mu, m1, 1, 3, 2);
k2=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0.02, 0.02, 0.04, 0.02, 0.02, 1);
Ku=LinearTriangleAssemble(Ku, k2, 3, 5, 2);
m2=LinearTriangleElementMassMatrix(2000, 0.001, 0, 0.02, 0.02, 0.04, 0.02, 0.02);
Mu=LinearTriangleAssemble(Mu, m2, 3, 5, 2);
k3=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.02, 0.02, 0.02, 0.04, 0.04, 0.04, 1);
Ku=LinearTriangleAssemble(Ku, k3, 2, 5, 6);
m3=LinearTriangleElementMassMatrix(2000, 0.001, 0.02, 0.02, 0.02, 0.04, 0.04, 0.04);
Mu=LinearTriangleAssemble(Mu, m3, 2, 5, 6);
k4=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0.02, 0, 0.04, 0.02, 0.04, 1);
Ku=LinearTriangleAssemble(Ku, k4, 3, 4, 5);
m4=LinearTriangleElementMassMatrix(2000, 0.001, 0, 0.02, 0, 0.04, 0.02, 0.04);
Mu=LinearTriangleAssemble(Mu, m4, 3, 4, 5);
```

```
k5=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0.04, 0, 0.06, 0.02, 0.06, 1);
Ku=LinearTriangleAssemble(Ku, k5, 4, 7, 8);
m5=LinearTriangleElementMassMatrix(2000, 0.001,   0, 0.04, 0, 0.06, 0.02, 0.06);
Mu=LinearTriangleAssemble(Mu, m5, 4, 7, 8);
k6=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0.04, 0.02, 0.06, 0.02, 0.04, 1);
Ku=LinearTriangleAssemble(Ku, k6, 4, 8, 5);
m6=LinearTriangleElementMassMatrix(2000, 0.001,   0, 0.04, 0.02, 0.06, 0.02, 0.04);
Mu=LinearTriangleAssemble(Mu, m6, 4, 8, 5);
k7=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.02, 0.04, 0.02, 0.06, 0.04, 0.06, 1);
Ku=LinearTriangleAssemble(Ku, k7, 5, 8, 9);
m7=LinearTriangleElementMassMatrix(2000, 0.001,   0.02, 0.04, 0.02, 0.06, 0.04, 0.06);
Mu=LinearTriangleAssemble(Mu, m7, 5, 8, 9);
k8=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.02, 0.04, 0.04, 0.06, 0.04, 0.04, 1);
Ku=LinearTriangleAssemble(Ku, k8, 5, 9, 6);
m8=LinearTriangleElementMassMatrix(2000, 0.001, 0.02, 0.04, 0.04, 0.06, 0.04, 0.04);
Mu=LinearTriangleAssemble(Mu, m8, 5, 9, 6);
k9=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.04, 0.04, 0.04, 0.06, 0.06, 0.06, 1);
Ku=LinearTriangleAssemble(Ku, k9, 6, 9, 10);
m9=LinearTriangleElementMassMatrix(2000, 0.001, 0.04, 0.04, 0.04, 0.06, 0.06, 0.06);
Mu=LinearTriangleAssemble(Mu, m9, 6, 9, 10);

%          generate stiffness matrix "Kl"   and mass matrix "Ml" for the lower plate
Kl=zeros(12);
Ml=zeros(12);
k1=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0, 0, 0.03, 0.03, 0.03, 0, 1);
Kl=LinearTriangleAssemble(Kl, k1, 1, 4, 2);
m1=LinearTriangleElementMassMatrix(2000, 0.001, 0, 0, 0.03, 0.03, 0.03, 0);
Ml=LinearTriangleAssemble(Ml, m1, 1, 4, 2);
k2=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.03, 0, 0.06, 0.03, 0.06, 0, 1);
Kl=LinearTriangleAssemble(Kl, k2, 2, 5, 3);
m2=LinearTriangleElementMassMatrix(2000, 0.001, 0.03, 0, 0.06, 0.03, 0.06, 0);
Ml=LinearTriangleAssemble(Ml, m2, 2, 5, 3);
k3=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.03, 0, 0.03, 0.03, 0.06, 0.03, 1);
Kl=LinearTriangleAssemble(Kl, k3, 2, 4, 5);
m3=LinearTriangleElementMassMatrix(2000, 0.001, 0.03, 0, 0.03, 0.03, 0.06, 0.03);
Ml=LinearTriangleAssemble(Ml, m3, 2, 4, 5);
k4=LinearTriangleElementStiffness(150000000000, 0.25, 0.001, 0.03, 0.03, 0.06, 0.06, 0.06, 0.03, 1);
Kl=LinearTriangleAssemble(Kl, k4, 4, 6, 5);
m34=LinearTriangleElementMassMatrix(2000, 0.001, 0.03, 0.03, 0.06, 0.06, 0.06, 0.03);
Ml=LinearTriangleAssemble(Ml, m4, 4, 6, 5);
```

```
%          generate stiffness matrix and mass matrix for the un-assembled system
PK=-[Ku, zeros(20,12); zeros(12,20), Kl];
PM=[Mu, zeros(20,12); zeros(12,20), Ml];

%          generate constraint matrix according to the geometric relationship between
%          the component nodes and assembly nodes
S=[eye(20),zeros(20,6); zeros(12,26)];
S(21,1)=1;
S(22,2)=1;
S(23,21)=1;
S(24,22)=1;
S(25,23)=1;
S(26,24)=1;
S(27,1)=-(1/16);
S(28,2)=-(1/16);
S(27,3)=(27/48);
S(28,4)=(27/48);
S(27,11)=(27/48);
S(28,12)=(27/48);
S(27,19)=-(1/16);
S(28,20)=-(1/16);
S(29,25)=1;
S(30,26)=1;
S(31,19)=1;
S(32,20)=1;

%          Assemble global stiffness matrix and global mass matrix using MMM
PKA=S'*PK*S ;
PMA=S'*PM*S;

%          Apply boundary condition for displacement (fixed points)
for i=1:10    %
    for j=1:10   %
        Mc(i,j)=PMA(i+2,j+2);
        Kc(i,j)=PKA(i+2,j+2);
    end
end
for i=1:10    %
    for j=11:23   %
        Mc(i,j)=PMA(i+2,j+3);
        Kc(i,j)=PKA(i+2,j+3);
```

```
            Mc(j,i)=PMA(j+3,i+2);
            Kc(j,i)=PKA(j+3,i+2);
        end
end
for i=11:23     %
        for j=11:23   %
            Mc(i,j)=PMA(i+3,j+3);
            Kc(i,j)=PKA(i+3,j+3);
        end
end

%           solve for natl freq (in rad/s)
[V,D]=eig(Kc, Mc);
V;
D;

%           display natl freq results in Hz
for j=1:1:23
      natl_freq(j)=sqrt(D(j,j))/(2*pi);
end
natl_freq=sort(natl_freq)
```

LinearTriangleElementMassMatrix.m function for generating lumped mass matrix for a 2-D triangle FEM element

```
function y = LinearTriangleElementMassMatrix(rou, t, xi,yi,xj,yj,xm,ym)

%LinearTriangleMassMatrix    This function returns the lumped mass matrix
%                            of a 2-D triangle FEM element whose first
%                            node is at coordinates (xi,yi), second
%                            node is at coordinates (xj,yj), and
%                            third node is at coordinates (xm,ym).
%                            the density of the plate is rou and thickness
%                            is t.
%    by Zhen Ren        Oct-31-2007

% calculate the area of the triangle
A = abs((xi*(yj-ym) + xj*(ym-yi) + xm*(yi-yj))/2);

% generate the lumped mass matrix for a triangle FEM element
y=(A*t*rou/3)*[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 1];
```

References

[1] Zienkiewicz, O. C., and Taylor, R. L., 1989, *The finite element method – 4th ed. Vol. 1: Basic formulation and linear problems*, McGraw-Hill, London

[2] Byam, B., Radcliffe, C., 1999, *Modular Modeling of Engineering System Using Fixed Input-Output Structure*, IMECE, Proceedings of The Symposium of Systematic Modeling, ASME, New York, November

[3] Li, X., 2000, MS thesis, *The Application of Modular Modeling Methods in Finite Element Analysis*, Department of Mechanical Engineering, Michigan State University

[4] Farooq, U., 2003, MS thesis, *Modular Finite Element Modeling by Distributed Internet Engineering Design Agents*, Department of Mechanical Engineering, Michigan State University

[5] Motato, E., Radcliffe, C., and Reichenbach, D., 2006, "Networked Assembly of Linear Physical System Models", *J. Dyn. Sys. Meas. & Control*, submitted

[6] Reddy, J. N., Gartling, D.K., 2004, *The Finite Element Method in Heat Transfer and Fluid Dynamics*, CRC Press

[7] Reichenbach, D., 2003, MS thesis, *Modeling of Dynamic Systems Using Internet Engineering Design Agents*, Department of Mechanical Engineering, Michigan State University

[8] Kattan, P. I., 2003, *MATLAB Guide to Finite Elements*, Springer-Verlag Berlin Heidelber