Implementation Strategy for the Numerical Efficiency Improvement of the Multiscale Interpolation Wavelet-Galerkin Method

Jeong Hun Seo, Taemin Earmme
School of Mechanical and Aerospace Engineering and National Creative Research Initiatives Center for Multiscale Design, Seoul National University, Seoul 151-742, Korea
Gang-Won Jang
School of Mechanical Engineering, Kunsan National University, Kunsan, Chonbuk 573-701, Korea
Yoon Young Kim*
School of Mechanical and Aerospace Engineering and National Creative Research Initiatives Center for Multiscale Design, Seoul National University, Seoul 151-742, Korea

The multiscale wavelet-Galerkin method implemented in an adaptive manner has an advantage of obtaining accurate solutions with a substantially reduced number of interpolation points. The method is becoming popular, but its numerical efficiency still needs improvement. The objectives of this investigation are to present a new numerical scheme to improve the performance of the multiscale adaptive wavelet-Galerkin method and to give detailed implementation procedure. Specifically, the subdomain technique suitable for multiscale methods is developed and implemented. When the standard wavelet-Galerkin method is implemented without domain subdivision, the interaction between very long scale wavelets and very short scale wavelets leads to a poorly-sparse system matrix, which considerably worsens numerical efficiency for large-sized problems. The performance of the developed strategy is checked in terms of numerical costs such as the CPU time and memory size. Since the detailed implementation procedure including preprocessing and stiffness matrix construction is given, researchers having experiences in standard finite element implementation may be able to extend the multiscale method further or utilize some features of the multiscale method in their own applications.

Key Words : Multiscale, Wavelet-Galerkin Analysis, Subdomain Method

1. Introduction

The main advantage of the wavelet-based numerical method over the standard finite element method is that the wavelet method allows efficient adaptive analysis when the method is implemented in multiresolution. There have been several wavelet formulations to utilize the difference-checking nature of wavelets in developing multiresolution adaptive strategies (see Glowinski et al., 1994 ; Bertoluzza, 1997 ; Cohen and Masson, 1999 ; Cohen et al., 1998 ; Diaz, 1999 ; Dahmen, 2001) Although any wavelet may be used as the multiscale trial basis in Galerkin formulation, the hat interpolation wavelets employed by Christon and Roach (2000) and Jang et al.(2004a) have the simplest functional forms, and do not pose much difficulty in handling general boundary

* Corresponding Author,
E-mail : yykim@snu.ac.kr
TEL : +82-2-880-7154; **FAX** : +82-2-883-1513
School of Mechanical and Aerospace Engineering and National Creative Research Initiatives Center for Multiscale Design, Seoul National University, Seoul 151-742, Korea. (Manuscript **Received** July 15, 2005; **Revised** December 12, 2005)

conditions. Christon and Roach (2000) have shown that the hat interpolation wavelets are stable in H^1 and quite effective for problems with dominant elliptic characteristics. Jang et al. (2004a) have recently proposed a multiscale adaptive method that can handle general boundary conditions prescribed along curved boundaries. An interesting application of the multiscale adaptive analysis is made in shape optimization problems based on the fixed-grid method by Jang (2004) and Jang et al.(2004b). Since shape optimization involves the analysis of boundary-moving-problems, element-level multiscale methods such as the isoparametric-mapping-based method usually suffer from mesh distortion and remeshing difficulty. However, the fixed-grid-based multiscale method does not have such problems because the mesh is fixed throughout the optimization.

One drawback of the fixed-grid-based multiscale method is that it generally requires more computation time to construct the system matrix than the standard single-scale Galerkin method does. The wavelet integration rule by Latto et al. (1991) may not be used when the area of numerical integration involves more than one material as in the fixed-grid method. It is a common approach to construct a single-scale stiffness matrix first, and then to transform it into a multiscale version. However, the transformation requires a significant amount of CPU time and resources especially when the system is large-sized. Moreover, the sparsity of the resulting multiscale system matrix is considerably poor in comparison with the single-scale system matrix. The increased computation time and poor sparsity of the system matrix may overshadow the advantage of multiscale wavelets gained by the intrinsic adaptive characteristics. Thus, reducing the computational cost to construct and solve the multiscale system equation is critical to expand the wavelet-based multiscale method. Based on this motivation, we aim to develop an efficient implementation strategy for the fixed-grid-based multiscale wavelet-Galerkin method.

For the present multiscale formulation, the analysis domain is decomposed into several subdomains. Single-scale stiffness matrices are constructed and transformed into multiscale stiffness matrices separately for each subdomain. If the analysis domain is treated as a whole and no subdomain is introduced, the multiscale conversion process of the stiffness matrix requires tremendous amount of memory size as well as a substantially-increased computation time. The main role of the domain subdivision in the multiscale method is that the number of scales involved in each subdomain reduces as the number of subdomains increases. Therefore, the sparsity of the multiscale stiffness matrix can improve considerably. This will be explained clearly in the section discussing actual implementation procedure.

Since our main concern is the development of an efficient implementation strategy for the fixed-grid-based multiscale method, a detailed implementation procedure will be presented. There are some reports explaining the implementation procedure of the multiscale method, but (bi-)orthogonal wavelets, not the hat interpolation wavelets, are mainly considered. In this work, our wavelet-Galerkin implementation strategies will be presented for two-dimensional elasticity problems, and the work by Jang et al.(2004a) will be used as the starting point for our development.

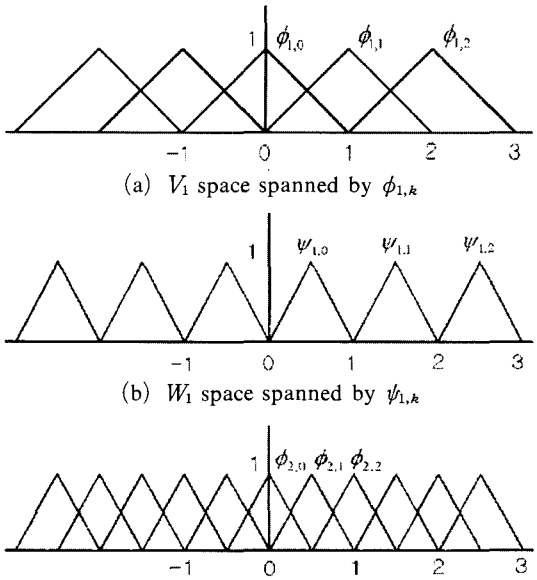
2. Interpolation Wavelet-Galerkin Formulation

In this section, the interpolation wavelet-Galerkin formulation without domain subdivision or element-wise multiscale transformation will be introduced. The formulation framework is based on the development by Jang et al.(2004a).

2.1 Hat interpolation wavelets for multiresolution analysis

The hat interpolation wavelets form the lowest-order basis which satisfies the multiresolution properties (see Mallat (1998) for the detailed discussion of the multiresolution analysis.) The one-dimensional hat interpolation scaling function $\phi(x)$ is expressed as:

$$\phi(x)=\sum_{k=-\infty}^{\infty} h_k \phi(2x-k) \tag{1}$$



(a) V_1 space spanned by $\phi_{1,k}$

(b) W_1 space spanned by $\psi_{1,k}$

(c) V_2 space spanned by $\phi_{2,k}$ ($V_2 = V_1 \oplus W_1$)

Fig. 1 The hat interpolation functions: (a) scaling functions $\phi_{1,k}$ at the resolution level $j=1$, (b) wavelets $\psi_{1,k}$ at $j=1$, and (c) scaling functions $\phi_{2,k}$ at $j=2$

where the coefficient h_k is given as $h_{-1}=0.5$, $h_0=1$, $h_1=0.5$, and $h_k=0$ for other k's. Note that $\phi(x)$ satisfies the following interpolation property

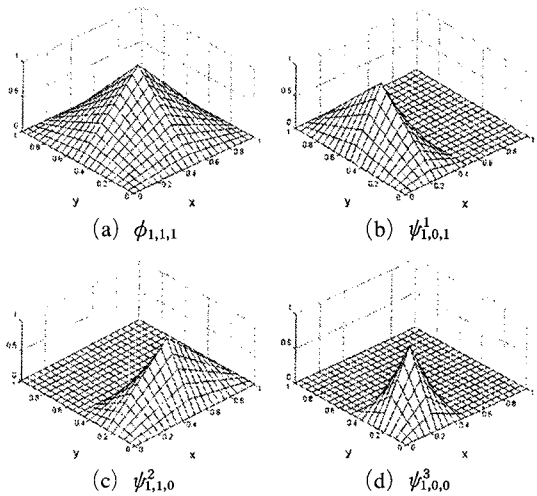
$$\phi(n)=\begin{cases}1 \text{ if } n=0\\ 0 \text{ else } (n\in\mathbb{Z})\end{cases} \tag{2}$$

The wavelet $\psi(x)$ is to be introduced to describe the difference between solutions at high and low resolution levels:

$$\psi(x)=\phi(2x-1) \tag{3}$$

$$\psi_{j,k}(x)=\psi(2^j x-k)=\phi_{j+1,2k+1}(x) \tag{4}$$

where the indices j and k denote the resolution level and the translation of the wavelet, respectively. Figs. 1(a) and (b) illustrate the one-dimensional hat interpolation scaling functions and wavelets at the resolution level $j=1$. Fig. 1(c) depicts the hat interpolation scaling functions at the resolution level $j=2$. Although the basis of $\phi_{2,k}$ and the basis of $\{\phi_{1,k}, \psi_{1,k}\}$ spans the same space, only the basis of $\{\phi_{1,k}, \psi_{1,k}\}$ is multiscaled.



(a) $\phi_{1,1,1}$ (b) $\psi^1_{1,0,1}$

(c) $\psi^2_{1,1,0}$ (d) $\psi^3_{1,0,0}$

Fig. 2 The two-dimensional hat interpolation functions on $\Omega=[0, 1]\times[0, 1]$

Two-dimensional hat interpolation wavelets can be constructed by the tensor product of the one-dimensional wavelets. The two-dimensional scaling function $\phi(x)$ and wavelets $\psi(x)$ are expressed as

$$\phi_{j,k,l}(x, y)=\phi(2^j x-k)\,\phi(2^j y-l) \tag{5}$$

$$\psi^1_{j,k,l}(x, y)=\psi_{j,k}(x)\,\phi_{j,l}(y) \tag{6a}$$

$$\psi^2_{j,k,l}(x, y)=\phi_{j,k}(x)\,\psi_{j,l}(y) \tag{6b}$$

$$\psi^3_{j,k,l}(x, y)=\psi_{j,k}(x)\,\psi_{j,l}(y) \tag{6c}$$

Fig. 2 illustrates the function shapes of some two-dimensional scaling function and wavelets defined on $[0, 1]\times[0, 1]$. Since ψ^1, ψ^2 and ψ^3 are the wavelets capturing the differences in the horizontal, vertical and diagonal directions, they are often called the horizontal, vertical and diagonal wavelets, respectively.

2.2 Multiscale wavelet-Galerkin formulation based on the fictitious domain approach

Fig. 3 describes a general two dimensional elastic problem defined on an elastic domain ω. A surface traction $\mathbf{t}$ is prescribed on the boundary Γ^h_ω while displacements are prescribed on Γ^g_ω. The weak formulation for the problem is written as

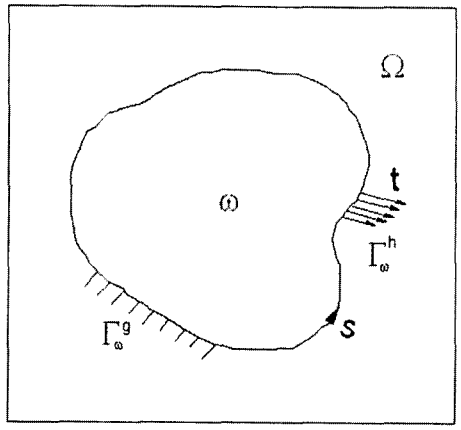


Fig. 3 The elasticity problem with a package domain ω embedded into a fictitious domain Ω

Fine $\mathbf{u}\in S_\omega$ for all $\mathbf{v}\in V_\omega$

$$\int_\omega \boldsymbol{\varepsilon}(\mathbf{v}) : \mathbf{C} : \boldsymbol{\varepsilon}(\mathbf{u})\, d\omega = \int_\omega \mathbf{f}\cdot\mathbf{v}\, d\omega + \int_{\Gamma_\omega^h} \bar{\mathbf{t}}\cdot\mathbf{v}\, d\Gamma_\omega \tag{7}$$

with

$$S_\omega=\{u_i\in H^1(\omega) \mid u_i=g_i \text{ on } \Gamma_\omega^g,\ i=1,\ 2\}$$
$$V_\omega=\{v_i\in H^1(\omega) \mid v_i=0 \text{ on } \Gamma_\omega^g,\ i=1,\ 2\}$$

where $\mathbf{C}$ denotes the elasticity tensor ε, the strain tensor and H^1, the Sobolev space of degree 1.

To be able to handle arbitrarily-shaped boundaries, we introduce a rectangular fictitious domain Ω, and embed ω inside Ω. In the domain Ω, the elasticity tensor $\mathbf{C}$ and the body force $\mathbf{f}$ are redefined as

$$\mathbf{C}_\Omega=\begin{cases}\mathbf{C} \text{ in } \omega\\ \gamma\mathbf{C} \text{ in } \Omega\backslash\omega\end{cases} \tag{8}$$

and

$$\mathbf{f}_\Omega=\begin{cases}\mathbf{f} \text{ in } \omega\\ 0 \text{ in } \Omega\backslash\omega\end{cases} \tag{9}$$

where γ is a small parameter.

Introducing the fictitious domain Ω, the weak formulation in Eq. (7) can be approximated as the problem of finding for $\mathbf{u}_\Omega\in S_\Omega$ all $\mathbf{v}_\Omega\in V_\Omega$:

$$\int_\Omega \varepsilon(\mathbf{v}_\Omega) : \mathbf{C}_\Omega : \boldsymbol{\varepsilon}(\mathbf{u}_\Omega)\, d\Omega = \int_\Omega \mathbf{f}_\Omega\cdot\mathbf{v}_\Omega d\Omega + \int_{\Gamma^h} \mathbf{t}\cdot\mathbf{v}_\Omega d\Gamma \tag{10}$$

with

$$S_\Omega=\{u_{\Omega i}\in H^1(\Omega) \mid u_{\Omega i}=g_i \text{ on } \Gamma_\omega^g,\ i=1,\ 2\}$$
$$V_\Omega=\{v_{\Omega i}\in H^1(\Omega) \mid v_{\Omega i}=0 \text{ on } \Gamma_\omega^g,\ i=1,\ 2\}$$

It is true that the solution accuracy and convergence are affected by the value of γ. Jang et al. (2004a) numerically showed that if γ is below 0.001, the solution difference between Eq. (10) and the original problem is negligible.

To obtain the multiscale discretized equation, the single-scale discretized equation may be first derived from (10) by using the hat interpolation scaling functions $\phi_{j,k,l}(x)$ as the basis functions at the resolution level J.[1] The discretized solution $\mathbf{u}_\Omega^J$ at the resolution level J can be expressed as

$$\mathbf{u}_\Omega^J=\sum_{k,l}\mathbf{s}_{j,k,l}\phi_{j,k,l}(x,\ y) \tag{11a}$$

or

$$\begin{Bmatrix}u_{\Omega,x}^J\\ u_{\Omega,y}^J\end{Bmatrix}=\sum_{k,l}\begin{Bmatrix}s_{x:J,k,l}\\ s_{y:J,k,l}\end{Bmatrix}\phi_{J,k,l}(x,\ y)$$

In the vector-matrix form, Eq. (11a) can be written as

$$\mathbf{u}_\Omega^J(x,\ y)=\mathbf{N}_J(x,\ y)\cdot\hat{\mathbf{U}}_J \tag{11b}$$

where $\mathbf{N}_J$ is the matrix consisting of scaling functions (or bilinear shape functions), and the vector $\hat{\mathbf{U}}_J$ is a solution vector consisting of $s_{x:J,k,l}$ and $s_{y:J,k,l}$. By using the same expression for $\mathbf{v}_\Omega^J$ as in Eqs. (11), and substituting both expressions into the weak form in (10), the following single-scale system equation can be obtained:

$$\hat{\mathbf{K}}_J\hat{\mathbf{U}}_J=\hat{\mathbf{F}}_J \tag{12}$$

with

$$\hat{\mathbf{K}}_J=\int_\Omega(\mathbf{N}_J)^T\mathbf{L}^T\mathbf{D}_\Omega\mathbf{L}\mathbf{N}_J d\Omega$$

where $\hat{\mathbf{K}}_J$ and $\hat{\mathbf{F}}_J$ denote the single-scale system stiffness matrix and the single-scale force vector, respectively. In the above, $\mathbf{D}_\Omega$ is the matrix form of $\mathbf{C}_\Omega$ in Eq. (8), and $\mathbf{L}$ denotes the usual differentiation operation.

1 For the resolution level J, the shorter edge of the rectangular domain Ω has 2^J+1 equally-spaced nodes.

To carry out the wavelet-Galerkin analysis in multiscale, we must express $\mathbf{u}_\Omega^J$ in the following multiscale form by using the wavelet basis functions ϕ and ψ:

$$\mathbf{u}_\Omega^J=\sum_{k,l}\mathbf{s}_{j_0,k,l}\phi_{j_0,k,l}(x,\ y)+\sum_{j=j_0}^{J-1}\sum_{m=1}^{3}\sum_{k,l}\mathbf{d}_{j,k,l}^m\psi_{j,k,l}^m(x,\ y) \tag{13}$$

or

$$\begin{Bmatrix} u_{\Omega,x}^J \\ u_{\Omega,y}^J \end{Bmatrix}=\sum_{k,l}\begin{Bmatrix} s_{x;j_0,k,l} \\ s_{y;j_0,k,l} \end{Bmatrix}\phi_{j_0,k,l}(x,\ y)+\sum_{j=j_0}^{J-1}\sum_{m=1}^{3}\sum_{k,l}\begin{Bmatrix} d_{x;j_0,k,l}^m \\ d_{y;j_0,k,l}^m \end{Bmatrix}\psi_{j,k,l}^m(x,\ y)$$

where the scale index j for wavelets ranges from j_0 to (j_0 is usually set to be 1.) When $j_0=1$, $J=2$ and $\Omega=[0,\ 1]\times[0,\ 1]$, there are 9 interpolation scaling functions $\phi_{1,k,l}(x,\ y)|_{(k,l)\in(0,1,2)}$, and 16 interpolation wavelets consisting of 6 horizontal wavelets ($\psi_{1,k,l}^1(x,\ y)$ with $k\in(0,\ 1)$, $l\in(0,\ 1,\ 2)$), 6 vertical wavelets ($\psi_{1,k,j}^2$ with $k\in(0,\ 1,\ 2)$, $l\in(0,\ 1)$), and 4 diagonal wavelets ($\psi_{1,k,l}^3(x,\ y)$ with $k,\ l\in(0,\ 1)$).

The direct approach to construct the multiscale system equation equivalent to the single-scale system equation in Eq. (12) requires somewhat involved analysis. The easiest way to derive the multiscale system equation is to transform the single-scale equation in Eq. (12) into the multiscale form (see Jang et al. (2004)):

$$\mathbf{K}_J\mathbf{U}_J=\mathbf{F}_J \tag{14}$$

through the following transforms

$$\hat{\mathbf{U}}_J=\mathbf{T}\mathbf{U}_J \tag{15}$$

$$\mathbf{F}_J=\mathbf{T}^T\hat{\mathbf{F}}_J \tag{16}$$

$$\mathbf{K}_J=\mathbf{T}^T\hat{\mathbf{K}}_J\mathbf{T} \tag{17}$$

In the above, $\mathbf{T}$ is the transformation matrix to convert the single-scale solution into the multiscale solution. The detailed process of constructing $\mathbf{T}$ will be discussed in the next section. The column array $\mathbf{U}_J$ is defined as

$$\mathbf{U}_J=\{\mathbf{s}_{j_0},\ \mathbf{d}_{j_0},\ \mathbf{d}_{j_0+1},\ \cdots,\ \mathbf{d}_{J-1}\}^T \tag{18}$$

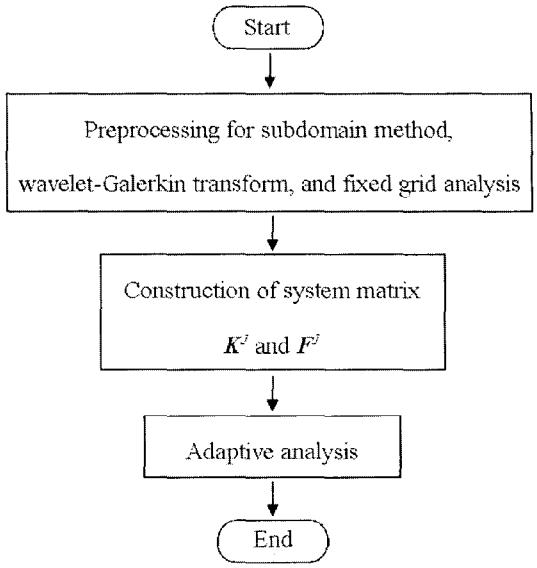


Fig. 4 The overall process of multiscale adaptive analysis

where

$$\mathbf{s}_{j_0}=\{\mathbf{s}_{j_0,k,l}\}^T,\ (k,\ l)\in\{0,\ 1,\ 2,\ \cdots,\ 2^{j_0}\}$$

$$\mathbf{d}_j=\{\mathbf{d}_j^1,\ \mathbf{d}_j^2,\ \mathbf{d}_j^3\}^T$$

with

$$\mathbf{d}_j^1=\{\mathbf{d}_{j,k,l}^1\}^T,\ k\in\{0,\ 1,\ \cdots,\ 2^j-1\},\ l\in\{0,\ 1,\ \cdots,\ 2^j\}$$

$$\mathbf{d}_j^2=\{\mathbf{d}_{j,k,l}^2\}^T,\ k\in\{0,\ 1,\ \cdots,\ 2^j\},\ l\in\{0,\ 1,\ \cdots,\ 2^j-1\}$$

$$\mathbf{d}_j^3=\{\mathbf{d}_{j,k,l}^3\}^T,\ k\in\{0,\ 1,\ \cdots,\ 2^j-1\},\ l\in\{0,\ 1,\ \cdots,\ 2^j-1\}$$

In the above definitions, the ranges of the translation indices k and l are differently given depending on the directions of the wavelets. The column array $\mathbf{F}_J$ can be similarly defined.

In the next two sections, we will explain the implementation procedure of the multiscale wavelet-Galerkin analysis in detail. The analysis procedure may be divided into three parts: preprocessing, construction of the system matrix, and the adaptive analysis as shown in Fig. 4.

3. Implementation : Preprocessing

The preprocessing procedure is illustrated in Fig. 5. Each process will be described below.

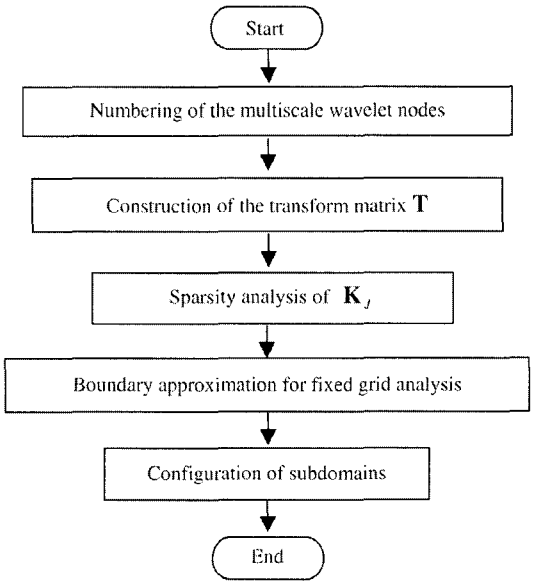


Fig. 5 The preprocessing flow chart for multiscale adaptive analysis

3.1 The numbering convention of the multiscale wavelet nodes

In the multiscale analysis, the single-scale coefficients (or solutions) $\hat{\mathbf{U}}_J$ are decomposed into multiscale coefficients $\mathbf{U}_J=\{\mathbf{s}_{j_0}, \mathbf{d}_{j_0}, \mathbf{d}_{j_0+1}, \cdots, \mathbf{d}_{J-1}\}^T$. As a numbering convention for the multiscale analysis, we number $\mathbf{s}_{j_0}$ first, and then $\mathbf{d}_{j_0}$ to $\mathbf{d}_{J-1}$. For example, let us consider the one-dimensional case. Fig. 6(a) illustrates the single-scale node numbering of $J=3$, and Fig. 6(b) depicts the corresponding multiscale node numbering. For the multiscale numbering, the nodes for the scaling functions are numbered first, and the nodes for the wavelets are numbered from the lowest scale to the highest scale.

For the two-dimensional node numbering, the directions of wavelets $\mathbf{d}_j^1$ are additionally considered. The coefficients of the horizontal wavelets $\mathbf{d}_j^2$ are numbered first, those of the vertical wave-

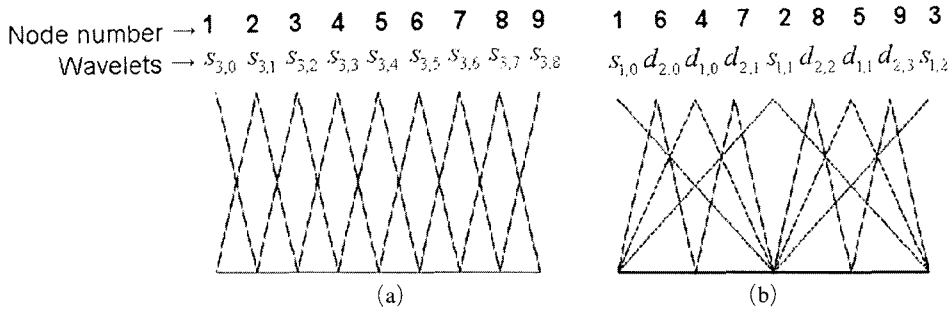


Fig. 6 One-dimensional node numbering: (a) the single-scale case, and (b) the multiscale case

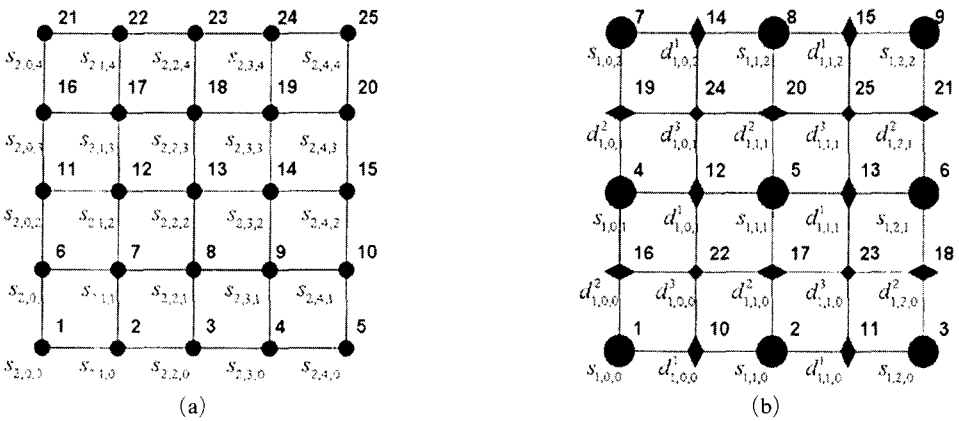


Fig. 7 Node numbering for a two-dimensional problem of $j=2$: (a) the single-scale case, and (b) the multiscale case (• : scaling functions of $s_{2,k,l}$, ● : scaling functions of $s_{1,k,l}$, ♦ : wavelets of $d^1_{1,k,l}$, ◆ : wavelets of $d^2_{1,k,l}$, and ◆ : wavelets of $d^3_{1,k,l}$)

lets are numbered next, and those of the diagonal wavelets $\mathbf{d}_j^3$ are numbered last, so the numbering sequence is

$$\mathbf{s}_{j_0} \rightarrow \mathbf{d}_{j_0}^1 \rightarrow \mathbf{d}_{j_0}^2 \rightarrow \mathbf{d}_{j_0}^3 \rightarrow \mathbf{d}_{j_0+1}^1 \rightarrow \mathbf{d}_{j_0+1}^2 \rightarrow \mathbf{d}_{j_0+1}^3 \rightarrow \cdots \rightarrow \mathbf{d}_{J-1}^1 \rightarrow \mathbf{d}_{J-1}^2 \rightarrow \mathbf{d}_{J-1}^3 \tag{19}$$

Fig. 7 compares the single-scale node numbering and the multiscale node numbering for a two-dimensional domain discretized by 5×5 fixed grids.

3.2 The construction of the transform matrix T

The role of the transform matrix $\mathbf{T}$ in Eq. (15) is to relate the single-scale solution $\hat{\mathbf{U}}_J$ with the multiscale solution $\mathbf{U}_J$, which is explicitly written as

$$\{\mathbf{s}_J\} = [\mathbf{T}] \begin{Bmatrix} \mathbf{s}_{j_0} \\ \mathbf{d}_{j_0} \\ \mathbf{d}_{j_0+1} \\ \vdots \\ \mathbf{d}_{J-1} \end{Bmatrix} \tag{20}$$

Since $\mathbf{T}$ involves several scales, $\mathbf{T}$ may be constructed by iterative multiplications of the one-level transform matrix $\mathbf{T}_j$ relating $\{\mathbf{s}_j\}$ and $\{\mathbf{s}_{j-1}, \mathbf{d}_{j-1}\}$ for $j=j_0, \cdots, J$ so that the process to construct $\mathbf{T}_j$ should be explained first.

To obtain the one-level transform matrix $\mathbf{T}_j$, the domain with 5×5 fixed grids in Fig. 7 is considered. For ease of explanation, we assume that each node has only one degree of freedom. First, twenty five single-scale grid points in Fig. 7(a) are divided into four groups depending on their locations:

Group 1 : single-scale nodes for multiscale s_{j-1} (1, 3, 5, 11, 13, 15, 21, 23, 25),
Group 2 : single-scale nodes for multiscale d_{j-1}^1 (2, 4, 12, 14, 22, 24),
Group 3 : single-scale nodes for multiscale d_{j-1}^2 (6, 8, 10, 16, 18, 20),
Group 4 : single-scale nodes for multiscale d_{j-1}^3 (7, 9, 17, 19).

As a typical node belonging to Group 1, consider the single-scale node 13 on which $s_{2,2,2}$ is defined. As shown in Fig. 2(a), $\phi_{1,1,1}$ on the multiscale node 5 is 1 at the multiscale node 5 and vanishes at the multiscale nodes 1, 2, 3, 4, 6, 7, 8, 9. Also $\phi_{2,2,2}$ corresponding to the single-scale node 13 is 1 at the single-scale node 13. Therefore, the relation between the multiscale coefficient $s_{1,1,1}$ and the single-scale coefficient $s_{2,2,2}$ is simply

$$1 \times s_{1,1,1} = s_{2,2,2} \tag{21}$$

Other single-scale coefficients of the nodes in Group 1 are related to the multiscale coefficients in the same way as in Eq. (21).

Next, nodes in Group 2 are considered. The single-scale node 14 on which $s_{2,3,2}$ is defined is examined. To determine multiscale wavelets not vanishing at the single-scale node 14, one may examine the plots in Fig. 2. By noting that only $\phi_{1,1,1}$, $\phi_{1,2,1}$ and $\psi_{1,1,1}^1$ do not vanish at the single-scale node 14, and taking into account the values of $\phi_{1,1,1}$, $\phi_{1,2,1}$ and $\psi_{1,1,1}^1$ at node 14, one can derive the following relation:

$$d_{1,1,1}^1 + 0.5 s_{1,1,1} + 0.5 s_{1,2,1} = s_{2,3,2} \tag{22}$$

Similarly, one can derive the relations for nodes belonging to Group 3 and Group 4. For instance, the single-scale coefficient $s_{2,2,3}$ (in Group 3) defined on the single-scale node 18 is related to the multiscale coefficients $d_{1,1,1}^2$, $s_{1,1,1}$, $s_{1,1,2}$ as

$$d_{1,1,1}^2 + 0.5 s_{1,1,1} + 0.5 s_{1,1,2} = s_{2,2,3} \tag{23}$$

and the relation of the single-scale coefficient $s_{2,1,3}$ (in Group 4) with the corresponding multiscale coefficients is given as

$$d_{1,0,1}^3 + 0.5 d_{1,0,1}^2 + 0.5 d_{1,1,1}^2 + 0.5 d_{1,0,2}^1 + 0.5 d_{1,0,1}^1 + 0.25 s_{1,0,1} + 0.25 s_{1,1,1} + 0.25 s_{1,0,2} + 0.25 s_{1,1,2} = s_{2,1,3} \tag{24}$$

Using the relations in Eqs. (21) ~ (24) and extending them for other nodes, the one-level transform matrix $\mathbf{T}_2$ of size 25×25 for 5×5 fixed grids can be constructed. Once the one-level transform matrix is known, the matrix $\mathbf{T}$ in Eq. (15) is constructed by the successive multiplication of $\mathbf{T}_j$ as depicted in Fig. 8.

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{J-1} \end{bmatrix} \begin{bmatrix} \mathbf{T}_{J-2} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{T}_{j_0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

Fig. 8 Construction of the total transform matrix $\mathbf{T}$

3.3 Sparsity pattern of the multiscale stiffness matrix $\mathbf{K}_J$ and its storage

The single-scale system stiffness matrix $\hat{\mathbf{K}}_J$ is a sparse matrix; only a few nonzero components need to be stored in the computer memory. So, before constructing $\hat{\mathbf{K}}_J$, sparsity analysis to find the locations of nonzero components should be performed. The fixed grids on a rectangular fictitious domain have a simple sparsity pattern for the single-scale stiffness matrix $\hat{\mathbf{K}}_J$. However, the sparsity of the multiscale system stiffness matrix $\mathbf{K}_J$ is somewhat complicated since the supports of the multiscale wavelets are intricately overlapped to each other. Subsequently, the sparsity analysis of $\mathbf{K}_J$ is not a trivial matter and should be performed by considering the overlapped supports of the multiscale basis functions. Fig. 9 illustrates an example of the sparsity pattern for $\mathbf{K}_J$. Nonzero components of $\mathbf{K}_J$ are marked black in the figure.

In this work, we use the ELL format for the storage of the sparse matrix $\mathbf{K}_J$ (BLAST Forum, 1997). A sparse matrix can be stored in the ELL format using three arrays:

- $INDEX$: a two-dimensional integer array of the size $[NROW \times MAXNZ]$ to store the column indices of the nonzero components of each row of a sparse matrix.

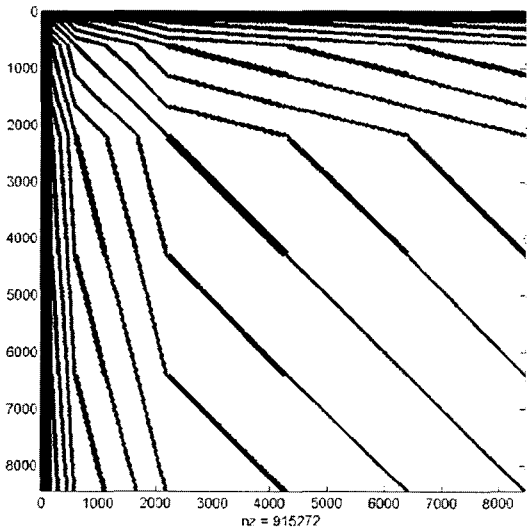


Fig. 9 The sparsity structure of the multiscale stiffness matrix $\mathbf{K}_J$

- $DATA$: a two-dimensional array of the size $[NROW \times MAXNZ]$ to store the values of the nonzero components of each row of a sparse matrix.
- NNZ: a one-dimensional integer array of the size $[NROW]$ to store the number of the nonzero components of each row of a sparse matrix.

In the above, $NROW$ denotes the number of rows of the matrix, and $MAXNZ$ is the maximum value among the components of NNZ.

As an example of the ELL format, the following matrix $\mathbf{A}$ is considered

$$\mathbf{A}=\begin{bmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{bmatrix} \tag{25}$$

The ELL representation of $\mathbf{A}$ is given as

$$NNZ=\begin{bmatrix} 2 \\ 3 \\ 3 \\ 4 \\ 4 \\ 3 \end{bmatrix},\ INDEX=\begin{bmatrix} 1 & 5 & & \\ 1 & 2 & 6 & \\ 2 & 3 & 4 & \\ 1 & 3 & 4 & 5 \\ 2 & 4 & 5 & 6 \\ 2 & 5 & 6 & \end{bmatrix},$$

$$DATA=\begin{bmatrix} 10 & -2 & & \\ 3 & 9 & 3 & \\ 7 & 8 & 7 & \\ 3 & 8 & 7 & 5 \\ 8 & 9 & 9 & 13 \\ 4 & 2 & -1 & \end{bmatrix} \tag{26}$$

The sparsity analysis is to find column indices of the nonzero components in each row of a sparse matrix. In other words, the sparsity analysis is to obtain the array NNZ and the array $INDEX$ of the multiscale system matrix $\mathbf{K}_J$.

3.4 Boundary element information

For the boundary approximation of the fixed grid method, the approach developed in Jang (2004) and Jang et al. (2004b) is used. Boundary curves are approximated by piecewise oblique

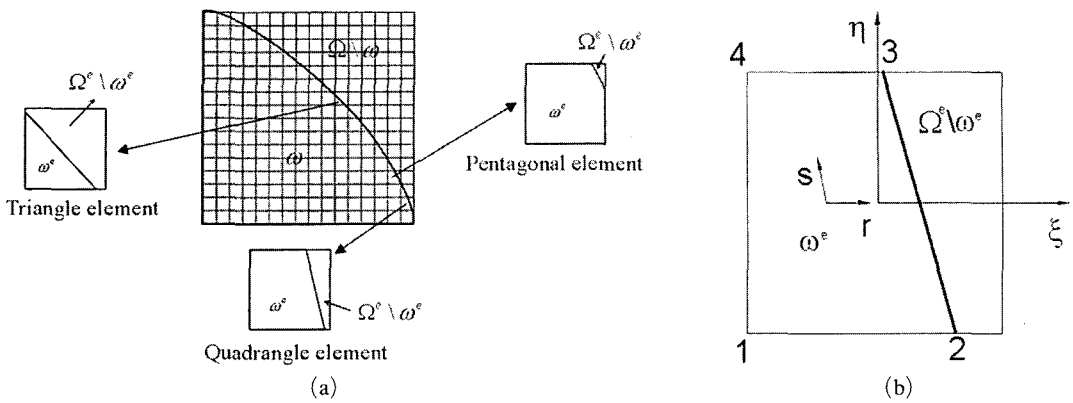


Fig. 10 The boundary approximation for the fixed grid analysis: (a) three types of boundary elements, and (b) the quadrilateral boundary element

lines connecting intersection points between the original boundaries of the domain and the fixed mesh. In this approach, boundary elements have two material regions: the real material for the original domain ω and the fictitious material for the fictitious domain $\Omega\backslash\omega$. Also, there exist three kinds of boundary elements depending on the shape of the area lying inside ω: triangular elements, quadrilateral elements and pentagonal elements, as illustrated in Fig. 10(a). In Fig. 10 (a), Ω^e denotes the region of an element, and ω^e denotes the region of the element lying inside ω.

Next, the evaluation of the stiffness matrices of the boundary elements is considered. Since the multiscale stiffness matrices can be obtained through the multiscale transforms of the single-scale stiffness matrices as in Eq. (17), only the single-scale element stiffness calculation is investigated. Moreover, a triangular boundary element can be considered as a degenerated case of a quadrilateral boundary element, and the stiffness of a pentagonal boundary element is obtained simply by subtracting the stiffness of $\Omega^e\backslash\omega^e$ from the stiffness of an element concerning Ω^e. So, only the stiffness evaluation of a quadrilateral element is presented in this work.

The stiffness matrix of a quadrilateral element can be written as

$$\hat{\mathbf{K}}_j^e=\int_{-1}^{1}\int_{-1}^{1}\mathbf{B}^T(\xi,\eta)\mathbf{D}_\Omega(\xi,\eta)\mathbf{B}(\xi,\eta)|\mathbf{J}|d\xi d\eta \quad (27)$$

where (ξ, η) are the element local coordinates, $\mathbf{B}$ denotes the matrix relating strains with displacements, and $\mathbf{D}_\Omega$ is the matrix form of $\mathbf{C}_\Omega$ in Eq. (8). In (27), the Jacobian $|\mathbf{J}|$ is simply one quarter of the element area because the element shape is a rectangular. Note that $\mathbf{D}_\Omega$ is not constant over a boundary element because the element has two material regions.

To facilitate the stiffness evaluation of a boundary element, we introduce additional coordinates (r, s) mapping a normalized rectangular domain $[-1, 1]\times[-1, 1]$ to , the region bounded by 1-2-3-4 in Fig. 10(b). As in the standard bilinear finite element, the element local coordinates (ξ, η) can be expressed as

$$\begin{pmatrix}\xi\\ \eta\end{pmatrix}=\sum_{i=1}^{4}N_i(r,s)\begin{pmatrix}\xi_i\\ \eta_i\end{pmatrix} \quad (28)$$

where (ξ_i, η_i) are the coordinates of the intersection points, e.g., points 1, 2, 3 and 4 in Fig. 10(b), $N_i(r, s)$ and represent standard bilinear functions. Using Eqs. (27) and (28), one may write the single-scale stiffness matrix of a boundary element as

$$\begin{aligned}\hat{\mathbf{K}}_j^e&=\gamma\int_{\Omega^e/\omega^e\Gamma}\mathbf{B}^T(\xi,\eta)\mathbf{D}\mathbf{B}(\xi,\eta)|\mathbf{J}|d\xi d\eta\\&\quad+(1-\gamma)\int_{\omega^e}\mathbf{B}^T(\xi,\eta)\mathbf{D}\mathbf{B}(\xi,\eta)|\mathbf{J}|d\xi d\eta\\&\cong\int_{-1}^{1}\int_{-1}^{1}\mathbf{B}^T(\xi(r,s),\eta(r,s))\\&\quad\mathbf{D}\mathbf{B}(\xi(r,s),\eta(r,s))|\mathbf{J}||\tilde{\mathbf{J}}|drds\end{aligned} \quad (29)$$

where $|\tilde{\mathbf{J}}|$ is the Jacobian relating (r, s) and

(ξ, η), and γ is a small positive parameter discussed in Eq. (8).

Since a boundary element requires special integration rule in Eq. (29), an identification number ID is allotted to each element during the preprocessing to tell the type of an element:

$ID=0$ for elements lying outside ω,
$ID=1$ for elements lying inside ω,
$ID=2$ for triangular boundary elements,
$ID=3$ for quadrilateral boundary elements,
$ID=4$ for pentagonal boundary elements.

In case of $ID=2$, 3, and 4, the coordinates of the intersection points should be also recorded to compute $|\tilde{\mathbf{J}}|$ in Eq. (29). Therefore nine data are stored for each element in the element table to calculate the stiffness matrix of the element: ID, ξ_1, ξ_2, ξ_3, ξ_4, η_1, η_2, η_3, and η_4.

4. Implementation : Multiscale System Matrix Construction and Adaptive Analysis

For the wavelet-Galerkin analysis, the multiscale system matrix $\mathbf{K}_J$ can be easily constructed from the single-scale system matrix $\hat{\mathbf{K}}_J$ by using Eq. (17). However, when the size of $\mathbf{K}_J$ is very large, it generally requires too much time and very large memory. For instance, if $2^{10}\times 2^{10}$ plane stress elements are used for the analysis, the total degrees of freedom for $\hat{\mathbf{K}}_J$ will be $(2^{10}+1)\times(2^{10})\times 2=2,101,250$. To convert this large-sized stiffness matrix $\hat{\mathbf{K}}_J$ to its multiscale version $\mathbf{K}_J$ requires a transform matrix $\mathbf{T}$ having the same size as $\hat{\mathbf{K}}_J$. Therefore, a tremendous amount of computation time and memory size cannot be avoided unless the actual matrix size for the transformation is reduced.

In order to overcome this difficulty, we propose to divide the analysis domain into several subdomains. Then the multiscale transform of the system matrix is performed separately in each subdomain, and the transformed multiscale subdomain system matrices are finally assembled. As the number of subdomains (will be denoted as ND) increases, not only the size of the stiffness matrix but also the size of the transform matrix reduces. Therefore, the system matrix construction of $\mathbf{T}^T\mathbf{K}_J\mathbf{T}$ with many subdomains (i.e., large ND) is computationally much more efficient than the case of no subdomain ($ND=1$). However, the use of too many subdomains deteriorates the efficiency of the wavelet-based adaptive analysis (the adaptive strategy will be explained later.) The effect of the number of subdomains on the CPU time and the required memory size will be studied numerically with benchmark problems.

In forming the multiscale system matrix $\mathbf{K}_J$, SPARSKIT (Saad, 1994) was used for the multiplication of matrices stored in the sparse format. The following is the multiplication process for $\mathbf{K}_J$ using SPARSKIT where the CSR (Compressed Sparse Row) format is used for the sparse matrix operations:

$$\textcircled{1}\ \mathbf{T}^T \leftarrow \mathbf{T},\ \textcircled{2}\ \mathbf{K}_{\_temp} \leftarrow \mathbf{T}^T\hat{\mathbf{K}}_{j,sub},$$
$$\textcircled{3}\ \mathbf{K}_{j,sub} \leftarrow \mathbf{K}_{\_temp}\mathbf{T},\ \textcircled{4}\ \mathbf{K}_J \leftarrow \sum_{sub=1}^{ND} \mathbf{K}_{J,sub}$$

where $\mathbf{K}_{j,sub}$ is the multiscale subdomain system matrix. In the above, the fourth step is the assemble process of the subdomain stiffness matrices into the global system matrix. It is difficult or inefficient to impose Dirichlet boundary conditions on the multiscale system matrix $\mathbf{K}_J$. Therefore, the boundary conditions are imposed on the single-scale system matrix before the transformation.

The procedure to transform the single-scale force vector $\hat{\mathbf{F}}_J$ to the multiscale force vector $\mathbf{F}_J$ is similarly conducted. For completeness, we write the implementation process:

$$\textcircled{1}\ \mathbf{T}^T \leftarrow \mathbf{T},\ \textcircled{2}\ \mathbf{F}_{J,sub} \leftarrow \mathbf{T}^T\hat{\mathbf{F}}_{J,sub},$$
$$\textcircled{3}\ \mathbf{F}_J \leftarrow \sum_{sub=1}^{ND} \mathbf{F}_{J,sub}$$

where all matrices and vectors are stored in sparse form, and Sparse BLAS (Carney et al, 1996) was used for the matrix-vector multiplications.

The main advantage of using multiscale wavelets for numerical analysis is that simple and effective adaptive analysis can be performed by utilizing the difference-checking nature of wavelets. Since the wavelet coefficients represent the differences between the solutions at different resolutions, they can be used as good error es-

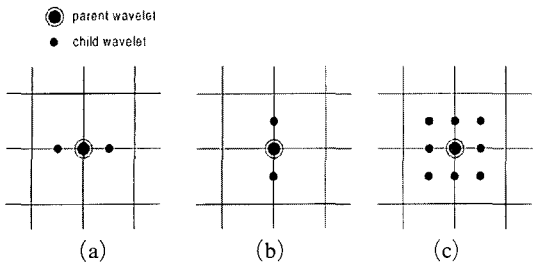


Fig. 11 The insert positions of child wavelets according to the parent wavelet's type: (a) horizontal parent wavelet, (b) vertical parent wavelet, and (c) diagonal parent wavelet

timators for adaptive analysis. In this work, we employ the same adaptive scheme as in the previous works (Jang et al., 2004a ; Jang, 2004). The two thresholding parameters δ_j^{up} and δ_j^{low} ($\delta_j^{up} > \delta_j^{low} > 0$) will be used to control the degree of the adaptivity. The adaptive scheme is given as

- Exclude wavelets $\psi_{j,k,l}^m$ from the basis set if $|d_{j,k,l}^m| < \delta_j^{low}$,
- Preserve $\psi_{j,k,l}^m$ in the basis set if $\delta_j^{low} < |d_{j,k,l}^m| < \delta_j^{up}$ (30)
- Add child wavelets of $\psi_{j,k,l}^m$ into the basis set if $|d_{j,k,l}^m| > \delta_j^{up}$.

In (30), the child wavelets of the parent wavelet $\psi_{j,k,l}^m$ are the wavelets emerging around the support center of $\psi_{j,k,l}^m$ to decrease the solution error around $\psi_{j,k,l}^m$. Depending on the parent wavelet's type, the child wavelets are inserted in different locations as illustrated in Fig. 11.

The magnitude of the thresholding parameters should decrease as the resolution level increases. The following reduction strategies are commonly used. For horizontal and vertical wavelets,

$$\delta_{j+1}^{up} = \delta_j^{up}/2, \quad \delta_{j+1}^{low} = \delta_j^{low}/2 \tag{31}$$

and for diagonal wavelets,

$$\delta_{j+1}^{up} = \delta_j^{up}/4, \quad \delta_{j+1}^{low} = \delta_j^{low}/4 \tag{32}$$

5. Numerical Study

In this section, we mainly investigate the effect of the domain subdivision on the computational CPU time and memory usage. The adaptive strategy described in the previous section will be

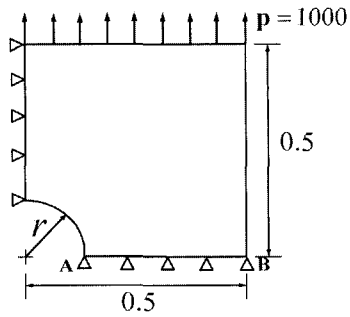


Fig. 12 The quarter model of a plate with a center hole under uniform uniaxial tension ($r = 0.1$)

employed for all calculations (we used $\delta_{j_0}^{up} = 0.001 \times \max(\mathbf{s}_{j_0})$ and $\delta_{j_0}^{low} = \delta^{up} \times 0.01$.)

Case study 1: rectangular panel with a circular hole

The stress concentration problem in a rectangular panel having a circular center hole is considered. A quarter of the rectangular panel is depicted in Fig. 12 with its boundary conditions prescribed. Young's modulus and Poisson's ratio of the panel are given as $E = 2.0 \times 10^8$ and $\nu = 0.3$, respectively. For the treatment of the circular boundary, we used the fictitious domain approach, and $\gamma = 0.00001$ was used for Eq. (8).

Fig. 13 shows the locations of the adaptively-added wavelets at the final resolution level for different numbers of subdomains (one may see in Jang et al.(2004a) and Jang (2004) how the nodal distributions vary as the resolution level increases.) The initial resolution levels of the adaptive analysis are assigned differently for each configuration depending on the number of subdomains: $j_0 = 1$ for $ND = 1 \times 1$, $j_0 = 2$ for $ND = 2 \times 2$, $j_0 = 3$, for $ND = 4 \times 4$, and $j_0 = 4$ for $ND = 8 \times 8$. The adaptive analysis proceeds up to the final resolution level of $J = 7$, i.e., 128×128 single-scale elements. The interpolation wavelets are densely located near the hole due to the stress concentration effect on the hole boundary. From Fig. 13, one can see that the distributions of the interpolation wavelets are different depending on the number of subdomains. This is due to the fact that more wavelets are used at the starting resolu-

tion level j_0 as the number of subdomains increases. For example, while 9 scaling functions and 16 wavelets of the resolution $j_0=1$ are initially used in case of $ND=1\times1$, 36 scaling functions and 64 wavelets of the resolution $j_0=2$ are placed as the basis functions in case of $ND=2\times2$.

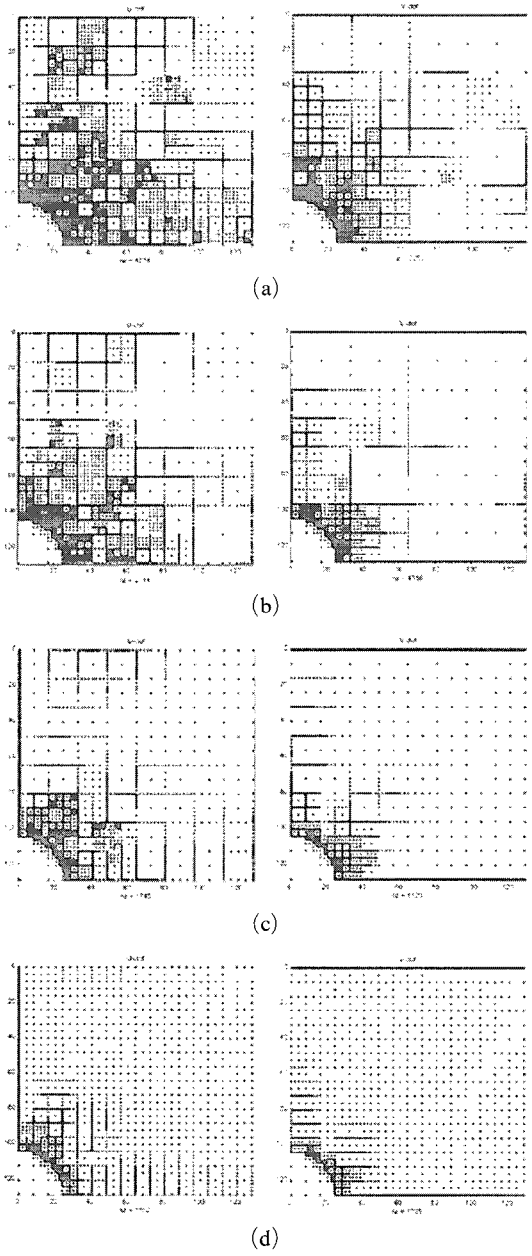


Fig. 13 The locations of the adaptively-inserted interpolation wavelets at the final resolution level for various numbers of subdomains: (a) $ND=1\times1$, (b) $ND=2\times2$, (c) $ND=4\times4$, (d) $ND=8\times8$ (left figures: wavelets for horizontal displacement, right figures: wavelets for vertical displacement)

To make sure the accuracy of the proposed approach, the solution obtained by using $ND=8\times8$ is compared with the converged ANSYS result obtained with 12,053 PLANE42 elements. Fig. 14 shows the vertical stress σ_y along the edge AB of the plate. As shown in the figure, the two methods give almost identical results.

Consumed CPU time and memory size according to the number of subdomains are listed in Table 1. These numerical results indicate that as the number of subdomains increases, the numerical cost for the analysis rapidly decreases. Nevertheless, using subdomains more than 8×8 (in this case) is not so meaningful since the room for adaptivity becomes smaller (when $ND=8\times8$ and $J=7$, maximum number of wavelets in each subdomain is only 17×17.)

Now the solution convergence is checked during the adaptive analysis. The L^2 error norms for

Table 1 Time cost and required computer memory size for Case study 1

ND	1×1	2×2	4×4	8×8
Time cost (sec)	11.296	4.203	2.421	1.593
Memory (MB)	191.1	71.0	38.9	28.5

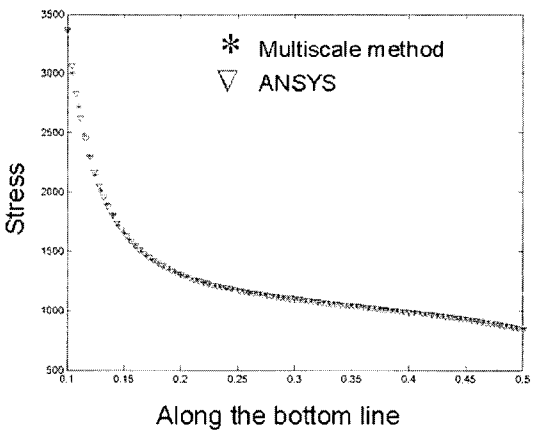


Fig. 14 Stress plot along the bottom edge $\overline{AB}$ of the plate in Fig. 12 ($ND=8\times8$)

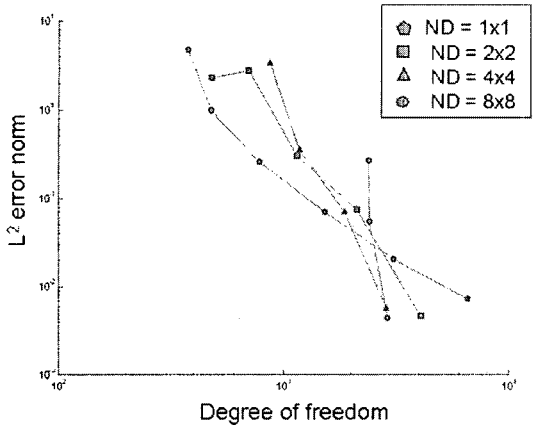


Fig. 15 The L^2 error norm for various numbers of subdomains

the different number of subdomains are plotted in Fig. 15. The L^2 error norm employed here is defined as

$$L^2=\sum_{e=1}^{NE}\frac{(\hat{\mathbf{U}}_j^e-\bar{\mathbf{U}}_j^e)^T\hat{\mathbf{K}}_j^e(\hat{\mathbf{U}}_j^e-\bar{\mathbf{U}}_j^e)}{(\bar{\mathbf{U}}_j^e)^T\hat{\mathbf{K}}_j^e\bar{\mathbf{U}}_j^e}\times 100(\%) \quad (33)$$

where $\hat{\mathbf{U}}_j^e$ is the element solution in single-scale form, $\bar{\mathbf{U}}_j^e$ is the reference solution obtained by using the non-adaptive analysis (or full analysis) at the highest resolution, and NE is the number of total elements. As can be seen in Fig. 15, when the number of used degrees of freedom is small, more accurate solutions are obtained for the case of using less number of subdomains. However, the situation is reversed when the number of used degrees of freedom is large. The behavior of the error norms in Fig. 15 is related to the orthogonality of wavelets. The interpolation wavelets in this work are not orthogonal unlike the popular Daubechies wavelets, so the interaction between long-scale and short-scale wavelets can affect the effectiveness of adaptivity. When the number of subdomains becomes larger, the number of interacting wavelets decreases, and more wavelets can be concentrated near the stress concentration region.

Case study 2 : hanger bracket

As the next example, we consider a hanger bracket problem in Fig. 16. The locations of the adaptively-inserted wavelets are depicted in

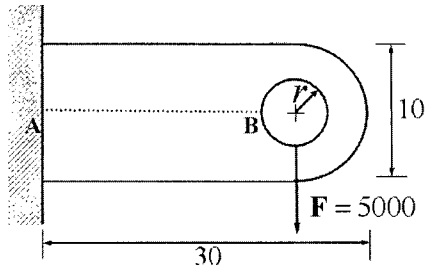


Fig. 16 Hanger bracket with a point load ($r=5$, $E=2.0\times10^8$, and $\nu=0.3$)

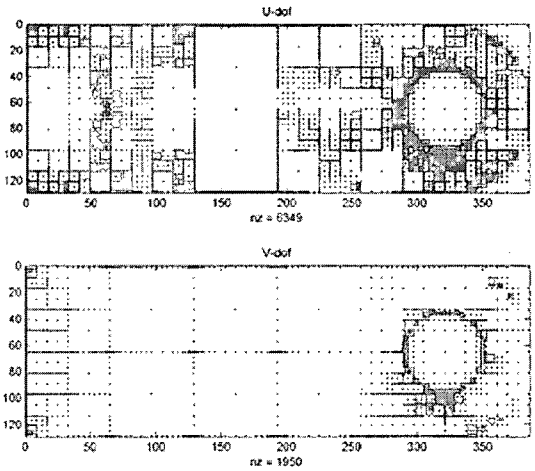


Fig. 17 The locations of the adaptively-added interpolation wavelets ($ND=3\times1$) (left figure : wavelets for horizontal displacement, right figure : wavelets for vertical displacement)

Fig. 17. Just as in the first example, the grid distribution is dense where the stress concentration is expected. Table 2 lists the computational time and the memory usage according to different subdomain configuration. In Fig. 18, the stress component σ_y along the line $\overline{AB}$ of the bracket calculated by using the proposed wavelet-Galerkin method is compared with the converged result of ANSYS (with 43,380 PLANE42 elements).

Table 2 Time cost and required computer memory size for Case study 2

ND	3×1	6×2	12×4
Time cost (sec)	16.687	7.843	4.750
Memory (MB)	274.7	137.8	96.6

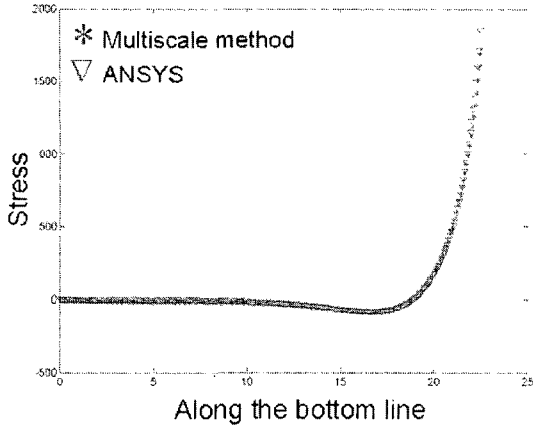


Fig. 18 Stress plot along the line $\overline{AB}$ of the hanger bracket in Fig. 16 ($ND=12\times4$)

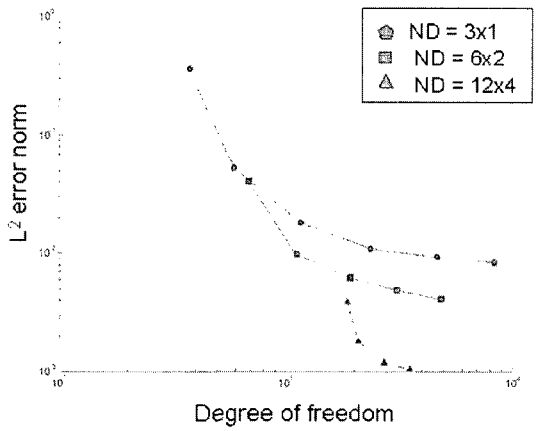


Fig. 19 The L^2 error norm for various numbers of subdomains

The L^2 error norms for different subdomain configuration are also illustrated in Fig. 19. The observations made in the first example are equally applied to this problem, so no further remark on the solution behavior will be given.

5. Conclusions

The implementation of the adaptive multiscale wavelet-Galerkin method using the subdomain technique was presented. The special attention was paid to give the detailed implementation procedure of the multiscale wavelet method. The effect of the proposed subdomain technique on the computation time, the required memory size, and the solution convergence was numerically investigated. When the wavelet method was implemented in the adaptive setting for large-sized numerical problems, the use of many subdomains increased the solution accuracy as well as it reduced the required CPU time and the memory size for the analysis. Since the proposed subdomain method is suitable for parallel processing, the parallelization of the subdomain-based wavelet-Galerkin method may be a good alternative to the one proposed here.

Acknowledgments

During the course of this study, SPARSKIT and Sparse BLAS were used, and we would like to thank the developers of the programs.

References

Basic Linear Algebra Subprograms Technical (BLAST) Forum, 1997, DRAFT Document for the Basic Linear Algebra Subprograms (BLAS) Standard.

Bertoluzza, S., 1997, "An Adaptive Collocation Method Based on Interpolating Wavelets," Multiscale Wavelet Methods for PDEs, Dahmen W, Kurdila AJ, Oswald P (eds). Academic Press, San Diego, pp. 109~135.

Carney, S., Heroux, M. A., Li, G., Pozo, R., Remington, K. A. and Wu, K., 1996, A Revised Proposal for Sparse BLAS Toolkit, SPARKER Working Note #3.

Christon, M. A. and Roach, D. W., 2000, "The Numerical Performance of Wavelets for PDEs: the Multi-scale Finite Element," *Computational Mechanics*, Vol. 25, pp. 230~244.

Cohen, A. and Masson. R., 1999, "Wavelet Methods for Second Order Elliptic Problems — Preconditioning and Adaptivity," *SIAM J. Sci. Comp.*, Vol. 21, pp. 1006~1026.

Cohen, A., Dahmen, W. and DeVore, R., 1998, Adaptive Wavelet Methods for Elliptic Operator Equations — Convergence Rates, RWTH Aachen, IGPM Preprint No. 165.

Dahmen, W., 2001, "Wavelet Methods for PDEs — Some Recent Developments," *Journal*

of Computational and Applied Mathematics, Vol. 128, pp. 133~185.

Diaz, A.R., 1999, "A Wavelet-Galerkin Scheme for Analysis of Large-Scale Problems on Simple Domains," *Int. J. Numer. Meth. Engng.*, Vol. 44, pp. 1599~1616.

Glowinski, R., Pan, T. W., Wells, R. O. Jr. and Zhou X., 1994, *Wavelet and Finite Element Solutions for the Neumann Problem Using Fictitious Domains*, Technical Report, Computational Mathematics Lab, Rice University.

Jang, G. W., 2004, *Shape and Topology Optimization Using the Multiscale Method*, Ph.D. Thesis, Seoul National University.

Jang, G. W., Kim, J. E. and Kim, Y. Y., 2004a, "Multiscale Galerkin Method Using Interpolation Wavelets for Two-Dimensional Elliptic Problems in General Domains," *Int. J. Numer. Meth. Engng.*, Vol. 59, pp. 225~253.

Jang, G. W., Kim, Y. Y. and Choi, K. K., 2004b, "Remesh-Free Shape Optimization Using the Wavelet-Galerkin Method," *International Journal of Solids and Structures*, Vol. 41, pp. 6465~6483.

Latto, A., Resnikoff, H. K. and Tenenbaum, E., 1991, "The Evaluation of Connection Coefficients of Compactly Supported Wavelets," *In Proceedings of the USA-French Workshop on Wavelets and Turbulance*, Princeton University.

Mallat, S., 1998, A Wavelet Tour of Signal Processing. Academic Press : London.

Saad, Y., 1994, SPARSKIT : A Basic Tool-Kit for Sparse Matrix Computations. VERSION2.