N94-23679

VECTORIZATION AND PARALLELIZATION OF THE FINITE STRIP METHOD FOR DYNAMIC MINDLIN PLATE PROBLEMS *

56-64
197138
p. 15

Hsin-Chu Chen
Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA

Ai-Fang He
Department of Mathematics
Illinois State University
Bloomington, IL

SUMMARY

The finite strip method is a semi-analytical finite element process which allows for a discrete analysis of certain types of physical problems by discretizing the domain of the problem into finite strips. This method decomposes a single large problem into m smaller independent subproblems when m harmonic functions are employed, thus yielding natural parallelism at a very high level. In this paper we address vectorization and parallelization strategies for the dynamic analysis of simply-supported Mindlin plate bending problems and show how to prevent potential conflicts in memory access during the assemblage process. The vector and parallel implementations of this method and the performance results of a test problem under scalar, vector, and vector-concurrent execution modes on the Alliant FX/80 are also presented.

INTRODUCTION

More and more parallel computers have been developed and made available to the engineering and scientific computing community in recent years. To take advantage of current and future advanced multiprocessors, however, a great deal of efforts remain to be made in the search for efficient and parallel implementations. In this paper we address both the coarse-grain and fine-grain parallelism offered by the finite strip method (FSM) for the dynamic analysis of Mindlin plate bending problems and present our vector and parallel implementations on multiprocessors with vector processing capabilities. FSM, first developed in the context of thin plate bending analysis, is a semi-analytical finite element process [6, 22]. This method allows for a discrete analysis of

*This work was supported by the U.S. Department of Energy under Grant No. DOE DE-FG02-85ER25001 while the authors were with the Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign.

PAGE 76 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

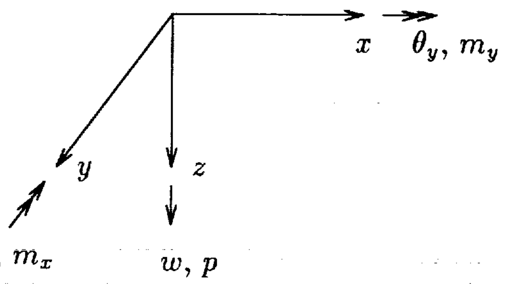

Figure 1: The coordinate system and sign convention.

certain types of physical problems by discretizing their domains into finite strips, involving an approximation of the true solution using a continuous harmonic series in one direction and piecewise interpolation polynomials in the others. Because of the orthogonality properties of the harmonic functions in the stiffness and mass matrix formulation, FSM decomposes a problem, when applicable, into many smaller and independent subproblems which yields coarse-grain parallelism in an extremely easy and natural way.

Although not as versatile as the finite element method, FSM has been applied to a wide range of plate, folded plate, shell, and bridge deck problems [4, 6, 7, 8, 10, 18] because of its efficiency and simplicity. The performance induced by the coarse-grain parallelism of this method in a multiprocessing environment has been shown in [9] for the static analysis of Mindlin plate problems and in [20] for groundwater modeling. In this paper, we report and compare the performance results of our implementation for the dynamic analysis of a simply-supported rectangular Mindlin plate using scalar, vector, and vector-concurrent execution modes on an Alliant FX/80.

THE PROBLEM

In this section we describe briefly the mathematical modeling of Mindlin plate problems [17]. Let Ω be the space domain in $\Re^2$, Γ the boundary, and T the time domain. Let also the stress resultants, generalized strains, displacements, dynamic surface loadings, and inertia forces be denoted respectively by **s**, **r**, **d**, **p**, and **q**:

$$\mathbf{s} = \begin{bmatrix} M_x \\ M_y \\ M_{xy} \\ Q_x \\ Q_y \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \gamma_x \\ \gamma_y \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} w \\ \theta_x \\ \theta_y \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p \\ m_x \\ m_y \end{bmatrix}, \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} -\rho h \ddot{w} \\ \frac{1}{12}\rho h^3 \ddot{\theta}_x \\ \frac{1}{12}\rho h^3 \ddot{\theta}_y \end{bmatrix}$$

where ρ stands for the mass density (per unit volume), h the thickness of the plate, and $\ddot{v}$ ($v = w$, θ_x, or θ_y) the second derivative of v with respect to time t: $\ddot{v} = \partial^2 v/\partial t^2$. The subscripts x, y, and z above represent the directions in the Cartesian coordinate system. The sign convention for the displacements and external loadings is shown in Figure 1. Neglecting the damping effect of the plate, the differential equations which govern the state of stress resultants, generalized strains, and displacements in an elastic plate can be expressed as

1. Equilibrium equations: $L_1^T\mathbf{s} + \mathbf{p} + \mathbf{q} = 0$ in $\Omega \otimes T$, subject to some appropriate boundary conditions on Γ,
2. Stress-strain equations: $\mathbf{s} = \mathbf{D}\mathbf{r}$, and
3. Strain-displacement equations: $\mathbf{r} = L_2\mathbf{d}$.

Here $\mathbf{D}$ is the material property matrix of an elastic plate. L_1 and L_2 are the differential operators:

$$L_1^T = \begin{bmatrix} 0 & 0 & 0 & \partial/\partial x & \partial/\partial y \\ \partial/\partial x & 0 & \partial/\partial y & -1 & 0 \\ 0 & \partial/\partial y & \partial/\partial x & 0 & -1 \end{bmatrix} \tag{1}$$

and

$$L_2^T = \begin{bmatrix} 0 & 0 & 0 & \partial/\partial x & \partial/\partial y \\ -\partial/\partial x & 0 & -\partial/\partial y & -1 & 0 \\ 0 & -\partial/\partial y & -\partial/\partial x & 0 & -1 \end{bmatrix} \tag{2}$$

where the superscript T denotes the transpose of a matrix.

For orthotropic material, the matrix $\mathbf{D}$ takes the form

$$\mathbf{D} = \begin{bmatrix} D_x & D_1 & & & \\ D_1 & D_y & & & \\ & & D_{xy} & & \\ & & & \alpha G_x & \\ & & & & \alpha G_y \end{bmatrix} \tag{3}$$

where D_x, D_1, ..., G_y are the standard flexural and shear rigidities of plates and α is a modification coefficient to account for the deviation of shear strain distribution from uniformity [4] ($\alpha = 5/6$ for rectangular cross section; see [21, p. 371]). The rest of the entries in $\mathbf{D}$ are zero. If the material is isotropic, then the nonzero entries take the following values:

$$D_x = D_y = \frac{Eh^3}{12(1-\nu^2)}, \quad D_1 = \nu D_x, \quad D_{xy} = \frac{1-\nu}{2}D_x, \quad \text{and} \quad G_x = G_y = \frac{Eh}{2(1+\nu)}$$

where E, h, and ν represent the material modulus, plate thickness, and Poisson's ratio, respectively. The total potential energy of the plate due to the dynamic surface loading $\mathbf{p}$ [17, 16, 14] can be written as

$$\Pi = \int_0^t \left(\frac{1}{2}\int_\Omega (L_2\mathbf{d})^T\mathbf{D}(L_2\mathbf{d})\, d\Omega - \int_\Omega \mathbf{p}^T\mathbf{d}\, d\Omega - \frac{1}{2}\int_\Omega \dot{\mathbf{d}}^T\mathbf{\Lambda}\dot{\mathbf{d}}\, d\Omega \right) dt \tag{4}$$

where $\dot{\mathbf{d}} = \partial\mathbf{d}/\partial t$ and $\mathbf{\Lambda} = \text{diag}\left[-\rho h,\ \frac{1}{12}\rho h^3,\ \frac{1}{12}\rho h^3\right]$, a diagonal matrix.

A STRIP ELEMENT FOR MINDLIN PLATES

We now outline the FSM formulation for the Mindlin plates using linear elements [4, 19]. We shall confine our discussions to rectangular Mindlin plate problems simply supported on two

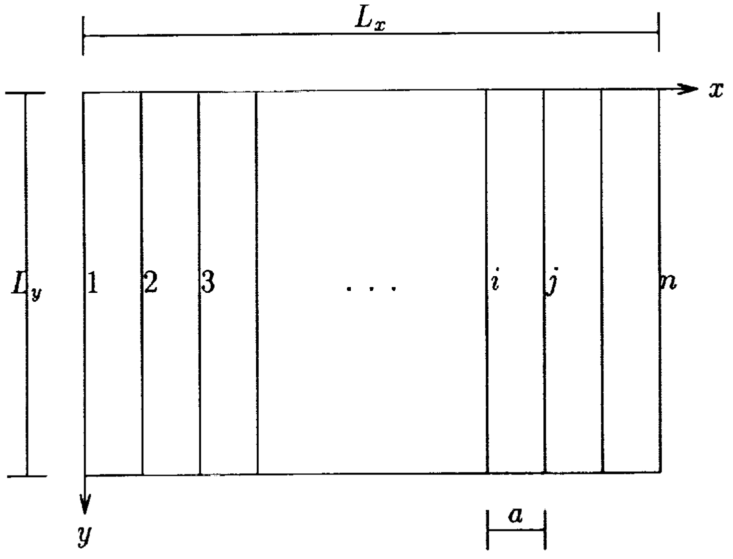

Figure 2: A discretized plate.

opposite sides. Figure 2 shows a rectangular plate discretized into $n-1$ finite strips. The plate is assumed to be simply supported on edges $y = 0$ and $y = L_y$. Shown in Figure 3 is the mid-plane of a typical linear strip plate element of constant thickness h, whose local coordinate system is denoted by $(x',\ y',\ z')$ where $x' = x - x_i$, $y' = y$, and $z' = z$. Let $\Omega_{(e)}$ be the domain of the e^{th} strip element and i and j be the two longitudinal edges (nodal lines) of the element, as shown in Figure 3. Let $\mathbf{d}_{(e)}(x,y,t)$ and $\mathbf{u}^l_{(e)}(t)$ be defined as

$$\mathbf{d}_{(e)}(x,y,t) = [w(x,y,t)\ \ \theta_x(x,y,t)\ \ \theta_y(x,y,t)]^T, \quad (x,y) \in \Omega_{(e)}$$

and

$$\mathbf{u}^l_{(e)}(t) = \begin{bmatrix} \mathbf{u}^l_i(t) \\ \mathbf{u}^l_j(t) \end{bmatrix} = \left[w^l_i(t)\ \ \theta^l_{xi}(t)\ \ \theta^l_{yi}(t) \mid w^l_j(t)\ \ \theta^l_{xj}(t)\ \ \theta^l_{yj}(t)\right]^T$$

where $w^l_i(t)$ denotes the l^{th} harmonic coefficient (amplitude) of $w_i(y,t)$ which is the displacement along edge i, etc. For a linear strip element with m harmonic terms specified, the approximation to $\mathbf{d}_{(e)}$ is given [4, 18] by

$$\mathbf{d}_{(e)}(x,y,t) \approx \sum_{l=1}^{m} \mathbf{F}^l(x,y)\mathbf{u}^l_{(e)}(t) \tag{5}$$

with

$$\mathbf{F}^l = \begin{bmatrix} N_iS_l & 0 & 0 & N_jS_l & 0 & 0 \\ 0 & N_iS_l & 0 & 0 & N_jS_l & 0 \\ 0 & 0 & N_iC_l & 0 & 0 & N_jC_l \end{bmatrix}$$

where S_l and C_l are the l^{th} harmonic functions of y, and N_i and N_j are the linear shape functions of x, defined by

$$S_l = sin\frac{l\pi y}{L_y}, \quad C_l = cos\frac{l\pi y}{L_y},$$

$$N_i = \frac{1-r_{(e)}}{2}, \text{ and } N_j = \frac{1+r_{(e)}}{2}$$

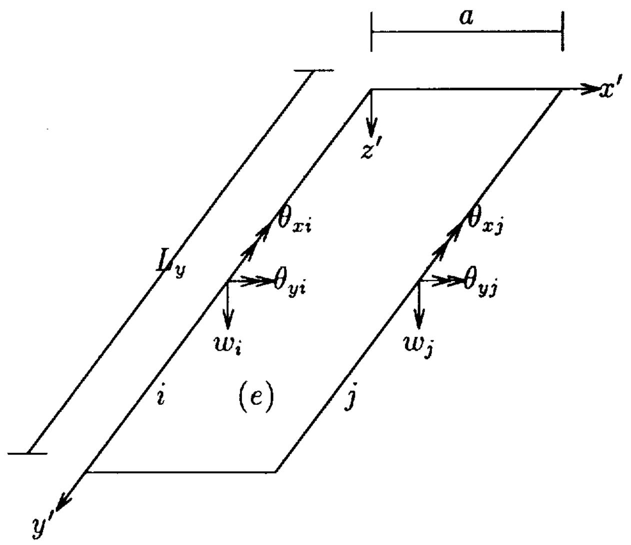

Figure 3: A typical plate strip element.

where $r_{(e)}$, ranging from -1 to 1, is the natural coordinate in x-direction of the e^{th} element. Note that $r_{(e)} = -1 + 2\frac{x-x_i}{x_j-x_i}$ for the element shown in Figure 3. It should be observed that the approximation to the displacement vector in (5) satisfies the simply supported boundary conditions on edges $y = 0$ and $y = L_y$; i.e., w, θ_x, $\partial w/\partial x$, $\partial\theta_x/\partial x$, and $\partial\theta_y/\partial y$ all vanish on these two edges. The dynamic surface loading on the e^{th} element, $\mathbf{p}_{(e)}(x,y,t)$, can often be approximated by the sum of a harmonic series in the longitudinal direction as shown below

$$\mathbf{p}_{(e)}(x,y,t) \approx \sum_{l=1}^{m} \mathbf{H}^l(y)\mathbf{p}^l_{(e)}(x,t) \tag{6}$$

where $\mathbf{H}^l = \text{diag}\,[S_l,\ \ S_l,\ \ C_l]$ and $\mathbf{p}^l_{(e)} = \begin{bmatrix} q^l & m^l_x & m^l_y \end{bmatrix}^T_{(e)}$. The subscript (e) outside the brackets indicates that every component of the vector is associated only with the e^{th} element.

Following the standard finite element procedure and taking advantage of the orthogonality properties of the harmonic functions, we obtain a linear algebraic differential system of block diagonal form [5] depicted by:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{7}$$

where

$$\mathbf{M} = \mathbf{M}^{11} \oplus \mathbf{M}^{22} \oplus \cdots \oplus \mathbf{M}^{mm} \quad \text{and} \quad \mathbf{K} = \mathbf{K}^{11} \oplus \mathbf{K}^{22} \oplus \cdots \oplus \mathbf{K}^{mm}$$

are block diagonal matrices of the same block structure. The vectors $\mathbf{u}$ and $\mathbf{f}$ are accordingly partitioned,

$$\mathbf{u}^T = \begin{bmatrix} (\mathbf{u}^1)^T & (\mathbf{u}^2)^T & \cdots & (\mathbf{u}^m)^T \end{bmatrix} \quad \text{and} \quad \mathbf{f}^T = \begin{bmatrix} (\mathbf{f}^1)^T & (\mathbf{f}^2)^T & \cdots & (\mathbf{f}^m)^T \end{bmatrix}.$$

In (7), the symbol $\oplus$ stands for the direct sum of square matrices. $\mathbf{M}^{ll}$, $\mathbf{K}^{ll}$, $\mathbf{u}^l$, and $\mathbf{f}^l$ are the system mass matrix, system stiffness matrix, system displacement amplitude vector, and system load amplitude vector due to the l^{th} harmonic mode, respectively. In the rest of the paper, we shall drop the term *amplitude* and simply call $\mathbf{u}^l$ ($\mathbf{f}^l$) the l^{th} system displacement (load) vector for brevity. $\mathbf{M}^{ll}$ is assembled from the strip mass matrix $\mathbf{M}^{ll}_{(e)}$, $\mathbf{K}^{ll}$ from the strip stiffness matrix $\mathbf{K}^{ll}_{(e)}$, and $\mathbf{f}^l$ from the strip load vector $\mathbf{f}^l_{(e)}$ where

$$\mathbf{M}^{ll}_{(e)} = \int_{\Omega_{(e)}} (\mathbf{F}^l)^T \mathbf{\Lambda} \mathbf{F}^l d\Omega_{(e)}, \quad l = 1,\ m, \tag{8}$$

$$\mathbf{K}^{ll}_{(e)} = \int_{\Omega_{(e)}} (L_2\mathbf{F}^l)^T \mathbf{D} (L_2\mathbf{F}^l) d\Omega_{(e)}, \quad l = 1,\ m, \tag{9}$$

$$\mathbf{f}^l_{(e)} = \int_{\Omega_{(e)}} (\mathbf{F}^l)^T \mathbf{H}^l \mathbf{p}^l_{(e)} d\Omega_{(e)}, \quad l = 1,\ m. \tag{10}$$

For a plate discretized with n nodal lines, $\mathbf{K}^{ll}$ and $\mathbf{M}^{ll}$ are square matrices of order $3n$ for each l. ($\mathbf{K}^{ll}_{(e)}$ and $\mathbf{M}^{ll}_{(e)}$ are of order 6.) Once the entire system stiffness matrix $\mathbf{K}$, system mass matrix $\mathbf{M}$, and system load vector $\mathbf{f}$ are assembled and the boundary conditions imposed, the remaining major work is to solve the linear algebraic differential system (7) for $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$.

PARALLEL AND VECTOR IMPLEMENTATIONS

Computational Procedure. Similar to the finite element method, FSM normally consists of the following three main computational components: (1) the generation of strip stiffness/mass matrices and strip load vectors for all strip elements, (2) the assemblage of the entire system stiffness/mass matrix and system load vector, and (3) the solution process of the resulting linear differential system $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}$. There are many step-by-step integration methods available for solving the 2nd-order linear differential equations. Among them are the central difference, Houbolt, Wilson θ, and Newmark β methods. The central difference method is an explicit scheme and the other three are implicit. Regardless of whether the method employed is implicit or explicit, the procedure basically involves an initial calculation of an effective coefficient matrix and then solves an effective linear system, after an effective load vector is formed, at each time step. In this paper, we employ the Newmark integration method whose procedure is shown below, where a_0, a_1, $\cdots$, a_7 are the Newmark integration constants [3, pp. 311]:

(1) initial calculation of the effective stiffness matrix $\hat{\mathbf{K}} = \mathbf{K} + a_0\mathbf{M}$, the factorization of $\hat{\mathbf{K}}$ into $\mathbf{LL^T}$ or $\mathbf{LDL}^T$ form, and then for each time step t_{k+1}, $k = 0, 1, \cdots$
(2) forming the effective load vector $\hat{\mathbf{f}}$ at time t_{k+1}: $\hat{\mathbf{f}}_{k+1} = \mathbf{f}_{k+1} + \mathbf{M}(a_0\mathbf{u}_k + a_2\dot{\mathbf{u}}_k + a_3\ddot{\mathbf{u}}_k)$,
(3) solving the effective linear system at time t_{k+1}: $\hat{\mathbf{K}}^{ll}\mathbf{u}_{k+1} = \hat{\mathbf{f}}_{k+1}$,
(4) calculating the acceleration and velocity vectors $\ddot{\mathbf{u}}_{k+1}$ and $\dot{\mathbf{u}}_{k+1}$:

$$\ddot{\mathbf{u}}_{k+1} = a_0(\mathbf{u}_{k+1} - \mathbf{u}_k) - a_2\dot{\mathbf{u}}_k - a_3\ddot{\mathbf{u}}_k, \quad \dot{\mathbf{u}}_{k+1} = \dot{\mathbf{u}}_k + a_6\ddot{\mathbf{u}}_k + a_7\ddot{\mathbf{u}}_{k+1}.$$

Note that the first step need be performed only once. The last three steps, however, must be performed at every time step and therefore constitute the most time-consuming part in the entire analysis.

To address the parallel implementation of FSM, we should first employ the decoupled structure of the system stiffness matrix depicted by (7), due to the orthogonality properties of harmonic functions. This decoupling leads to m independent sets of differential equations. Therefore, solving (7) is equivalent to solving

$$\mathbf{M}^{ll}\ddot{\mathbf{u}}^l + \mathbf{K}^{ll}\mathbf{u}^l = \mathbf{f}^l, \quad l = 1,\ m$$

where $\mathbf{K}^{ll}$ and $\mathbf{M}^{ll}$, $l = 1, \cdots, m$, are block tridiagonal matrices with each block of order only 3×3 for the ordering shown in Figure 2. Furthermore, each $\mathbf{M}^{ll}$ consists of only three nonzero diagonals. Since there is no data dependency among these m subsystems, not only can the generation of $\mathbf{M}^{ll}_{(e)}$, $\mathbf{K}^{ll}_{(e)}$, and $\mathbf{f}^l_{(e)}$ and the assemblage of $\mathbf{M}^{ll}$, $\mathbf{K}^{ll}$, and $\mathbf{f}^l$ for each harmonic term be performed independently, but all the subsystems can be solved in parallel. In a parallel computing environment with parallelism of two levels (considering vectorization as the first level), this special feature leads FSM to a fully parallelizable approach when the number of harmonic terms matches the number of processors. The following pseudo-Fortran code outlines its computational procedure and indicates where parallelism can be exploited for vector/concurrent executions.

```
C — Initial calculations
    DO 200 $l = 1,\ m$              (concurrent, one CPU per iteration)
        DO 100 $e = 1,\ N_s$                               (to be discussed)
            Generate $\mathbf{K}^{ll}_{(e)}$, $\mathbf{M}^{ll}_{(e)}$, and $\mathbf{f}^l_{(e)}$
            Assemble $\mathbf{K}^{ll}$, $\mathbf{M}^{ll}$, and $\mathbf{f}^l$
        END 100
        Initialize $\mathbf{u}^l$, $\dot{\mathbf{u}}^l$, and $\ddot{\mathbf{u}}^l$                          (vector)
        Form $\hat{\mathbf{K}}^{ll}$ from $\mathbf{K}^{ll}$ and $\mathbf{M}^{ll}$                       (vector)
        Factorize $\hat{\mathbf{K}}^{ll}$ into $\mathbf{LL^T}$ or $\mathbf{LDL^T}$ form          (vector)
    END 200
C — Calculations for each time step
    DO until the last time step                        (sequential)
        DO 400 $l = 1,\ m$          (concurrent, one CPU per iteration)
            DO 300 $e = 1,\ N_s$                           (to be discussed)
                Generate $\mathbf{f}^l_{(e)}$ and assemble $\mathbf{f}^l$
            END 300
            Form effective load vector $\hat{\mathbf{f}}^l$                (vector)
            Solve $\hat{\mathbf{K}}^{ll}\mathbf{u}^l = \hat{\mathbf{f}}^l$ for $\mathbf{u}^l$                   (vector)
            Calculate $\ddot{\mathbf{u}}^l$ and $\dot{\mathbf{u}}^l$                      (vector)
        END 400
        DO 600 $l = 1,\ m$                                 (sequential)
            Accumulate displacements $w$ for all strips  (vector-concurrent)
        END 600
    END DO
```

In the above pseudo-code, we neglect the step of imposing boundary conditions because they can be performed in the generation step. The word *concurrent* inside the parentheses after the DO statements is used to show that all iterations in this loop may be performed in parallel, on the basis of one processor per iteration ; and the word *vector* (or *vector-concurrent*) indicates computations involved in the statement should be performed in vector (or vector-concurrent) mode whenever possible and desirable. Whether a vector operation is desirable depends on the startup overhead and the vector length of the operation.

Data Structure and Parallelization. To allow current code restructurers to automatically vectorize or parallelize certain computations, the Fortran statements related to that part of computations are usually written in the form of DO loops or array constructs . Potential memory access conflict must also be resolved. Therefore, the data structure of the code plays an essential role. In our implementations, the system stiffness matrix **K** and system mass matrix **M** are represented by two 3D arrays SK(1:nbk,1:n,1:m) and SM(1:nbm,1:n,1:m), respectively, where nbk (nbm) is the semi-bandwidth of **K** (**M**), n the number of equations in each harmonic term, and m the number of harmonic terms. It should be noted that in many situations, it is more beneficial to interchange the first two dimensions of both **K** and **M**, or to concatenate the first two dimensions into a single dimension. The system load vector **f** is represented by a 2D array SF(1:n,1:m) and the vectors **u**, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ are similarly represented by 2D arrays SU, SV, and SA, respectively. This representation allows parallelization across harmonic terms to be performed in the outermost loop. It also makes the passing of references to subroutines an easy task.

To serve as an example, we consider the *DO 200* loop where the computations inside the loop are now translated into subroutines as shown below (the *DO 400* loop follows the same approach).

```
CVD$L CNCALL          ! an Alliant directive
      DO 200 L = 1, m        ! concurrent, one CPU per iteration
         CALL GenAss (SK(1,1,L), SM(1,1,L), SF(1,L), L, n, nbk, nbm, ns, ...)
         CALL Initialize (SU(1,L), SV(1,L), SA(1,L), ...)  ! Initialize $\mathbf{u}_0$, $\dot{\mathbf{u}}_0$, and $\ddot{\mathbf{u}}_0$.
         CALL Form (SK(1,1,L), SM(1,1,L), n, nbk, nbm, a0)  ! Form $\hat{\mathbf{K}}^{ll}$ and overwrite SK.
         CALL Factorize (SK(1,1,L), n, nbk)  ! Factorize $\hat{\mathbf{K}}^{ll}$ and overwrite SK.
      END 200
```

where GenAss is a subroutine performing the task of the *DO 100* loop in the previous pseudo code. The other three subroutines are self-explanatory. In the above code, the argument ns denotes the number of strips N_s and $a0$ is the Newmark constant a_0. Using this approach, each processor will have an identical local copy, automatically generated by the compiler, of the subroutines inside the loop and its own reference space (via the index L) in locating $\mathbf{K}^{ll}$, $\mathbf{M}^{ll}$, and $\mathbf{f}^{l}$; yielding concurrent execution for all harmonic terms because distinct processors will hold different values of L. This not only prevents memory access conflicts in performing these tasks but also enables us to use a single set of subroutines for all harmonic terms. The same applies to the other three subroutines as well. Note that the index L is also passed to the subroutine GenAss as a local variable because it is required for evaluating $\mathbf{K}^{ll}_{(e)}$, $\mathbf{M}^{ll}_{(e)}$, and $\mathbf{f}^{l}_{(e)}$ whose dimensions should be declared inside GenAss and will become local variables.

Vectorization. To address vectorization, we now turn to the computations for a single harmonic term. First we note that the formation of the effective stiffness matrix $\hat{\mathbf{K}}^{ll}$ and effective load vector $\hat{\mathbf{f}}^{l}$, and the calculation of $\ddot{\mathbf{u}}^{l}$ and $\dot{\mathbf{u}}^{l}$ consist mainly of matrix-matrix (vector-vector) additions and matrix-vector multiplications and are thus highly vectorizable. The vectorization and parallelization of factorizing $\hat{\mathbf{K}}^{ll}$ and solving the linear system $\hat{\mathbf{K}}^{ll}\mathbf{u}^{l} = \hat{\mathbf{f}}^{l}$ have been under intensive studies; see [13, 15, 23] for example. In this paper, we shall only focus on approaches to vectorizing the generation of $\mathbf{K}^{ll}_{(e)}$ and the assemblage of $\mathbf{K}^{ll}$. The generation of $\mathbf{M}^{ll}_{(e)}$ ($\mathbf{f}^{l}_{(e)}$) and the assemblage of $\mathbf{M}^{ll}$ ($\mathbf{f}^{l}$) follow the same way and, thus, need not be discussed.

There are two approaches to vectorizing the generation of $\mathbf{K}^{ll}_{(e)}$. The first, referred to as Vectorization within a Single Strip (VSS), is to generate the entries of $\mathbf{K}^{ll}_{(e)}$ in vector mode. This approach requires a minimal storage because $\mathbf{K}^{ll}_{(e)}$ for all strips can share the same storage of a single strip stiffness matrix, which is usually the case for most traditional finite strip or finite element programs. The disadvantage is that the vector length available for vectorization is limited by the order of the strip stiffness matrix, 6 in our case, which is rather small. In addition, the generation step may not even involve any loop structure because most of the Fortran statements may simply be assignment statements when the entries of $\mathbf{K}^{ll}_{(e)}$ are explicitly integrated. Therefore, we resort to the second approach: Vectorization across Multiple Strips (VMS). This approach generates the matrix entries component-wise across many different strips by employing the fact that each strip matrix can be generated independently of the others. It, however, requires a manual change in the data structure of the strip matrix in the computer program because current code restructurers can hardly accomplish this task automatically. One way of achieving our goal is to add one more dimension (preferably the first dimension) to the array that stores a strip matrix so that the new array can store all strip stiffness matrices. For example, let EKL(1:6,1:6) be the array used in the VSS approach for storing a single strip stiffness matrix and be shared by all strips, one at a time. (For simplicity, we ignore the symmetry of the matrix.) When the VMS approach is employed, we can simply change EKL to a 3D array, say EKL(1:ns,1:6,1:6), so that the first dimension is associated with strip identifications, allowing vector execution to be performed across strips. Although the change in data structure may impose some programming difficulty in modifying an existing code, this approach indeed provides a very good way for both vectorization and parallelization.

So far as the assemblage of the l^{th} system stiffness matrix $\mathbf{K}^{ll}$ is concerned, both VSS and VMS are still applicable if potential data dependencies are avoided. Note that assemblying an entry of $\mathbf{K}^{ll}_{(e)}$ to $\mathbf{K}^{ll}$ has no conflict with assemblying the other entries of the same matrix to $\mathbf{K}^{ll}$. Vectorization obviously can be performed within any single strip matrix without any difficulty, subject to the same disadvantage of short vector length as the case in the generation step. The following Fortran code indicates where vectorization can be performed using VSS for assemblying the stiffness matrix, where the rows of SKL store the upper diagonals of the band symmetric matrix $\mathbf{K}^{ll}$ using the Linpack format [12] with the main diagonal of $\mathbf{K}^{ll}$ stored in the last row of SKL.

```
DO 100 I = 1, NBK                    ! NBK (=6): Semi-bandwidth of K^ll
    SKL(I, 1:N) = 0.0    (vector)    ! Initialization. N: No. of equations of K^ll
END 100
DO 300 K = 1, NS                     ! NS: No. of strips
    K1 = 3 * (K-1)
    DO 200 J = 1, 6
        J1 = K1 + J
        I1 = NBK - J + 1
        SKL(I1:NBK, J1) = SKL(I1:NBK, J1) + EKL(1:J, J)    (vector)
                                     ! Vector length too short.
    END 200
END 300
```

Care, however, must be taken when the VMS approach is employed for assembling $\mathbf{K}^{ll}$. This is because different strips may have some nodes in common, which amounts to saying that the entries of $\mathbf{K}^{ll}_{(e)}$ from different strips may contribute themselves to the same location in $\mathbf{K}^{ll}$. Therefore, in order to vectorize the assemblage of $\mathbf{K}^{ll}$ from $\mathbf{K}^{ll}_{(e)}$ across multiple strip elements, we must find a way to avoid potential simultaneous updates of a common matrix entry. A general approach to avoid this situation is to use graph coloring techniques to partition strips so that all strips in the same group do not contain any common nodes. For our plate problems under consideration, two colors are enough: one for odd strips and the other for even strips. When a natural ordering is imposed as shown in Figure 2, however, a better approach to enhancing vectorization can be employed by assemblying entries component-wise (or node-wise) across all strip elements as shown below, assuming the i^{th} strip starts from nodal line i to nodal line $i+1$ and all strip stiffness matrices are available.

```
DO 100 I = 1, NBK                    ! NBK (=6): Semi-bandwidth of K^ll
    SKL(I, 1:N) = 0.0    (vector)    ! N: No. of equations of K^ll
END 100
DO 300 J = 1, 6
    JS = 3 * (NS-1) + J              ! NS: No. of strips
    DO 200 I = 1, J
        IJ = NBK - J + I
        SKL(IJ, J:JS:3) = SKL(IJ, J:JS:3) + EKL(1:NS, I, J)    (vector)
    END 200
END 300
```

Note that the array EKL now has one dimension more than the one used in the previous code. The storage can be reduced by about half if symmetry of the matrix is taken into account. Finally, we would like to mention that for a cluster-based multiprocessor with parallelism of three levels like the Cedar [11], FSM is a perfect candidate because the decoupling at the system level offers

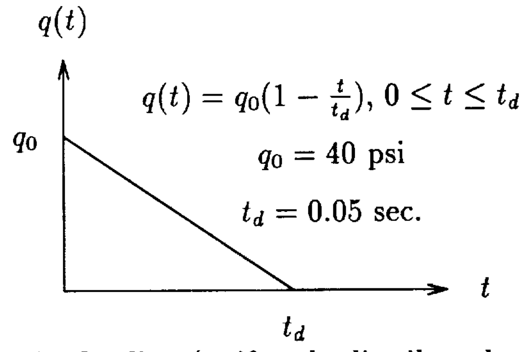

Figure 4: The triangular loading (uniformly distributed on the entire plate).

a great deal of freedom for the problem to be solved using all levels of parallelism. For example, we need exploit only the first two levels of parallelism in a linear system solver instead of three because the highest level of parallelism can be employed across multiple linear subsystems.

NUMERICAL EXPERIMENTS

To demonstrate the effectiveness and parallelizability of FSM, we consider the dynamic Mindlin analysis of a thin steel plate that is simply supported on all of its four edges and is subject to a uniformly distributed triangular loading $q(t)$ as shown in Figure 4. This plate, adapted from [2], is 60 inches (L_x) wide, 40 inches (L_y) long, and one inch thick throughout the entire plate. The material of the plate is assumed to be isotropic with Young's modulus $E = 30 \times 10^6$ ksi, Poisson ratio $\nu = 0.25$, and a mass density of $m = 0.00073$ lb-sec^2/in^4. The time step size Δt is set to 0.00001 sec. In evaluating the strip stiffness matrices, reduced integration with one Gaussian point is used to overcome the shear locking behavior [18]. The strip mass matrices are evaluated using the consistent mass approach. The linear algebraic differential equations are solved using the Newmark integration method with parameters $\alpha = 0.25$ and $\delta = 0.50$ [3, pp. 311]. A banded direct solver is used to solve the resulting linear subsystems in each time step.

In Figure 5, we compare the numerical solution of the displacement w at the center of the plate using 16 Mindlin strip elements with the exact solution (Fourier series) derived from the Kirchhoff thin plate theory. Eight harmonic terms are used in the finite strip approximation. From Figure 5, it is clear that the finite strip solution is in good agreement with the exact solution of the Kirchhoff theory. The performance of this method on an Alliant FX/80 is shown in Tables 2 and 3. In Table 2, we compare the CPU time (all in seconds) consumed in the entire analysis, including the generation, assemblage, and solution of the linear algebraic differential equations and finally the calculation of the displacements. Three different execution modes: scalar (S), vector (V), and vector-concurrent (VC) are considered. The compiler options [1] used for these modes are shown in Table 1.

Table 2 shows the vector speedup (the ratio of the 1-processor CPU time spent under the S mode to that under the V mode) for the entire process. As seen from this table, the vector

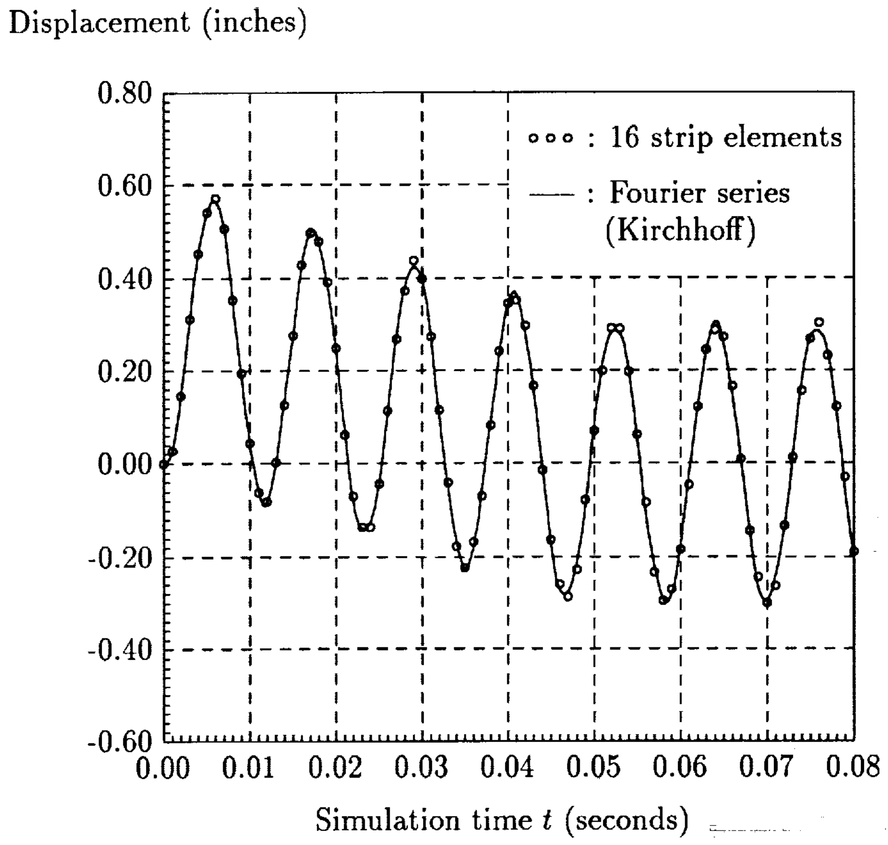

Figure 5: Displacement at the center of the plate.

Table 1: Compiler options

Execution mode	Compiler options	Subprograms compiled
Scalar (S)	-Og -AS -pg	the entire program
Vector (V)	-Ogv -AS -pg	the entire program
Vector-Concurrent (VC)	-Ogv -AS -Ogvc -AS	recursively-called subroutines others

Table 2: CPU time (in seconds) and vector speedup on the Alliant FX/80 using one processor.

Step	Scalar (S)	Vector (V)	S/V	Remark
Solve $\mathbf{LDL}^T\mathbf{u} = \hat{\mathbf{f}}$	177.1	137.1	1.29	semi-bandwidth too small
Compute $\hat{\mathbf{f}}$, $\dot{\mathbf{u}}$, $\ddot{\mathbf{u}}$ (Newmark)	91.0	25.3	3.60	mainly DAXPY operations.
Generate $\mathbf{f}^{l}_{(e)}$ and assemble $\mathbf{f}$	42.7	12.4	3.45	using the VMS approach
Initialization and I/O	1.72	1.70	1.01	no manual optimization
Total	312.4	176.4	1.77	

C-2.

Table 3: Parallel performance under the vector-concurrent mode.

No. of processors k	1	2	4	8
CPU time in seconds	165.7	84.14	45.01	25.08
Concurrency speedup S_k	1.00	1.97	3.68	6.61
Efficiency E_k (%)	100.0	98.5	92.0	82.6

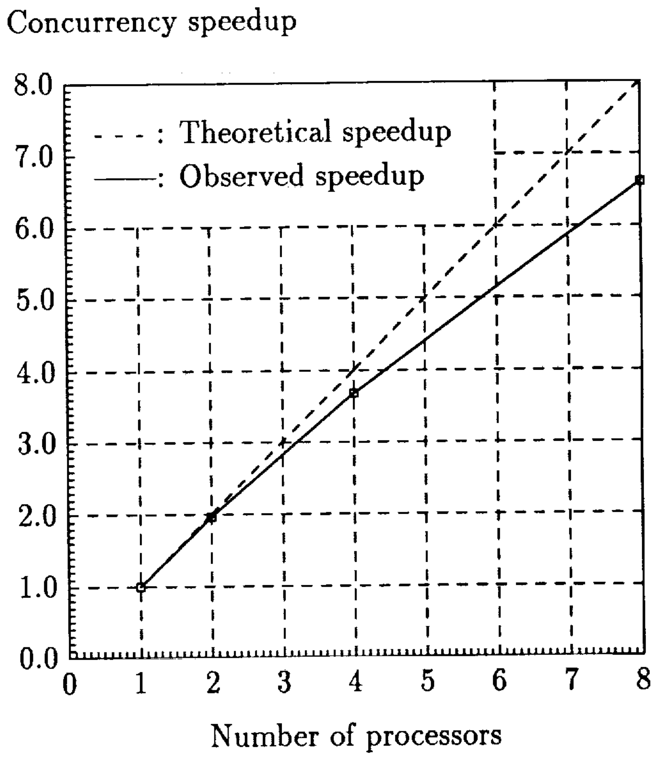

Figure 6: Concurrency speedup on the Alliant FX/80.

speedups for the three most time-consuming parts: (1) solving $\hat{K}u = \hat{\mathbf{f}}$, (2) computing $\hat{\mathbf{f}}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$, and (3) generating $\mathbf{f}^{l}_{(e)}$ and assemblying $\mathbf{f}$ are 1.29, 3.60, and 3.45, respectively. Note that the semi-bandwidth of the system stiffness matrix is only 6 in this example, which is obviously not long enough for a banded direct linear system solver to take advantage of vector instructions in solving the linear system. The vector speedups for the other two parts, however, are very satisfactory. It deserves mentioning that in generating $\mathbf{f}^{l}_{(e)}$ and assemblying $\mathbf{f}$, we employed the VMS approach which yields a much better vector performance than the VSS approach. Table 3 shows the concurrency speedup S_k, defined to be the ratio of the CPU time spent under the VC execution mode of the entire program using only one processor to that using k processors and the efficiency E_k $(= S_k/k)$, the ratio of the concurrency speedup S_k to the number of processors k. Figure 6 plots the speedup against the number of processors used. As seen from Table 3, the concurrency speedups observed using 2, 4, and 8 processors are 1.97, 3.68, and 6.61, respectively. This impressive performance clearly indicates the parallelizability of FSM on multiprocessors when the number of harmonic terms used matches the number of processors available.

CONCLUSIONS

The effectiveness and parallelizability of the finite strip method (FSM) for the dynamic analysis of a class of Mindlin plates have been addressed and vector/parallel implementations presented. The performance of this method on the Alliant FX/80 has also been tested using a rectangular plate that is simply supported on all edges and is subject to a uniformly distributed triangular loading. From the experiments performed, we have obtained concurrency speedups of 1.97, 3.68, and 6.61 using 2, 4, and 8 processors, respectively. These speedups are satisfactory and very encouraging. It clearly demonstrates the superiority of FSM in a parallel processing environment. For vectorization, good performance has also been observed for the Newmark integration scheme and for the generation/assemblage process using the VMS (vectorization across multiple strips) approach. In summary, we conclude that, although vector performance during the solution stage may be hindered by the small semi-bandwidth of the subsystems if a direct solver is employed, FSM is highly parallelizable and, therefore, suitable for computation on multiprocessor or multicluster computers. This is especially true when the problem requires a large number of harmonic terms to yield accurate results.

References

[1] Alliant Computer Systems Corporation, *FX/FORTRAN Programmer's Handbook*, Alliant Computer Systems Corporation, Acton, Massachusetts, 1987.

[2] A. Assadi-Lamouki and T. Krauthammer, An explicit finite difference approach for the Mindlin plate analysis, *Computers & Structures*, Vol.31, No.4 (1989), pp. 487-494.

[3] K. J. Bathe and E. L. Wilson, *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[4] P. R. Benson and E. Hinton, A thick finite strip solution for static, free vibration and stability problems, *Int. J. for Numer. Meth. in Eng.*, 10 (1976), pp. 665-678.

[5] J. M. Canet, B. Suárez, and E. Oñate, Dynamic analysis of structures using a Reissner-Mindlin finite strip formulation, *Computers & Structures*, Vol.31, No.6 (1989), pp. 967-975.

[6] Y-K. Cheung, The finite strip method in the analysis of elastic plates with two opposite simply supported ends, *Proc. Inst. Civ. Eng.*, 40(1968), pp. 1-7.

[7] Y-K. Cheung, Finite strip method analysis of elastic slabs, *ASCE J. of Mechanics Div.*, 94 (1968), pp. 1365-1378.

[8] Y-K. Cheung, *Finite Strip Method in Structural Analysis*, Pergamon Press, New York, 1976.

[9] H-C. Chen and A-F. He, Implementation of the finite strip method for structural analysis on a parallel computer, *Proc. 1990 Int'l. Conf. on Parallel Processing*, Vol. III: Algorithms and Applications (ed. P-C. Yew), August 1990, pp. 372-373.

[10] A. R. Cusens and Y. C. Loo, Applications of the finite strip method in the analysis of concrete box bridges, *Proc. Inst. Civ. Eng.*, 57-II (1974), pp. 251-273.

[11] E. Davidson, D. Kuck, D. Lawrie, and A. Sameh, Supercomputing tradeoffs and the Cedar system, *CSRD Tech. Rept. 577*, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, 1986.

[12] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK User's Guide*, SIAM, 1979.

[13] C. Farhat and E. Wilson, A parallel active column equation solver, *Computers & Structures*, Vol.28, No.2 (1988), pp. 289-304.

[14] D. G. Fertis, *Dynamics and Vibration of Structures*, John Wiley & Sons, New York, 1973.

[15] D. Goehlich, L. Komzsik, R. E. Fulton, Application of a parallel equation solver to static FEM problems, *Computers & Structures*, Vol.31, No.2 (1989), pp. 121-129.

[16] K. H. Huebner, *The Finite Element Method for Engineers*, John Wiley & Sons, New York, 1975.

[17] R. D. Mindlin, Influence of rotatory inertia and shear on flexural motions of isotropic, elastic plates, *J. of Applied Mechanics*, 18 (1951), pp. 31-38.

[18] E. Oñate and B. Suarez, A unified approach for the analysis of bridges, plates and axisymmetric shells using the linear Mindlin strip element, *Computers & Structures*, 17 (1983), pp. 407-426.

[19] E. Oñate and B. Suarez, A comparison of the linear quadratic and cubic Mindlin strip elements for the analysis of thick and thin plates, *Computers & Structures*, 17 (1983), pp. 427-439.

[20] J. A. Puckett and R. J. Schmidt, Finite strip method for groundwater modeling in a parallel computing environment, *Eng. Comput.*, 7 (1990), pp. 167-172.

[21] S. P. Timoshenko and J. M. Gere, *Mechanics of Materials*, Van Nostrand Co., New York, 1972.

[22] O. C. Zienkiewicz, *The Finite Element Method*, 3rd ed., McGraw-Hill, London, 1977.

[23] D. Zois, Parallel processing techniques for FE analysis: system solution, *Computers & Structures*, Vol.28, No.2 (1988), pp. 261-274.