INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 24, 1391–1415 (1997)

MULTIRESOLUTION REPRODUCING KERNEL PARTICLE METHOD FOR COMPUTATIONAL FLUID DYNAMICS

WING KAM LIU, SUKKY JUN, DIRK THOMAS SIHLING, YIJUNG CHEN AND WEI HAO

Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, U.S.A.

SUMMARY

Multiresolution analysis based on the reproducing kernel particle method (RKPM) is developed for computational fluid dynamics. An algorithm incorporating multiple-scale adaptive refinement is introduced. The concept of using a wavelet solution as an error indicator is also presented. A few representative numerical examples are solved to illustrate the performance of this new meshless method. Results show that the RKPM is a good candidate for tackling the widespread large-scale problems in fluid dynamics. © 1997 by John Wiley & Sons, Ltd.

Int. J. Numer. Meth. Fluids, **24**: 1391–1415, 1997

No. of Figures: 14. No. of Tables: 0. No. of References: 29.

KEY WORDS: meshless kernel particle method; multiresolution analysis; wavelets; adaptivity; computational fluid dynamics

1. INTRODUCTION

The subject of computational fluid dynamics has been dominated by finite difference, finite volume and finite element methods for many years. Only recently has a new family of meshless particle methods emerged as an alternative to solve fluid dynamics problems numerically. A common characteristic of all the proposed particle methods is that they aim at enhancing the accuracy for high-gradient problems, circumventing the deficiencies of the finite element method such as mesh distortion for large-deformation problems, among others.

Smoothed particle hydrodynamics (SPH) is one of the particle methods that has been popularly used for astrodynamic simulation.[1–4] Some researchers[5–7] have applied SPH to problems of solid mechanics involving impact and penetration simulations. However, it is well known that the success of SPH is limited to problems in which boundaries do not play an important role.[3] By introducing a new kernel function with a boundary correction term, Liu *et al.*[8–10] have developed the reproducing kernel particle method (RKPM), which can handle boundaries without losing the consistency condition and can improve the accuracy of the solution as well. The application of the RKPM ranges from large deformation and structural acoustics to micromechanics and compressible flows.[8–13] The first comprehensive theoretical introduction to the RKPM and the relationship between the RKPM and wavelet transformation has been presented by Liu *et al.*,[10] which leads to the study of multiresolution analysis by wavelet-reproducing kernel methods. Liu *et al.*[8,9] extended the application of the RKPM to 1D elastic–perfectly plastic deformation and 2D dynamics. Convergence studies of moving least square kernel Galerkin methods and the multiple-scale RKPM are given in

*Correspondence to: W.K. Liu, Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, U.S.A.

CCC 0271–2091/97/121391–25 $17.50

Received 14 February 1996
Revised 8 April 1996

Reference 14 and References 8 and 15 respectively. Other related meshless methods include the element-free Galerkin (EFG) method proposed by Belytschko *et al.*,[16] hp-clouds by Duarte and Oden,[17] partition of unity finite element method by Babuska and Melenk[18], finite points method by Oñate *et al.*[19] and the free mesh method by Yagawa *et al.*[20]

Having all the features of meshless methods, the RKPM opens a new chapter of multiresolution analysis for particle methods based on wavelet theory. Liu and Oberste-Brandenburg[21] first proposed the concept of the multiple-scale RKPM in conjunction with wavelet analysis. Then Liu and Chen[22] extensively studied the multiresolution analysis by wavelet and reproducing kernel methods, including edge detection and aliasing control. Later, multiresolution analysis by the RKPM and hp-adaptivity by multiple-scale analysis were applied to structural acoustic problems by Liu *et al.*[8] A review of meshless kernel particle methods and wavelet analysis is presented by Liu *et al.*[15]

This paper is concerned with applications of the reproducing kernel particle method to computational fluid dynamics. It is our intention to employ multiresolution analysis based on discrete and continuous reproducing kernels, wavelets and integral window transforms to address some of the fundamental issues related to the dynamic analysis of compressible flows.

The composition of this paper is as follows. The general formulation of the RKPM is developed in Section 2. Section 3 is devoted to the streamline upwind Petrov–Galerkin formulation of the RKPM. Multiresolution analysis and multiple-scale adaptivity are given in Section 4. Several numerical examples and conclusions are presented in Sections 5 and 6 respectively.

2. FORMULATION OF REPRODUCING KERNEL PARTICLE METHOD (RKPM)

2.1. Kernel approximation

The kernel approximation of any function $u(\mathbf{x})$ is defined as

$$u^{\mathrm{R}}(\mathbf{x}) = \int_{-\infty}^{\infty} u(\tilde{\mathbf{x}})K(\mathbf{x}-\tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}}, \tag{1}$$

where $K(\mathbf{x}-\tilde{\mathbf{x}})$ is the kernel function. Equation (1) is also regarded as an integral transformation; alternatively, as the starting point for multiple-scale analysis, it is interpreted as a convolution. If the kernel function is the Dirac delta function $\delta(\mathbf{x}-\tilde{\mathbf{x}})$, the kernel approximation results in the trivial identity. To mimic the delta function, a Gaussian distribution function has usually been selected for the kernel function. A spline-type function is also a popular choice for the kernel function since it has a compact support.

We here consider the finite domain Ω rather than the infinite domain $[-\infty, \infty]$, which means that the kernel function is not defined outside the domain Ω. Within this finite region the kernel approximation can be written as

$$u^{\mathrm{R}}(\mathbf{x}) = \int_{\Omega} u(\mathbf{x}-\tilde{\mathbf{x}})K(\mathbf{x}-\tilde{\mathbf{x}}; a)\mathrm{d}\tilde{\mathbf{x}}, \tag{2}$$

where a is the dilation parameter that gives the kernel function possessing the property

$$K(\mathbf{x}-\tilde{\mathbf{x}}; a) \to \delta(\mathbf{x}) \quad \text{as } a \to 0. \tag{3}$$

The major contribution of $K(\mathbf{x}-\tilde{\mathbf{x}})$ to the kernel approximation is usually confined to the neighbourhood of $\tilde{\mathbf{x}}$ by adjusting the dilation parameter a, which is referred to as the smoothing length.[2] The kernel function also has the normalization property with respect to integration over the domain Ω i.e.

$$\int_{\Omega} K(\mathbf{x}; a)\mathrm{d}\mathbf{x} = 1. \tag{4}$$

However, this normalization property is not always satisfied, because the window function is cut off at the boundary of the finite domain problem. The integration of (4) is less than one if $\mathbf{x}$ is near the boundary of the domain Ω.

2.2. *Moments, correction function and window function*

In SPH the kernel approximation defined by (2) has a major deficiency because of the presence of the finite domain boundary as pointed out before. To remedy this deficiency, we introduce a new kernel function and a window function in the form

$$K(\mathbf{x}, \tilde{\mathbf{x}}) = C(\mathbf{x}, \tilde{\mathbf{x}})\phi(\mathbf{x} - \tilde{\mathbf{x}}) = [C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x}) \cdot (\mathbf{x} - \tilde{\mathbf{x}})]\phi(\mathbf{x} - \tilde{\mathbf{x}}), \tag{5}$$

where the correction function $C(\mathbf{x}, \tilde{\mathbf{x}})$ is assumed to be linear with respect to $\mathbf{x}$ and the window function $\phi(\mathbf{x} - \tilde{\mathbf{x}})$ is the same as the SPH kernel function introduced in the previous subsection. The properties of the window function are explained by Liu *et al.*[9] in detail. We will here define the correction function for the RKPM kernel function of (5) to satisfy the consistency conditions $u(\mathbf{x}) = 1$ and $u(\mathbf{x}) = x_i$ $(i = 1, \cdots, NP)$ everywhere in the domain regardless of the presence of the boundary, where NP is the number of particles. It has already been shown that this correction function enhances the accuracy of the discretized kernel representation not only near the boundary but also inside the entire domain.[9,10]

First, to satisfy the consistency condition $u(\mathbf{x}) = 1$, the new kernel function must have property

$$1 = \int_\Omega 1\, K(\mathbf{x}, \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}}, \tag{6}$$

which is identical with the normalization condition given by (4). The second consistency condition we impose on the RKPM kernel approximation is that $u(\mathbf{x}) = x_i$, which can be written as

$$\mathbf{x} = \int_\Omega \tilde{\mathbf{x}} K(\mathbf{x}, \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} = \mathbf{x} \int_\Omega K(\mathbf{x}, \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} - \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}}) K(\mathbf{x}, \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}}. \tag{7}$$

In order to satisfy the identity relationship, we must have the condition that the integration of the second term on the right-hand side becomes a zero vector, as well as the condition (6). The zero-vector condition is rewritten as

$$\int_\Omega (\mathbf{x} - \tilde{\mathbf{x}}) K(\mathbf{x}, \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}} = \mathbf{0} \tag{8}$$

Before we derive the correction function, it is convenient to first define the zeroth, first and second moments in terms of which the correction function is expressed. The zeroth moment is a scalar defined by the integration of the window function over the domain as

$$m_0(\mathbf{x}) \equiv \int_\Omega \phi(\mathbf{x} - \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}}. \tag{9}$$

The first-moment vector, whose dimension is NSD, is given by

$$\mathbf{m}_1(\mathbf{x}) \equiv \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}}) \phi(\mathbf{x} - \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}}. \tag{10}$$

The second moment is a tensor quantity of dimensions $NSD \times NSD$:

$$\mathbf{m}_2(\mathbf{x}) \equiv \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}})(\mathbf{x} - \tilde{\mathbf{x}}) \phi(\mathbf{x} - \tilde{\mathbf{x}}) \mathrm{d}\tilde{\mathbf{x}}. \tag{11}$$

To define the functions C_0 and $\mathbf{C}_1$, we insert our new kernel function (5) into (6) and (8). We then have

$$\begin{aligned} 1 &= \int_\Omega [C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x}) \cdot (\mathbf{x} - \tilde{\mathbf{x}})]\phi(\mathbf{x} - \tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}} \\ &= C_0(\mathbf{x}) \int_\Omega \phi(\mathbf{x} - \tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}} + \mathbf{C}_1(\mathbf{x}) \cdot \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}})\phi(\mathbf{x} - \tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}} \\ &= C_0(\mathbf{x})m_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x}) \cdot \mathbf{m}_1(\mathbf{x}) \end{aligned} \tag{12}$$

and from (8) we also have

$$\begin{aligned} \mathbf{0} &= \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}})[C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x}) \cdot (\mathbf{x} - \tilde{\mathbf{x}})]\phi(\mathbf{x} - \tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}} \\ &= C_0(\mathbf{x}) \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}})\phi(\mathbf{x} - \tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}} + \mathbf{C}_1(\mathbf{x}) \cdot \int_\Omega (\mathbf{x} - \tilde{\mathbf{x}})(\mathbf{x} - \tilde{\mathbf{x}})\phi(\mathbf{x} - \tilde{\mathbf{x}})\mathrm{d}\tilde{\mathbf{x}} \\ &= C_0(\mathbf{x})\mathbf{m}_1(\mathbf{x}) + \mathbf{C}_1(\mathbf{x}) \cdot \mathbf{m}_2(\mathbf{x}). \end{aligned} \tag{13}$$

With these two conditions (12) and (13) we can construct a set of linear algebraic equations for C_0 and $\mathbf{C}_1$ as

$$\begin{bmatrix} m_0(1 \times 1) & \mathbf{m}_1^{\mathrm{T}}(1 \times NSD) \\ \mathbf{m}_1(NSD \times 1) & \mathbf{m}_2(NSD \times NSD) \end{bmatrix} \begin{bmatrix} C_0(1 \times 1) \\ \mathbf{C}_1(NSD \times 1) \end{bmatrix} = \begin{bmatrix} 1(1 \times 1) \\ \mathbf{0}(NSD \times 1) \end{bmatrix}, \tag{14}$$

where the partial the partial dimensions are given in parentheses and superscript T denotes the transpose. If we define the total moment matrix as a $(1 + NSD) \times (1 + NSD)$ non-singular matrix

$$\mathbf{M}(\mathbf{x}) = \begin{bmatrix} m_0 & \mathbf{m}_1^{\mathrm{T}} \\ \mathbf{m}_1 & \mathbf{m}_2 \end{bmatrix}, \tag{15}$$

then we can finally get the expression for the correction function of the RKPM as

$$\text{RKPM correction function} = C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x}) \cdot (\mathbf{x} - \tilde{\mathbf{x}}), \tag{16}$$

where

$$\begin{bmatrix} C_0(\mathbf{x}) \\ \mathbf{C_1}(\mathbf{x}) \end{bmatrix} = \mathbf{M}^{-1}(\mathbf{x}) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}. \tag{17}$$

By using symbolic calculation, we can easily get the closed form for the correction function in terms of the moments defined by (9)–(11). Since we need to calculate the inverse of the matrix $\mathbf{M}(\mathbf{x})$, it is clear that we have to have a non-zero value for the determinant of $\mathbf{M}(\mathbf{x})$. This constraint requires that the dilation parameters (or smoothing length) be larger than its critical value. For example, a 1D window function must cover at least three different particles. This is called the kernel stability condition.[9] The correction function and moments are continuous functions of the variable $\mathbf{x}$. In References 9 and 10 it was shown that $C_0 = 1$ and $\mathbf{C}_1 = \mathbf{0}$ when the support of the window function $\phi(\mathbf{x} - \tilde{\mathbf{x}})$ is away from the boundary, whereas they are higher-order functions if the support is close to the boundary. It is noted that the definition of the correction function yields the same results as those Liu *et al.*[9,10] derived from moving least squares interpolants and linear polynomials.

2.3. Reproducing kernel particle interpolation function and its derivatives

By introducing the discretized kernel particle form, the kernel approximation is written by summation over the *NP* distinct particles as

$$u^{\mathrm{h}}(\mathbf{x}) = \sum_{J=1}^{NP} N_J(\mathbf{x}) u_J \tag{18}$$

and the reproducing kernel interpolation function is given by

$$N_J(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}_J)\phi(\mathbf{x} - \mathbf{x}_J)\Delta V_J, \tag{19}$$

where $\mathbf{x}_J$ is the position vector and $\Delta\mathbf{V}_J$ the lumped volume element of the Jth particle. Liu *et al.*[10] showed that the discretized correction function $C(\mathbf{x}, \mathbf{x}_J)$ apparently enhances the accuracy and stability of kernel interpolation around the boundary. The derivatives of the reproducing kernel interpolation function can be easily obtained by taking the derivatives of the correction function and window function. For example,

$$N_{J,\mathbf{x}}(\mathbf{x}) = \left[C_{,\mathbf{x}}(\mathbf{x}, \mathbf{x}_J; a)\phi\left(\frac{\mathbf{x} - \mathbf{x}_J}{a}\right) + C(\mathbf{x}, \mathbf{x}_J; a)\phi_{,\mathbf{x}}\left(\frac{\mathbf{x} - \mathbf{x}_J}{a}\right)\right]\Delta V_J. \tag{20}$$

The 3D computations of the derivatives of the correction function and window function are given in detail by Liu *et al.*[9,15]

3. FUNDAMENTAL EQUATIONS FOR COMPUTATIONAL FLUID DYNAMICS

3.1. Strong form of Navier–Stokes equations

For numerical analysis it is useful to start with the Navier–Stokes (N–S) equations in conservation form. The equations for the conservation of mass, momentum and energy are

$$\rho_{,t} + (\rho u_j)_{,j} = 0 \quad \text{(mass)}, \tag{21a}$$

$$(\rho u_i)_{,t} + (\rho u_i u_j)_{,j} + p_{,i} = \tau_{ij,j} + \rho b_i \quad \text{(momentum)}, \tag{21b}$$

$$(\rho e)_{,t} + (\rho u_i e)_{,i} + (p u_i)_{,i} = (\tau_{ij} u_j)_{,i} - q_{i,i} + \rho b_j u_j + \rho r \quad \text{(energy)}, \tag{21c}$$

where ρ is the density, $\mathbf{u} = [u_1, u_2, u_3]^{\mathrm{T}}$ is the velocity vector, p is the thermodynamic pressure, $\tau = [\tau_{ij}]$ is the stress tensor, $\mathbf{b} = [b_1, b_2, b_3]^{\mathrm{T}}$ is the body force vector, e is the total energy density, $\mathbf{q} = [q_1, q_2, q_3]^{\mathrm{T}}$ is the heat flux vector and r is the heat source.

So far, equations (21) still depend on the co-ordinates and the solution variables. To write the equations only in terms of the unknowns in which we are interested, we start by defining the conservation variables $\mathbf{U} = [U_1, U_2, U_3, U_4]^{\mathrm{T}}$ for 2D problems.

The following derivation of equations and matrices follows Shakib *et al.*[23] and is only applicable to 2D problems. Note that the problem to be solved does not contain any body forces or heat sources and these terms are therefore omitted.

$$\mathbf{U} = \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{Bmatrix} = \rho \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ e \end{Bmatrix}. \tag{22}$$

It is also useful to define the Euler and diffusive flux vectors

$$\mathbf{F}_i = \rho u_i \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ e \end{Bmatrix} + p \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ u_i \end{Bmatrix}, \tag{23}$$

$$\mathbf{F}_i^{\mathrm{d}} = \begin{Bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{ij}u_j \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ -q_i \end{Bmatrix}. \tag{24}$$

With the definitions (22)–(24), equations (21) can be rewritten as

$$\mathbf{U}_{,t} + \mathbf{F}_{i,i} = \mathbf{F}_{i,i}^{\mathrm{d}}. \tag{25}$$

To describe the material, the following constitutive equations are used:

$$\imath = c_{\mathrm{v}}\theta, \tag{26}$$

$$p = (\gamma - 1)\rho\imath, \tag{27}$$

$$\tau_{ij} = \lambda u_{k,k}\delta_{ij} + \mu(u_{i,j} + u_{j,i}), \tag{28}$$

$$q_i = -\kappa\theta_{,i}, \tag{29}$$

where $\imath$ is the internal energy density, c_{v} is the heat capacity at constant volume, θ is the absolute temperature, γ is the ratio of specific heats (defined as $\gamma = c_{\mathrm{p}}/c_{\mathrm{v}}$, where c_{p} is the heat capacity at constant pressure), λ and μ are the viscosity coefficients (for isotropic materials, $\lambda = \frac{2}{3}\mu$) and κ is the coefficient of thermal conductivity. Equations (26) and (27) together represent the perfect gas law, equation (28) defines the viscous stress components and equation (29) is Fourier's law of heat conduction.

To get equation (25) in a form dependent only on the conservation variables, the chain rule is used on the Euler flux term, i.e.

$$\mathbf{F}_{i,i} = \mathbf{F}_{i,\mathbf{U}}\mathbf{U}_{,i},$$

and the Euler Jacobian matrices $\mathbf{A}_i$ are defined as

$$\mathbf{A}_i = \mathbf{F}_{i,\mathbf{U}}. \tag{30}$$

The diffusive flux is assumed to be proportional to the first derivatives of the conservation variables and is therefore defined as

$$\mathbf{F}_i^{\mathrm{d}} = \mathbf{K}_{ij}\mathbf{U}_{,j}, \tag{31}$$

where $\mathbf{K} = [\mathbf{K}_{ij}]$ is the diffusivity matrix.

With the definitions (30) and (31), equation (25) can be written in quasi-linear form as

$$\mathbf{U}_{,t} + \mathbf{A}_i\mathbf{U}_{,i} = (\mathbf{K}_{ij}\mathbf{U}_{,j})_{,i}. \tag{32}$$

3.2. Development of the variational form

The governing equations cannot be solved in the strong form, so it is necessary to establish a variational form to get a matrix equation that can be solved with the numerical scheme considered here. This approach uses the widely used Bubnov–Galerkin approximation[24] and a stabilizing term of the streamline upwind type. The result is known as the streamline upwind Petrov–Galerkin (SUPG) formulation.

3.2.1. Euler equations

Galerkin formulation

For the Euler equations the diffusive flux is not considered and the strong form is

$$\mathbf{U}_{,t} + \mathbf{F}_{i,i} = \mathbf{0} \quad \text{on } \Omega\times]0, T[, \tag{33}$$

$$\mathbf{U} = \mathbf{g} \quad \text{on } \Gamma_{\mathrm{g}}\times]0, T[, \tag{34}$$

$$-\mathbf{F}_n = \mathbf{h} \quad \text{on } \Gamma_{\mathrm{h}}\times]0, T[, \tag{35}$$

where $[0, T]$ is the time interval of interest, Ω is an open set with a piecewise smooth boundary Γ and

$$\Gamma_{\mathrm{g}} \cup \Gamma_{\mathrm{h}} = \Gamma \quad \text{is the boundary,} \tag{36}$$

$$\Gamma_{\mathrm{g}} \cap \Gamma_{\mathrm{h}} = \emptyset \quad \text{is an empty set,} \tag{37}$$

$$\Omega \cup \Gamma = \bar{\Omega} \quad \text{is the total domain,} \tag{38}$$

The boundary conditions for inviscid flow problems are not as crucial as for viscous flow and the weak formulation used here is stable enough for simple explicit time integration schemes. For this formulation the flux term $\mathbf{F}_{i,i}$ is first written as $\mathbf{A}_i\mathbf{U}_{,i}$:

$$\mathbf{U}_{,t} + \mathbf{A}_i\mathbf{U}_{,i} = \mathbf{0} \quad \text{in } \Omega. \tag{39}$$

To get a variational form, we need two sets of functions, one for the trial functions $\mathbf{U}$ and one for the weighting functions $\mathbf{w}$. Both $\mathbf{U}$ and $\mathbf{w}$ are H^1 functions, i.e. $\mathbf{U}, \mathbf{w} \in H^1$.

Next, equation (39) is premultiplied by the weighting function and integrated over the computational domain, yielding the variational form of the Euler equations:

$$\int_\Omega \mathbf{w}^{\mathrm{T}}(\mathbf{U}_{,t} + \mathbf{A}_i\mathbf{U}_{,i})\mathrm{d}\Omega = 0. \tag{40}$$

Streamline upwind operator

The SUPG formulation used here was first introduced in Reference 25 and later described in more detail in Reference 26. The purpose of the streamline upwind term is to dissipate numerical noise in the main flow direction. Therefore a perturbation of the weighting or trial function is applied to the strong form. One of the choices used by Hughes and Tezduyar[26] is

$$\mathbf{p} = \mathbf{T}_k\mathbf{w}_{,k}, \quad \text{with } \mathbf{T}_k = \tau_{\mathrm{k}}[\mathbf{A}_k]^{\mathrm{T}},$$

where $\mathbf{p}$ is the perturbation of the weighting function and τ_k is a parameter selected to improve accuracy according to some criterion.

The multiplication of equation (39) by $\mathbf{p}$ and integration over the domain yields

$$\int_\Omega (\mathbf{T}_k\mathbf{w}_{,k})^{\mathrm{T}}(\mathbf{U}_{,t} + \mathbf{A}_i\mathbf{U}_{,i})\mathrm{d}\Omega = 0. \tag{41}$$

SUPG form of the Euler equations

The terms of (40) and (41) together represent the streamline upwind Petrov–Galerkin formulation of the Euler equations:

$$\int_\Omega \mathbf{w}^{\mathrm{T}}\mathbf{U}_{,t}\mathrm{d}\Omega + \int_\Omega \mathbf{w}^{\mathrm{T}}\mathbf{A}_i\mathbf{U}_{,i}\mathrm{d}\Omega + \int_\Omega \mathbf{w}_{,k}^{\mathrm{T}}\mathbf{T}_k^{\mathrm{T}}\mathbf{U}_{,t}\mathrm{d}\Omega + \int_\Omega \mathbf{w}_{,k}^{\mathrm{T}}\mathbf{T}_k^{\mathrm{T}}\mathbf{A}_i\mathbf{U}_{,i}\mathrm{d}\Omega = 0. \tag{42}$$

3.2.2. Navier–Stokes equations

Bubnov–Galerkin formulation

For the strong form of the problem considered, equation (32) is rewritten as

$$\mathbf{U}_{,t} + \mathbf{F}_{i,i} - \mathbf{F}^{\mathrm{d}}_{i,i} = \mathbf{0} \quad \text{in } \Omega. \tag{43}$$

Next, equation (43) is multiplied by a weighting function and integrated over the whole domain:

$$\int_{\Omega} \mathbf{w}^{\mathrm{T}}(\mathbf{U}_{,t} + \mathbf{F}_{i,i} - \mathbf{F}^{\mathrm{d}}_{i,i})\mathrm{d}\Omega = 0. \tag{44}$$

Integration by parts is used only on the diffusive flux term $\mathbf{F}^{\mathrm{d}}_{i,i}$ in order to get a formulation that helps to impose the boundary conditions:

$$\int_{\Omega} \mathbf{w}^{\mathrm{T}}\mathbf{F}^{\mathrm{d}}_{i,i}\mathrm{d}\Omega = \int_{\Omega} (\mathbf{w}^{\mathrm{T}}\mathbf{F}^{\mathrm{d}}_{i})_{,i}\mathrm{d}\Omega - \int_{\Omega} \mathbf{w}^{\mathrm{T}}_{,i}\mathbf{F}^{\mathrm{d}}_{i}\mathrm{d}\Omega \tag{45a}$$

$$= \int_{\Gamma_{\mathrm{h}}} \mathbf{w}^{\mathrm{T}}\mathbf{F}^{\mathrm{d}}_{i}n_i\mathrm{d}\Gamma + \int_{\Gamma_{\mathrm{g}}} \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{h}}\mathrm{d}\Gamma - \int_{\Omega} \mathbf{w}^{\mathrm{T}}_{,i}\mathbf{F}^{\mathrm{d}}_{i}\mathrm{d}\Omega. \tag{45b}$$

The Galerkin form of (43) looks as follows:

$$\int_{\Omega} \mathbf{w}^{\mathrm{T}}\mathbf{U}_{,t}\mathrm{d}\Omega + \int_{\Omega} \mathbf{w}^{\mathrm{T}}\mathrm{F}_{i,i}\mathrm{d}\Omega + \int_{\Omega} \mathbf{w}^{\mathrm{T}}_{,i}\mathbf{F}^{\mathrm{d}}_{i}\mathrm{d}\Omega = \int_{\Gamma_{\mathrm{h}}} \mathbf{w}^{\mathrm{T}}\mathbf{F}^{\mathrm{d}}_{i}n_i\mathrm{d}\Gamma + \int_{\Gamma} \mathrm{g}\, \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{h}}\mathrm{d}\Gamma. \tag{46}$$

With the linearization of the flux terms, equation (46) can be written in the form

$$\int_{\Omega} \mathbf{w}^{\mathrm{T}}\mathbf{U}_{,t}\mathrm{d}\Omega + \int_{\Omega} \mathbf{w}^{\mathrm{T}}\mathbf{A}_i\mathbf{U}_{,i}\mathrm{d}\Omega + \int_{\Omega} \mathbf{w}^{\mathrm{T}}_{,i}\mathbf{K}_{ij}\mathbf{U}_{,j}\mathrm{d}\Omega = \int_{\Gamma_{\mathrm{h}}} \mathbf{w}^{\mathrm{T}}\mathbf{F}^{\mathrm{d}}_{i}n_i\mathrm{d}\Gamma + \int_{\Gamma_{\mathrm{g}}} \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{h}}\mathrm{d}\Gamma. \tag{47}$$

Since the shape function of the RKPM does not satisfy the Kronecker delta condition

$$N_I(\mathbf{x}_J) \neq \delta_{IJ}, \tag{48}$$

we need the extra condition

$$\int_{\Gamma_{\mathrm{g}}} \delta\tilde{\mathbf{h}}(\mathbf{U} - \mathbf{g})\mathrm{d}\Gamma = 0. \tag{49}$$

Remark. The essential boundary term $\int_{\Gamma_{\mathrm{g}}} \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{h}}d\Gamma$ brings about the distinct difference between the RKPM and classical finite element methods, which generally do not include this term in the variational formulations.

Streamline upwind operator

The perturbation used for the Navier–Stokes equations is essentially the same as for the Euler equations, but the design of τ_k is more complicated since it has to take the viscous term into account. A general design for τ_k is shown in Reference 23.

The terms $\mathbf{U}_{,t} + \mathbf{F}_{i,i}$ of (43) are multiplied by $\mathbf{p}$, giving

$$\int_{\Omega} (\mathbf{T}_k\mathbf{w}_{,k})^{\mathrm{T}}(\mathbf{U}_{,t} + \mathbf{F}_{i,i})\mathrm{d}\Omega = 0. \tag{50}$$

Using $\mathbf{F}_{i,i} = \mathbf{A}_i \mathbf{U}_{,i}$ gives

$$\int_\Omega (\mathbf{T}_k \mathbf{w}_{,k})^{\mathrm{T}} (\mathbf{U}_{,t} + \mathbf{A}_i \mathbf{U}_{,i}) \mathrm{d}\Omega = 0. \tag{51}$$

SUPG Form

Now all terms are collected and the variational form can be written as

$$\begin{aligned} &\int_\Omega \mathbf{w}^{\mathrm{T}} \mathbf{U}_{,t} \mathrm{d}\Omega + \int_\Omega \mathbf{w}^{\mathrm{T}} \mathbf{A}_i \mathbf{U}_{,i} \mathrm{d}\Omega + \int_\Omega \mathbf{w}_{,i}^{\mathrm{T}} \mathbf{K}_{ij} \mathbf{U}_{,j} \mathrm{d}\Omega + \int_\Omega \mathbf{w}_{,k}^{\mathrm{T}} \mathbf{T}_k^{\mathrm{T}} \mathbf{U}_{,t} \mathrm{d}\Omega + \int_\Omega \mathbf{w}_{,k}^{\mathrm{T}} \mathbf{T}_k^{\mathrm{T}} \mathbf{A}_i \mathbf{U}_{,i} \mathrm{d}\Omega \\ &\quad = \int_{\Gamma_{\mathrm{h}}} \mathbf{w}^{\mathrm{T}} \mathbf{F}_i^{\mathrm{d}} n_i \mathrm{d}\Gamma + \int_{\Gamma_{\mathrm{g}}} \mathbf{w}^{\mathrm{T}} \tilde{\mathbf{h}} \mathrm{d}\Gamma. \end{aligned} \tag{52}$$

4. DISCRETIZATION

4.1. Euler equations

The variational form (42) is discretized via the nodal distribution in the domain.

$$\int_\Omega (\mathbf{w}^{\mathrm{h}})^{\mathrm{T}} \mathbf{U}_{,t}^{\mathrm{h}} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}^{\mathrm{h}})^{\mathrm{T}} \mathbf{A}_i \mathbf{U}_{,i}^{\mathrm{h}} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}_{,k}^{\mathrm{h}})^{\mathrm{T}} \mathbf{T}_k^{\mathrm{T}} \mathbf{U}_{,t}^{\mathrm{h}} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}_{,k}^{\mathrm{h}})^{\mathrm{T}} \mathbf{T}_k^{\mathrm{T}} \mathbf{A}_i \mathbf{U}_{,i}^{\mathrm{h}} \mathrm{d}\Omega = 0. \tag{53}$$

The discrete trial and weighting functions can be written in terms of shape functions and nodal values:

$$\begin{aligned} \mathbf{w}^{\mathrm{h}} &= \sum_{I=1}^{NNP} N_I \mathbf{w}_I, & \mathbf{w}_{,i}^{\mathrm{h}} &= \sum_{I=1}^{NNP} N_{I,i} \mathbf{w}_I, \\ \mathbf{U}^{\mathrm{h}} &= \sum_{J=1}^{NNP} N_J \mathbf{U}_J, & \mathbf{U}_{,i}^{\mathrm{h}} &= \sum_{J=1}^{NNP} N_{J,i} \mathbf{U}_J, & \mathbf{U}_{,t}^{\mathrm{h}} &= \sum_{J=1}^{NNP} N_J \dot{\mathbf{U}}_J. \end{aligned} \tag{54}$$

With the definitions (54), equation (53) can be written in terms of discrete vector values at the nodes:

$$\begin{aligned} \sum_{I=1}^{NNP} \sum_{J=1}^{NNP} \Bigg(&\int_\Omega \mathbf{w}_I^{\mathrm{T}} N_I N_J \dot{\mathbf{U}}_J \mathrm{d}\Omega + \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_I \mathbf{A}_i N_{J,i} \mathbf{U}_J \mathrm{d}\Omega \\ &+ \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_{I,k} \mathbf{T}_k^{\mathrm{T}} N_J \dot{\mathbf{U}}_J \mathrm{d}\Omega + \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_{I,k} \mathbf{T}_k^{\mathrm{T}} \mathbf{A}_i N_{J,i} \mathbf{U}_J \mathrm{d}\Omega \Bigg) = 0. \end{aligned} \tag{55}$$

The nodal vectors have no influence on the integration and can therefore be taken out of the integrals. The vector $\mathbf{w}_I$ is the same for all integrals and can be taken out of the inner summation:

$$\begin{aligned} \sum_{I=1}^{NNP} \mathbf{w}_I^{\mathrm{T}} \sum_{J=1}^{NNP} \Bigg(&\int_\Omega N_I N_J \mathrm{d}\Omega \dot{\mathbf{U}}_J + \int_\Omega N_I \mathbf{A}_i N_{J,i} \mathrm{d}\Omega \mathbf{U}_J + \int_\Omega N_{I,k} \mathbf{T}_k^{\mathrm{T}} N_J \mathrm{d}\Omega \dot{\mathbf{U}}_J \\ &+ \int_\Omega N_{I,k} \mathbf{T}_k^{\mathrm{T}} \mathbf{A}_i N_{J,i} \mathrm{d}\Omega \mathbf{U}_J \Bigg) = 0. \end{aligned} \tag{56}$$

The integrals that are multiplied by $\dot{\mathbf{U}}_J$ can be summed up to build the mass matrices $\mathbf{M}_{IJ}$ and the integrals that belong to $\mathbf{U}_J$ build the convection matrices $\mathbf{C}_{IJ}$:

$$\mathbf{M}_{IJ} = \int_\Omega (N_I \mathbf{I} + N_{I,k}\mathbf{T}_k^{\mathrm{T}}) N_J \mathrm{d}\Omega \quad \text{(mass matrix)}, \tag{57a}$$

$$\mathbf{C}_{IJ} = \int_\Omega (N_I \mathbf{I} + N_{I,k}\mathbf{T}_k^{\mathrm{T}}) \mathbf{A}_i N_{J,i} \mathrm{d}\Omega \quad \text{(convection matrix)}. \tag{57b}$$

Thus equation (56) can be simplified and written in matrix form as

$$\sum_{I=1}^{NNP} \mathbf{w}_I^{\mathrm{T}} \sum_{J=1}^{NNP} (\mathbf{M}_{IJ}\dot{\mathbf{U}}_J + \mathbf{C}_{IJ}\mathbf{U}_J) = 0. \tag{58}$$

Because the nodal vectors $\mathbf{w}_I$ are arbitrary variations of the solution, they and the outer summation are dropped and the system of equations that needs to be solved can be obtained as

$$\sum_{J=1}^{NNP} (\mathbf{M}_{IJ}\dot{\mathbf{U}}_J + \mathbf{C}_{IJ}\mathbf{U}_J) = \mathbf{0}. \tag{59}$$

4.2. Navier–Stokes Equations

The same steps need to be done for the weak form of the Navier–Stokes equations. The same function sets and shape functions as in Section 4.1 are used. The discrete form of (52) is

$$\begin{aligned}\int_\Omega (\mathbf{w}^{\mathrm{h}})^{\mathrm{T}} \mathbf{U}^{\mathrm{h}}_{,t} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}^{\mathrm{h}})^{\mathrm{T}} \mathbf{A}_i \mathbf{U}^{\mathrm{h}}_{,i} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}^{\mathrm{h}}_{,i})^{\mathrm{T}} \mathbf{K}_{ij} \mathbf{U}^{\mathrm{h}}_{,j} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}^{\mathrm{h}}_{,k})^{\mathrm{T}} \mathbf{T}_k^{\mathrm{T}} \mathbf{U}^{\mathrm{h}}_{,t} \mathrm{d}\Omega + \int_\Omega (\mathbf{w}^{\mathrm{h}}_{,k})^{\mathrm{T}} \mathbf{T}_k^{\mathrm{T}} \mathbf{A}_i \mathbf{U}^{\mathrm{h}}_{,i} \mathrm{d}\Omega \\ = \int_{\Gamma_{\mathrm{h}}} (\mathbf{w}^{\mathrm{h}})^{\mathrm{T}} \mathbf{F}_i^{\mathrm{d}} n_i \mathrm{d}\Gamma + \int_{\Gamma_{\mathrm{g}}} \mathbf{w}^{\mathrm{T}} \tilde{\mathbf{h}} \mathrm{d}\Gamma\end{aligned} \tag{60}$$

and

$$\int_{\Gamma_{\mathrm{g}}} \delta\tilde{\mathbf{h}} (\mathbf{U} - \mathbf{g}) \mathrm{d}\Gamma = 0. \tag{61}$$

Here we employ the interpolation of the traction function on the essential boundary condition as

$$\tilde{\mathbf{h}}^{\mathrm{h}} = \sum_K^{ng} N_K \tilde{\mathbf{h}}_K \quad \text{and} \quad \delta\tilde{\mathbf{h}}^{\mathrm{h}} = \sum_K^{ng} N_K \delta\tilde{\mathbf{h}}_K. \tag{62}$$

Substituting the shape functions into (60) yields

$$\begin{aligned}\sum_{I=1}^{NNP} \Bigg[\sum_{J=1}^{NNP} \Bigg(\int_\Omega \mathbf{w}_I^{\mathrm{T}} N_I N_J \dot{\mathbf{U}}_J \mathrm{d}\Omega + \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_I \mathbf{A}_i N_{J,i} \mathbf{U}_J \mathrm{d}\Omega + \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_{I,i} \mathbf{K}_{ij} N_{J,j} \mathbf{U}_J \mathrm{d}\Omega \\ + \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_{I,k} \mathbf{T}_k^{\mathrm{T}} N_J \dot{\mathbf{U}}_J \mathrm{d}\Omega + \int_\Omega \mathbf{w}_I^{\mathrm{T}} N_{I,k} \mathbf{T}_k^{\mathrm{T}} \mathbf{A}_i N_{J,i} \mathbf{U}_J \mathrm{d}\Omega \Bigg) - \int_{\Gamma_{\mathrm{h}}} \mathbf{w}_I^{\mathrm{T}} N_I \mathbf{F}_i^{\mathrm{d}} n_i \mathrm{d}\Gamma \\ - \int_{\Gamma_{\mathrm{g}}} \mathbf{w}_I^{\mathrm{T}} N_I N_K \tilde{\mathbf{h}}_K \mathrm{d}\Gamma \Bigg] = 0.\end{aligned} \tag{63}$$

Again the vectors $\mathbf{w}_I$, $\mathbf{U}_J$ and $\dot{\mathbf{U}}_J$ are taken out of the integrals and the terms belonging to $\dot{\mathbf{U}}_J$ and $\mathbf{U}_J$ are collected:

$$\sum_{I=1}^{NNP} \mathbf{w}_I^{\mathrm{T}} \left[\sum_{J=1}^{NNP} \left(\int_\Omega (N_I \mathbf{I} + N_{I,k} \mathbf{T}_k^{\mathrm{T}}) N_J \mathrm{d}\Omega \dot{\mathbf{U}}_J + \int_\Omega N_{I,i} \mathbf{K}_{ij} N_{J,j} \mathrm{d}\Omega \mathbf{U}_J \right.\right.$$
$$\left.\left. + \int_\Omega (N_I \mathbf{I} + N_{I,k} \mathbf{T}_k^{\mathrm{T}}) \mathbf{A}_i N_{J,i} \mathrm{d}\Omega \mathbf{U}_J \right) - \int_{\Gamma_{\mathrm{h}}} N_I \mathbf{F}_i^{\mathrm{d}} n_i \mathrm{d}\Gamma - \int_{\Gamma_{\mathrm{g}}} \mathbf{w}_I^{\mathrm{T}} N_I N_K \tilde{\mathbf{h}}_K \mathrm{d}\Gamma \right] = 0. \quad (64)$$

The following matrices are defined to simplify the notation. This time there are the additional diffusion matrices $\mathbf{D}_{IJ}$ and the force vector $\mathbf{F}_I$ resulting from the natural boundary conditions.

$$\mathbf{M}_{IJ} = \int_\Omega (N_1 \mathbf{I} + \mathbf{T}_k N_{I,k})^{\mathrm{T}} N_J \mathrm{d}\Omega \quad \text{(mass matrix)}, \quad (65a)$$

$$\mathbf{C}_{IJ} = \int_\Omega (N_I \mathbf{I} + \mathbf{T}_k N_{I,k})^{\mathrm{T}} \mathbf{A}_i N_{J,i} \mathrm{d}\Omega \quad \text{(convection matrix)}, \quad (65b)$$

$$\mathbf{D}_{IJ} = \int_\Omega N_{I,i} \mathbf{K}_{ij} N_{J,j} \mathrm{d}\Omega \quad \text{(diffusive matrix)}, \quad (65c)$$

$$\mathbf{F}_I = \int_{\Gamma_{\mathrm{h}}} N_I \mathbf{F}_i^{\mathrm{d}} n_i \mathrm{d}\Gamma \quad \text{(traction boundary)}, \quad (65d)$$

$$\tilde{\mathbf{M}}_{IJ} = \int_{\Gamma_{\mathrm{g}}} N_I N_K \mathrm{d}\Gamma. \quad (65e)$$

With the definitions (65) the variational form can be written as

$$\sum_{I=1}^{NNP} \mathbf{w}_I^{\mathrm{T}} \left(\sum_{J=1}^{NNP} [\mathbf{M}_{IJ} \dot{\mathbf{U}}_J + (\mathbf{C}_{IJ} + \mathbf{D}_{IJ}) \mathbf{U}_J] - \mathbf{F}_I - \tilde{\mathbf{M}}_{IJ} \tilde{\mathbf{h}}_J \right) = 0. \quad (66)$$

Because the vectors $\mathbf{w}_I$ are arbitrary, it necessarily follows that

$$\sum_{J=1}^{NNP} [\mathbf{M}_{IJ} \dot{\mathbf{U}}_J + (\mathbf{C}_{IJ} + \mathbf{D}_{IJ}) \mathbf{U}_J] = \mathbf{F}_I + \tilde{\mathbf{M}}_{IJ} \tilde{\mathbf{h}}_J. \quad (67)$$

From the essential boundary condition we also have the extra matrix equation

$$\tilde{\mathbf{M}}_{KI} \mathbf{U}_I = \mathbf{G}_K. \quad (68)$$

Remark. In computation, equations (67) and (68) are solved in a coupled manner in order to satisfy the enforced essential and natural boundary conditions simultaneously.

5. ADAPTIVE REFINEMENT

In order to get sufficiently accurate numerical solutions with a minimum of memory and computational time, a grid as coarse as possible is needed. A high particle density is usually needed only in a few small regions of the domain. Therefore the programme needs an algorithm that is capable of detecting these areas and also a strategy to insert particles to capture the response that could not be resolved by the original particle distribution.

5.1. Multiresolution analysis

To understand the ability of the RKPM to be used for multiresolution analysis, it is necessary to look at its behaviour in the frequency or Fourier domain. The key to this ability is the dilation parameter a of the window function $\phi_a(x; x-y)$.

The RKPM can be viewed as a modified convolution formulae that can be applied in finite domains for arbitrary discretizations. The kernel function acts like a lowpass filter for the solution in this system. By changing the dilation parameter a, it is possible to construct a series of lowpass filters that provide different frequency parts of the solution. Subtracting two low-frequency solution parts that are obtained for two different dilations a, e.g. a and $2a$, we get the high-frequency part, which is simply the difference between these two scales.

This high-frequency part contains that part of the solution that is close to the resolution limit of the current particle distribution and therefore indicates the areas in the domain that need to be refined.

Another advantage is that aliasing shows up in the high-scale part of the solution. Thus, by refining the distribution in these areas, the amount of aliasing is decreased.

The window function with a dilation parameter looks as follows:

$$\phi_a(x-x_j) = \frac{1}{a\Delta x_j}\phi\left(\frac{x-x_j}{a\Delta x_j}\right). \tag{69}$$

It can easily be seen that by changing a, the window function changes its shape. The correction function is applied to the window function and we get the modified window $\bar{\phi}_a$. Then RKPM is

$$u^{\mathrm{R}_a}(x) = P_a u(x) = \sum_{j=1}^{NNP} u(x_j)\bar{\phi}_a(x; x-x_j)\Delta x_j, \tag{70}$$

where $\bar{\phi}_a(x; x-x_j)$ is the kernel function

$$\bar{\phi}_a(x; x-x_j) = C_a(x; x-x_j)\phi_a(x-x_j) \tag{71}$$

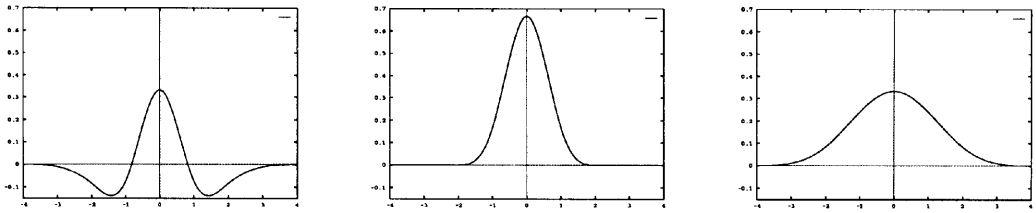
and P_a can be considered as a projection operator for the scale a. With the previously mentioned change of dilation we can obtain the following relation for the projections on different scales:

$$P_a u(x) \supset P_{2a}u(x) \supset P_{4a}u(x) \supset \cdots \supset \lim_{n\to\infty} P_{2^n a}u(x) = \{\emptyset\}. \tag{72}$$

A wavelet function can be defined by

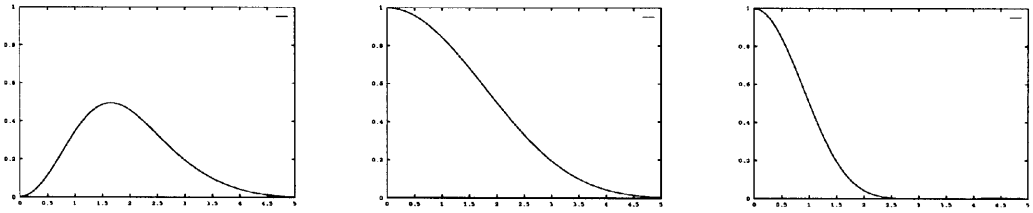
$$\bar{\psi}_{2a}(x; x-x_j) = \bar{\phi}_a(x; x-x_j) - \bar{\phi}_{2a}(x; x-x_j). \tag{73}$$

Figures 1 and 2 show this definition for a cubic spline in the space and frequency domains respectively.



(a) Wavelet ψ_{2a} (b) Scaling function ϕ_a (c) Scaling function ϕ_{2a}

Figure 1. Definition of wavelet space domain



(a) Wavelet $\bar{\psi}_{2a}$ (b) Scaling function $\bar{\phi}_a$ (c) Scaling function $\bar{\phi}_{2a}$

Figure 2. Definition of wavelet in frequency domain

This definition of a wavelet is not through a construction according to certain desired properties like the Daubechies[27] wavelets, but they are based on the same idea of representing the solution at different scales with more or less detailed information. A theorem showing that the reconstruction of the total solution with this definition is accurate enough for problems of engineering interest can be found in Reference 28.

With this wavelet the complementary projection operator Q_{2a} is defined to be

$$Q_{2a}u(x) = \sum_{j=1}^{NNP} u(x_j)\bar{\psi}_{2a}(x; x - x_j)\Delta x_j, \tag{74}$$

so that the projected solution $P_a u(x)$ at the scale a can be represented by the sum of its low-scale and complementary high-scale projections:

$$P_a u(x) = P_{2a}u(x) + Q_{2a}u(x). \tag{75}$$

The solutions $P_{2a}u(x)$ at the scale $2a$ could also be decomposed into its components $P_{4a}u(x)$ and $Q_{4a}u(x)$ and so on. This framework provides the necessary means to decompose the solution into as many scales as necessary.

Particles with high values of the reconstructed high-scale solution are marked as high-gradient nodes for the adaptive algorithm.

5.2. *The physical interpretation of multiresolution analysis and its application to hp-like adaptivity*

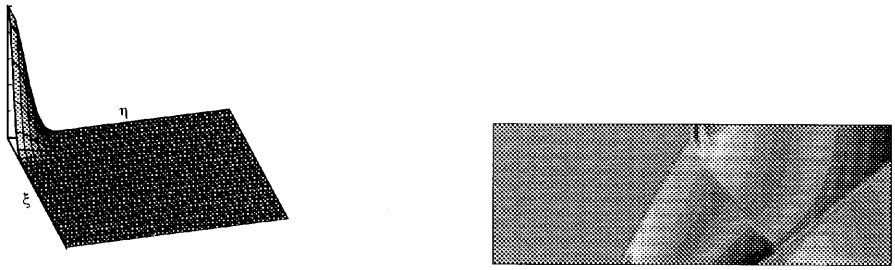
As a simple illustration, the two-level decomposition in 2D, specified by the product of the one-dimensional window functions in directions x and y, is given as

$$\phi_0(x, y) = \phi_2(x)\phi_2(y) + \phi_2(x)\psi_2(y) + \psi_2(x)\phi_2(y) + \psi_2(x)\psi_2(y), \tag{76}$$

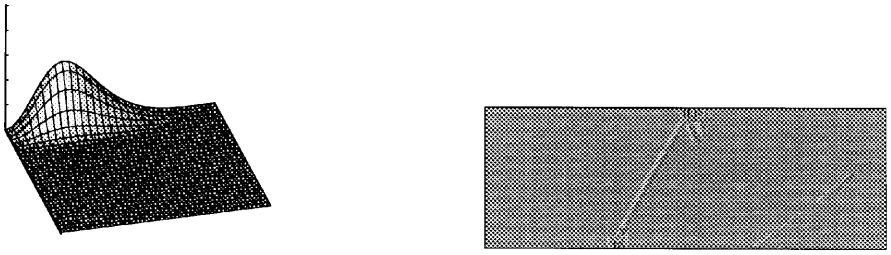
where $\phi_0(x, y) = \phi_0(x)\phi_0(y)$, $\phi_0(x) = \phi_2(x) + \psi_2(x)$ and $\phi_0(y) = \phi_2(y) + \psi_2(y)$. In the Fourier domain the 2×2 decomposition (via the product rule) is given as

$$\begin{aligned}\hat{\phi}_0(\xi, \eta) = \hat{\phi}_0(\xi)\hat{\phi}_0(\eta) &= [\hat{\phi}_2(\xi) + \hat{\psi}_2(\xi)][\hat{\phi}_2(\eta) + \hat{\psi}_2(\eta)] \\ &= \hat{\phi}_2(\xi)\hat{\phi}_2(\eta) + \hat{\phi}_2(\xi)\hat{\psi}_2(\eta) + \hat{\psi}_2(\xi)\hat{\phi}_2(\eta) + \hat{\psi}_2(\xi)\hat{\psi}_2(\eta).\end{aligned} \tag{77}$$

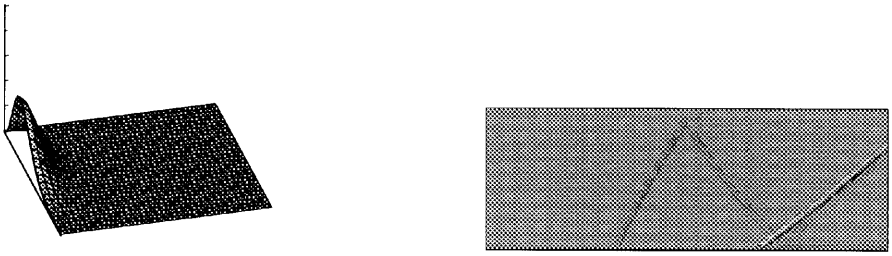
The frequency (or wave number) spectra of the four resulting windows are depicted on the left-hand side of Figure 3. The 2×2 decomposition of the multiple-scale RKPM solution of the flow past a biconvex aerofoil, described in Section 6.1, is depicted on the right-hand side of Figure 3. As can be seen in the figure, the low-scale (scaling function) component $\phi_2(x)\phi_2(y)$ filters out the oscillation; therefore it gives a very smooth solution around the shock front. The products of a scaling function and a wavelet, $\phi_2(x)\psi_2(y)$ and $\psi_2(x)\phi_2(y)$, show a mixture of high-wave-number response and



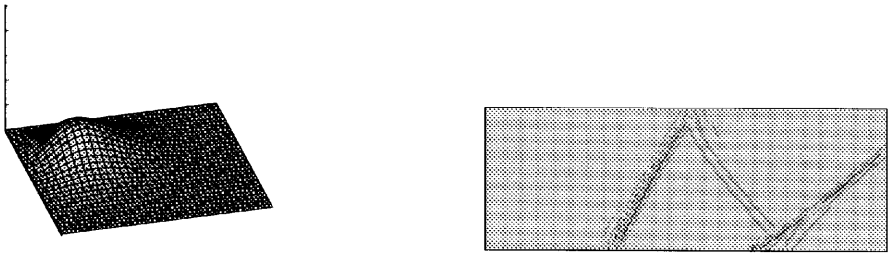
(a) $\hat{\phi}_2(\xi)\ \hat{\phi}_2(\eta)$



(b) $\hat{\phi}_2(\eta)\ \hat{\psi}_2(\xi)$



(c) $\hat{\phi}_2(\xi)\ \hat{\psi}_2(\eta)$



(d) $\hat{\psi}_2(\xi)\ \hat{\psi}_2(\eta)$

Figure 3. 2 × 2 decomposition of SUPG/RKPM

aliasing. However, these medium scales display mostly the high-wave-number approximation of the shock front, whereas the high-scale component (product of wavelets) $\psi_2(x)\psi_2(y)$ displays mostly the aliasing wave number that cannot be resolved with the current resolution, which is a typical example of the Gibbs phenomenon. Nevertheless, the high wavelet component $\psi_2(x)\psi_2(y)$ picks up the location of the high-gradient region. We are currently investigating the optimal level of multiple-scale decomposition in defining adaptivity.

It is also clearly illustrated in the wavelet component $\psi_2(x)\psi_2(y)$ that the magnitude of the aliasing solution and the width of the shock are much smaller. This example shows that the integral window transform process (i.e. multiresolution analysis) can zoom in to pick up the high gradient of the response and zoom out if no magnification of the response is necessary. The zoom-in and zoom-out capability of the multiple-scale reproducing kernel method shows great promise in meshless unstructured multigrid or hp-like adaptivity. Moreover, the physical interpretation of the computed results can further be synthesized, since each banded solution is governed by the frequency (wave number) band content of the corresponding Fourier transform. In Section 6.1 we shall employ the wavelet solution $\psi_2(x)\psi_2(y)$ as an error indicator. An h-adaptivity algorithm is also developed. The theoretical development of wavelet solution and edge detection is given in Reference 15.

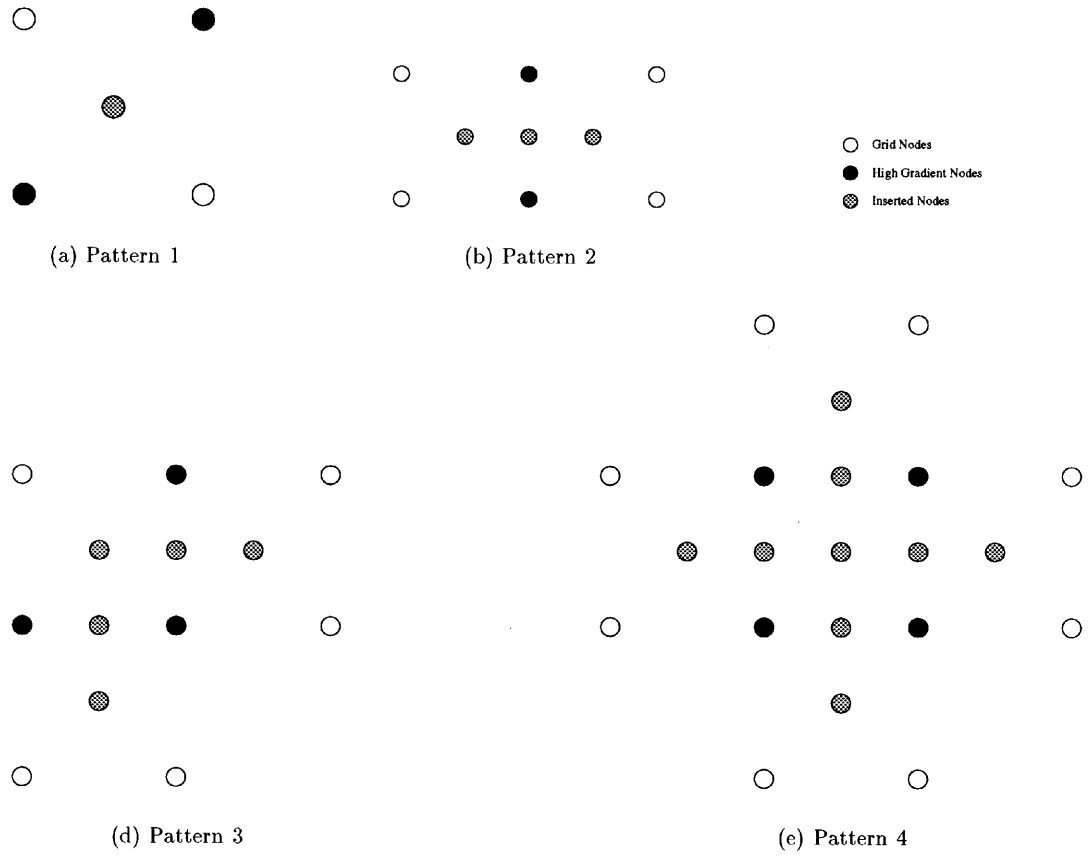


Figure 4. Node patterns

5.3. h-Adaptivity by particle method

After the high-gradient regions (i.e. wavelet solution $\psi_2(x)\psi_2(y)$) have been found, a strategy is necessary to systematically insert new nodes. In this example, only two basic patterns are used to build the various combinations that are needed to cover all cases.

The method employed here uses cells to keep track of what happens between four neighbouring particles by storing their connectivities. The algorithm checks every cell to see whether it has more than one high-gradient node. If this is the case, the cell gets an additional middle node. Then the algorithm checks all edges to see whether they have two high-gradient nodes. In this case a new node is inserted in the middle of the corresponding edge. The node patterns and algorithms are shown in Figures 4 and 5 respectively.

After all necessary new nodes have been inserted, the odd collocations are destroyed and four new cells with the appropriate collocations are generated. Because the area within single cells has changed for certain cells, the new weighting for each cell and node is computed and the corrected dilation parameters are set for all nodes.

This algorithm is only one of many possibilities that can be chosen. The advantage here is the easy use and fast refinement, because extensive computations are not necessary to find the areas that need additional nodes. A problem is the generation of cells with less than four nodes, causing difficulties for the subsequent refinement. This is illustrated in Figure 6.

Right now this problem is avoided by not refining cells that have less than four nodes. Because cells with only two or three nodes are generated only if the parent cell did not have four high-gradient points, these cells do not contain the highest gradients and therefore not refining them does not cause a significant error. The results in the example of the thin biconvex aerofoil show that this algorithm works well enough to improve the solution considerably compared with the solution that can be obtained with the original node distribution.

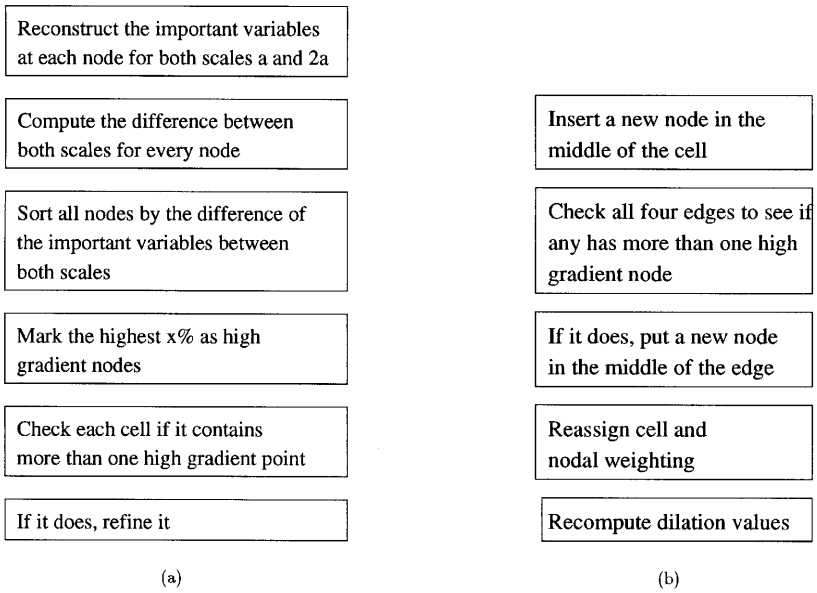


Figure 5. Algorithm for refinement

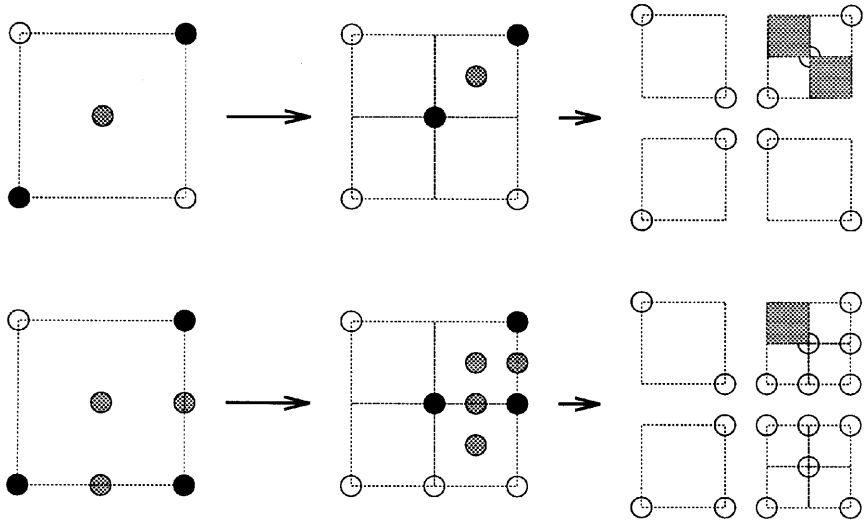


Figure 6. Generation of one-node cells

6. NUMERICAL EXAMPLES

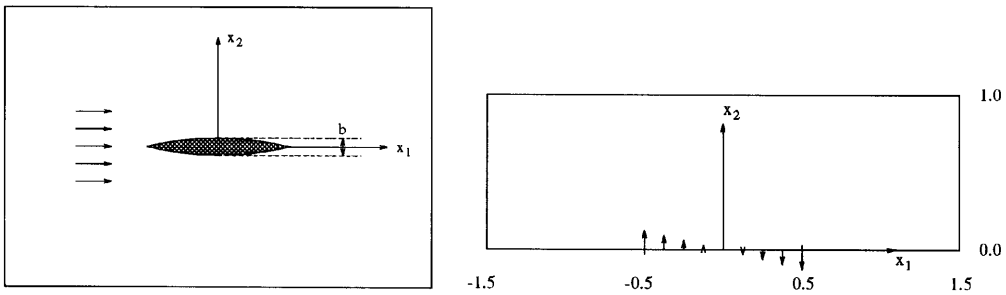
6.1. Biconvex Aerofoil

6.1.1. Geometry and Boundary Conditions. The problem considered here is that of a thin biconvex aerofoil in a uniform flow field.[25,29] Two symmetric parabolic arcs prescribe the geometry of the thin aerofoil. Figure 7 shows the configuration; b is the ratio of the maximum aerofoil thickness to the cord length. The subscript '∞' refers to freestream values.

Because the problem is symmetric, only the upper half of the domain, $x_2 \geqslant 0$, is considered. The parabolic arc bounding the upper half of the aerofoil is described by

$$x_2 = \tfrac{1}{2}b[1 - (2x_1)^2]. \tag{78}$$

The governing equations for this problem are the Euler equations and the freestream values are assumed to be $\rho_\infty = 1{\cdot}0$, $u_{2\infty} = 0{\cdot}0$ and $e_\infty = 1{\cdot}0$.



(a) Problem Statement (b) Computational Domain

Figure 7. Thin aerofoil

The value of $u_{1\infty}$ depends on the freestream Mach number M_∞ according to the formula

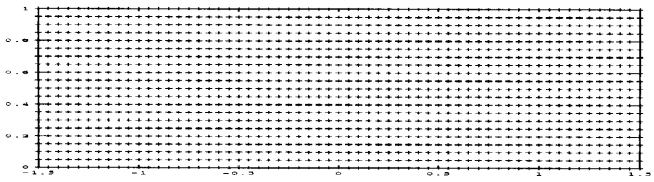
$$u_{1\infty} = \pm\sqrt{\left(\frac{M_\infty^2\gamma(\gamma-1)e_\infty}{\frac{1}{2}M_\infty^2\gamma(\gamma-1)+1}\right)}. \tag{79}$$

Along the symmetry axis but not on the aerofoil the following condition is imposed:

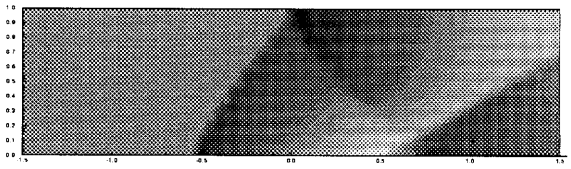
$$u_2 = 0, \quad x^2 = 0, \quad |x_1| > 0.5. \tag{80}$$

On the surface of the aerofoil the velocity vector must be tangential to the surface. This restriction can be expressed by

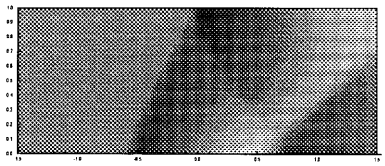
$$u_2/u_1 = \mathrm{d}x_2/\mathrm{d}x_1. \tag{81}$$



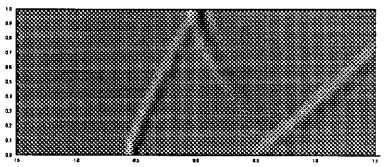
Original Particle Distribution



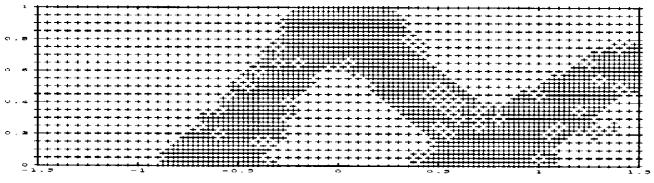
Solution at $t = 8.0$



Low Scale Solution



High Scale Solution



First Refined Particle Distribution

Figure 8. First refinement

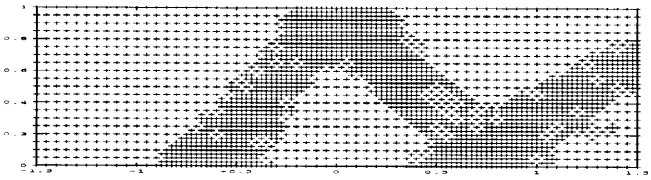
Because of the assumption that the aerofoil is thin and therefore perturbs the uniform flow field only slightly, the value of u_1 is approximated by its freestream value $u_{1\infty}$. Equation (81) then becomes

$$u_2/u_{1\infty} = \mathrm{d}x_2/\mathrm{d}x_1 \tag{82}$$

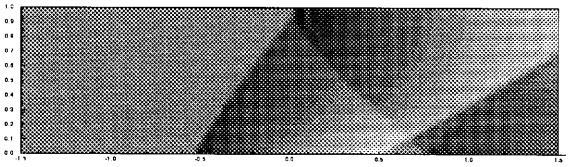
and the boundary condition on the surface of the aerofoil is prescribed by

$$u_2 = -4bx_1u_{1\infty}, \quad |x_1| \leqslant 0{\cdot}5. \tag{83}$$

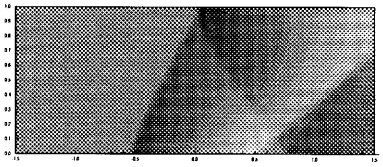
On the inflow boundary the values $\rho = \rho_\infty, u_1 = u_{1\infty}, u_2 = u_{2\infty}$ and $e = e_\infty$ are prescribed; at the upper boundary, only $u_2 = u_{2\infty}$ is prescribed; at the outflow boundary no values are prescribed.



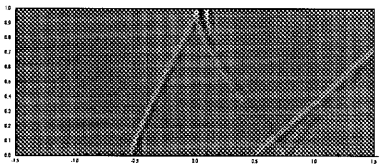
First Refined Particle Distribution



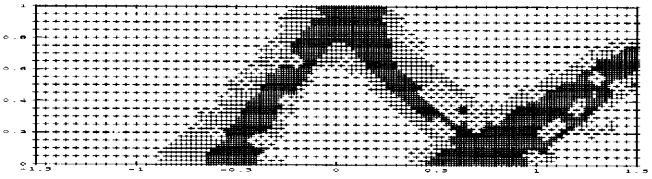
Solution at $t = 8.0$



Low Scale Solution



High Scale Solution



Second Refined Particle Distribution

Figure 9. Second refinement

6.1.2. Results. The solution in which we are interested is the pressure distribution over the domain. It is important to find the pressure shock fronts and refine the discretization around them so that numerical noise can be avoided as much as possible. Figure 8 shows the procedure starting from the original particle distribution. The solution for $t = 8{\cdot}0$ is decomposed into its high- and low-scale parts. Then nodes are inserted according to the high-scale solution part.

With this new discretization the solution is computed again (see Figure 9). It is easy to see the improvement compared with the previous solution. The decomposition shows that the high-scale solution area became smaller. The information of the high-scale solution is used for another refinement of the particle distribution.

6.2. 2D Advection–Diffusion Equation

The 2D steady state advection–diffusion equation is solved by the RKPM. Adaptivity based on multiresolution analysis is also conducted.

The governing equation of the advection–diffusion problem is

$$v\nabla^2\phi - \mathbf{u} \cdot \nabla\phi = 0. \tag{84}$$

The problem description is given in Figure 10, where the computation domain is defined. The flow is unidirectional, constant ($\|\mathbf{u}\| = 1$) and skew to the particle distribution. The diffusivity v is 10^{-6}. As shown in Figure 10, the inflow boundary condition is discontinuous and the natural boundary condition is applied to the outflow boundary. In implementation of the computation a cubic spline is used as the window function and for simplicity, the dilation parameter a is chosen to be unity in all cases. Four kinds of uniform particle distributions, 11×11, 21×21, 41×41 and 61×61, are utilized. In addition, four levels of adaptivity, starting from the base grid 11×11, are performed.

The numerical results of the four uniform spacing grids (121, 441, 1681 and 3721 nodes) are shown in Figure 11. As we could expect, more nodes are needed in the discontinuous region than in the smooth region. However, a uniform particle distribution cannot meet this requirement, because refinements are made over the entire region, in which most of the inserted nodes are useless. The algorithm of adaptive refinement is depicted in Figure 12. According to the wavelet solution, high-gradient points are detected, which are labelled by a different symbol in the figure. New nodes are simply added into this region to get higher resolution. Because of the meshless property, adaptivity is easy to implement even for large-motion flow problems, which often pose difficulties when the translational finite element method is used. As illustrated in Figure 13, the results of the adaptive

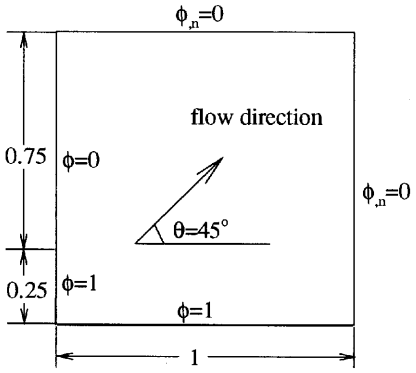


Figure 10. Problem statement: advection skew to mesh

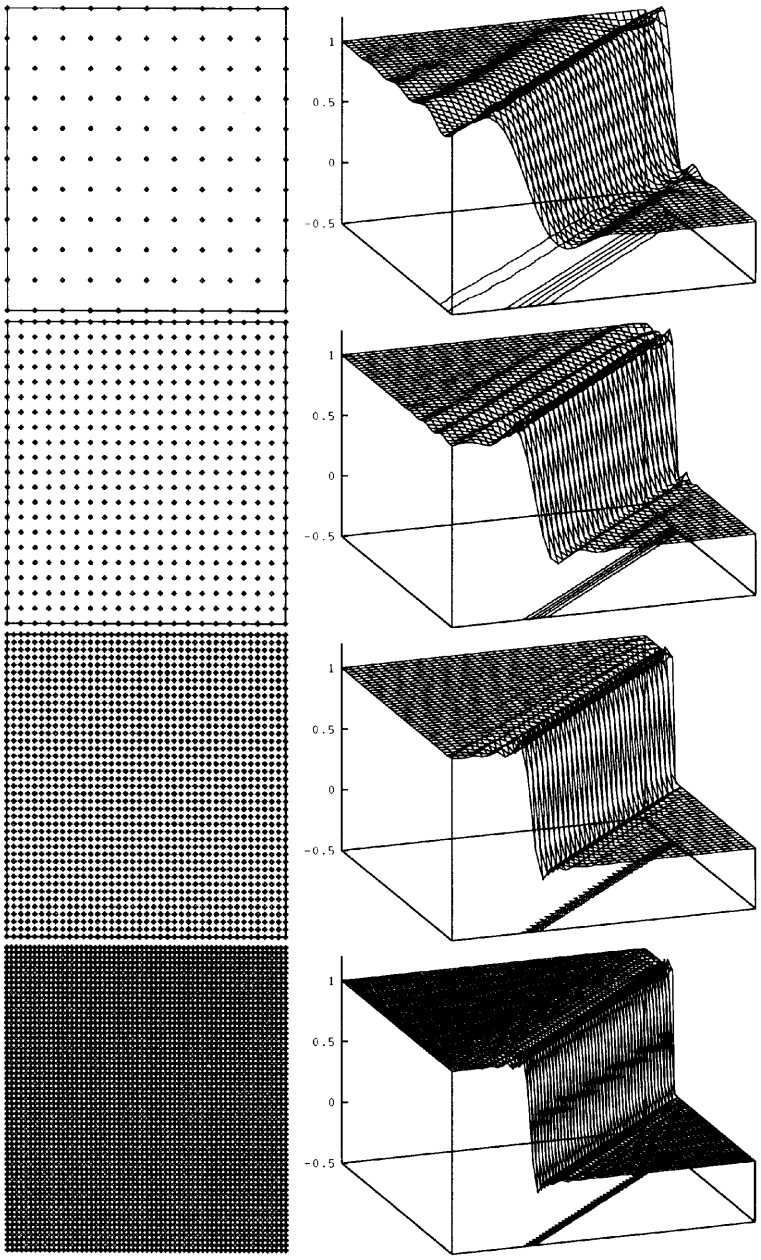


Figure 11. Uniform refinements (11 × 11, 21 × 21, 41 × 41 and 61 × 61)

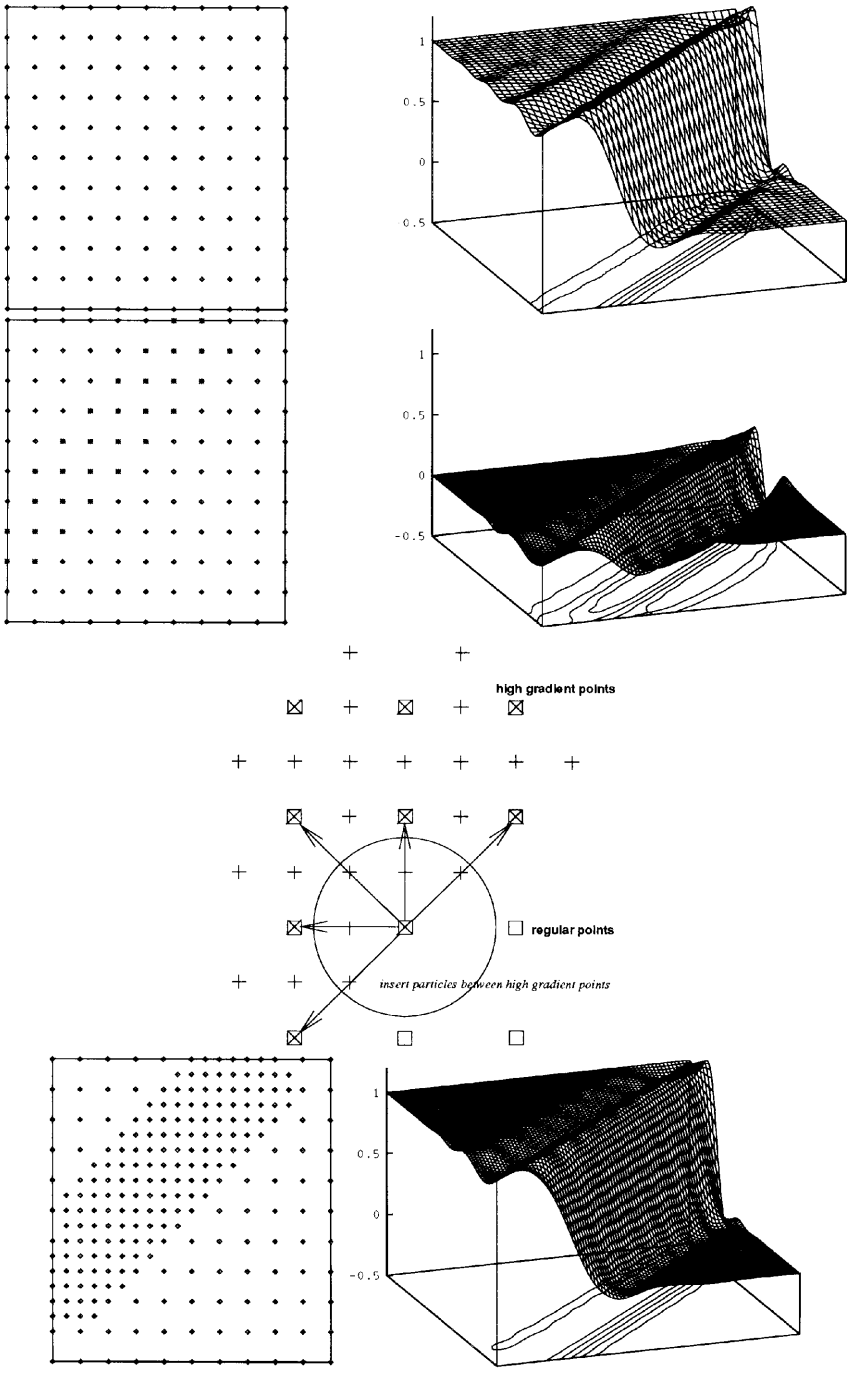


Figure 12. Adaptivity algorithm by multiresolutin RKPM

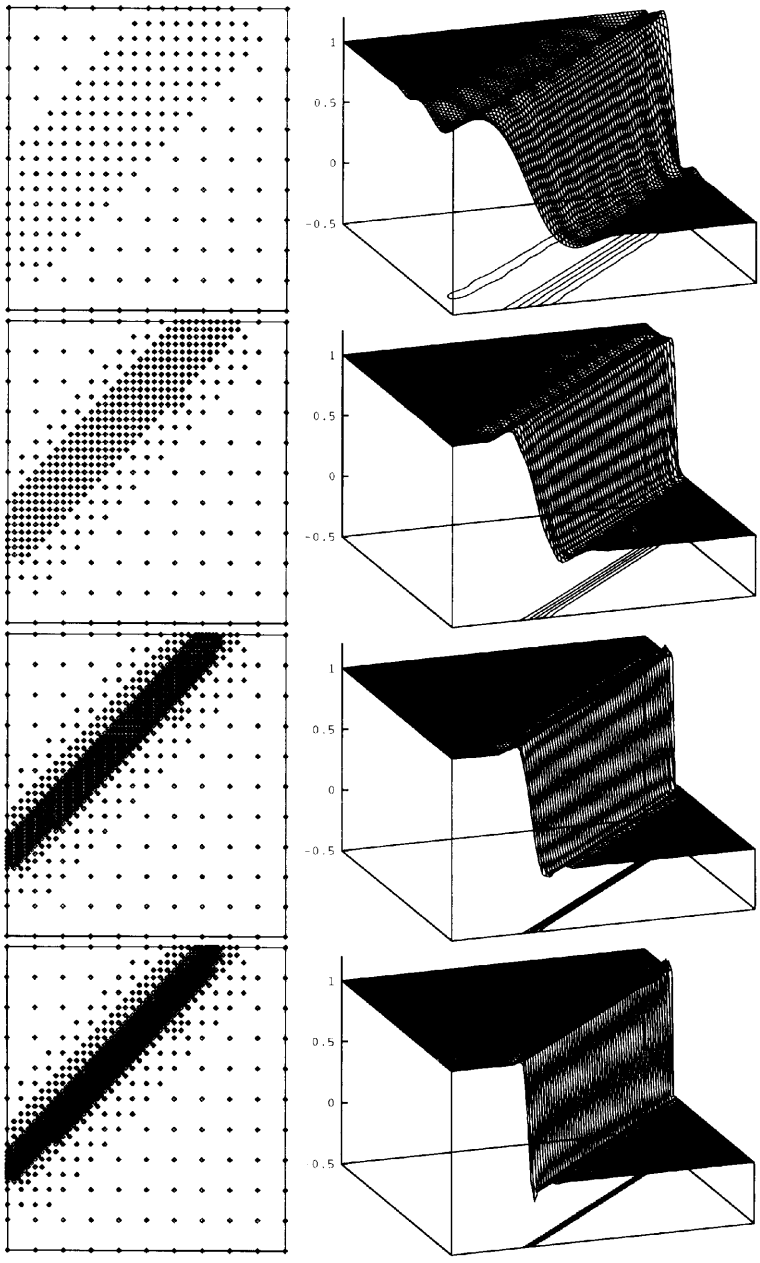


Figure 13. Adaptive refinements (238, 474, 1013 and 2179 particles)

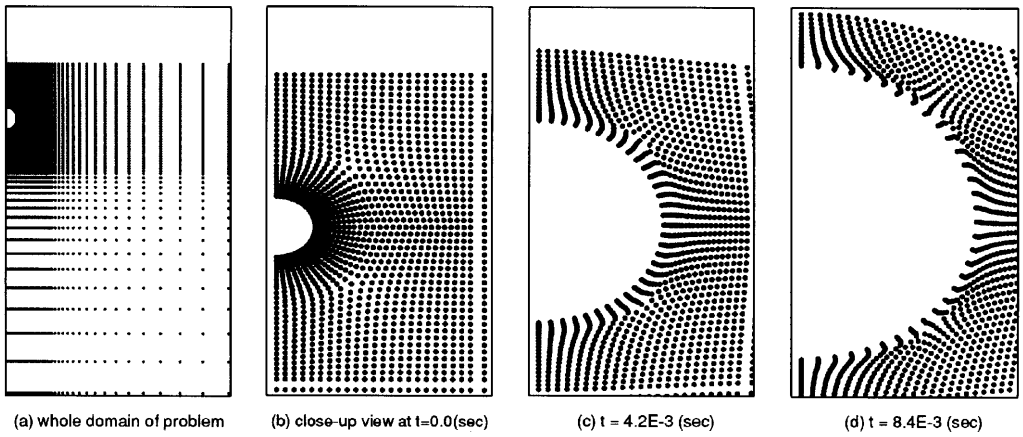


Figure 14. Bubble evolution

refinements, which only add nodes to the high-gradient region, give a more accurate prediction of the shock front with fewer nodes.

6.3. Large deformation by Lagrangian RKPM

In this subsection an example of large deformation of fluids using the Lagrangian RKPM is illustrated. An axisymmetric formulation is employed for the simulation of bubble explosion near the free surface. A uniform internal pressure is applied outwards to the bubble. Figure 14(a) shows the domain of the problem, which is $[0{\cdot}0, 5{\cdot}0] \times [-5{\cdot}0, 1{\cdot}0]$ (m). The total number of particles is 2655. Explicit time integration with a time step of $2{\cdot}0 \times 10^{-7}$ s is used and no numerical damping is added. Close-up views of selected steps are given in Figures 14(b)–(d). More examples of large deformation with different materials will be presented in a future paper.

7. CONCLUSIONS

It is demonstrated that a major advantage of the multiresolution reproducing kernel particle method (RKPM) over traditional adaptive methods lies in the higher accuracy and the built-in property of constructing hp-like adaptive refinement without a mesh. In the procedure of adaptive refinement, nodes are simply inserted into selected regions of high gradient, which can be obtained from the wavelet solution part of the total solution. The performances of the proposed algorithms are studied by solving the Euler equation, the 2D advection–diffusion equation and a large-deformation problem. Applications of the RKPM to more complicated problems in computational fluid dynamics are under way.

ACKNOWLEDGEMENTS

The support of this research provided by the Air Force Office of Scientific Research (AFOSR) to Northwestern University is gratefully acknowledged. We also acknowledge the valuable comments offered by Jeffrey Gosz.

REFERENCES

1. L. Lucy, 'A numerical approach to testing the fission hypothesis', *J. Fluids*, **82**, 1013–1024 (1977).
2. J. J. Monaghan, 'Why particle methods work', *SIAM J. Sci. Stat. Comput.*, **3**, 422–433 (1982).
3. J. J. Monaghan, 'An introduction to smooth particle hydrodynamics', *Comput. Phys. Commun.*, **48**, 89–96 (1988).
4. J. J. Monaghan and R. A. Gingold, 'Shock simulation by the particle method SPH', *J. Comput. Phys.*, **52**, 374–389 (1983).
5. S. W. Attaway, M. W. Heinstein, F. J. Mello and J. W. Swegle, 'Coupling of smooth particle hydrodynamics with PRONTO', preprint, 1993.
6. G. R. Johnson, E. H. Peterson and R. A. Stryrk, 'Incorporation of an SPH option into the EPIC code for a wide range of high velocity impact computations', preprint, 1993.
7. L. Libersky and A. G. Petschek, 'Smooth particle hydrodynamics with strength of materials', *Proc. Free-Lagrange Conf.*, Moran, WY, June 1990.
8. W. K. Liu *et al.*, 'Overview and applications of the reproducing kernel particle methods', *Arch. Comput. Methods Eng.; State of the Art Rev.*, **3**, 3–80 (1996).
9. W. K. Liu, S. Jun, S. Li, J. Adee and T. Belytschko, 'Reproducing kernel particle methods for structural dynamics', *Int. j. numer. methods eng.*, **38**, 1655–1679 (1995).
10. W. K. Liu, S. Jun and Y. F. Zhang, 'Reproducing kernel particle methods', *Int. j. numer. methods fluids*, **20**, 1081–1106 (1995).
11. W. K. Liu, J. Adee and S. Jun, 'Reproducing kernel and wavelet particle methods for elastic and plastic problems', in D. J. Benson and R. A. Asaro (eds), *Advanced Computational Methods for Material Modeling*, AMD Vol. 180/PVP Vo.. 268, ASME, New York, 1993, pp. 175–190.
12. W. K. Liu, C. T. Chang, Y. Chen and R. A. Uras, 'Multiresolution reproducing kernel particle metnods in acoustic problems', in *Acoustics, Vibrations, and Rotating Machines*, DE Vol. 84-2, Part B, pp. 881–900.
13. H. M. Shodja, T. Mura and W. K. Liu, 'Multiresolution analysis of a micromechanical model', in S. Ghosh and M. Ostoja-Starzewski (eds), *Computational Methods in Micromechanics*, AMD Vol. 212/MD Vol. 62, ASME, New York, 1995, pp. 33–54.
14. W. K. Liu, S. Li and T. Belytschko, 'Moving least square kernel Galerkin method (i) methodology and convergence', *Comput. Methods Appl. Mech. Eng.*, To appear.
15. W. K. Liu, Y. Chen, C. T. Chang and T. Belytschko, 'Advances in multiple scale kernel particle methods', *Comput. Mech.*, **18**, 73–111 (1997).
16. T. Belytschko, Y. Y. Lu and L. Gu, 'Element free Galerkin methods', *Int. j. numer. methods eng.*, **37**, 229–256 (1994).
17. C. A. Duarte and J. T. Oden, 'Hp clouds—a meshless method to solve boundary—value problems', *TICAM Rep. 95-05, 1995.*
18. I. Babuška and J. M. Melenk, 'The partition of unity finite element method', *University of Maryland Tech. Note BN-1185*, 1995.
19. E. Oñate, S. Idelsohn and O. C. Zienkiewicz, 'Finite point methods in computational mechanics', *Int. j. numer. methods eng.*, in press.
20. G. Yagawa, T. Yamada and Kawai, 'Some remarks on free mesh method: a kind of meshless finite element method', *Proc. ICES-95*, 1995.
21. W. K. Liu and C. Oberste-Brandenburg, 'Reproducing kernel and wavelet particle methods', in J. P. Cusumano, C. Pierre and J. T. Wu (eds), *Aerospace Structures: Nonlinear Dynamics and System Response*, AD Vol. 33, ASME, New York, 1993, pp. 39–56.
22. W. K. Liu and Y. Chen, 'Wavelet and multiple-scale reproducing kernel methods', *Int. j. numer. methods fluids*, **21**, 901–931 (1995).
23. F. Shakib, T. J. R. Hughes and Z. Johan, 'A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier–Stokes equations', *Comput. Methods Appl. Mech. Eng.*, **89**, 141–219 (1991).
24. T. J. R. Hughes, *The Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
25. T. E. Tezduyar and T. J. R. Hughes, 'Finite element formulations for convection dominated flows with particular emphasis on our compressible Euler equations', *AIAA Paper 83-0125*, 1983.
26. T. J. R. Hughes and T. E. Tezduyar, 'Finite element methods for first-order hyperbolic systms with particular emphasis on the compressible Euler equations', *Comput. Methods Appl. Mech. Eng.*, **45**, 217–284 (1984).
27. I. Daubechies, *Ten Lectures on Wavelets*, Vol. 61, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.
28. C. K. Chui, L. Montefusco and L. Puccio, *Wavelets: Theory, Algorithms and Applications*, Academic, New York, 1994.
29. G. J. Le Bean, S. E. Ray, S. K. Aliabadi and T. E. Tezduyar, 'SUPG finite element computation of compressible flows with the entropy and conservation variables formulations', *Comput. Methods Appl. Mech. Eng.*, **104**, 397–422 (1993).

