INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **46**:939–960
Published online 17 September 2004 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.753

Microfluidics simulation using adaptive unstructured grids

Jacob Waltz*,†

Laboratory for Computational Physics and Fluid Dynamics, Naval Research Laboratory, Washington, DC 20375, U.S.A.

SUMMARY

A methodology for microfluidics simulation is presented. The methodology solves the three-dimensional incompressible Navier–Stokes equations with an adaptive unstructured Finite Element method. A semi-implicit Fractional Step procedure is used for time integration. The entire methodology has been parallelized for shared-memory architectures via an algebraic domain decomposition. Results from both verification problems and prototypical applications are included. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: incompressible flows; unstructured grids; microfluidics; MEMS

1. INTRODUCTION

1.1. Motivation and background

The field of micro-electrical–mechanical systems (MEMS) is currently of widespread interest in numerous technological areas. Many MEMS devices are based on, or otherwise incorporate, some aspect of fluid dynamics [1]. Examples can be found in power technologies, medical implants, narcotics and explosives detection, environmental monitoring devices, and autonomous microvehicles. The role of MEMS devices in future engineering technologies is only expected to increase. Clearly, the ability to reliably and accurately predict the associated flow fields around such devices is an important asset. The purpose of this paper, therefore, is to present an overall computational methodology which provides such a capability for incompressible flows. This methodology is then applied to realistic three-dimensional (3D) geometries to demonstrate its effectiveness.

*Correspondence to: J. Waltz, Los Alamos National Laboratory, P.O. Box 1663, MS T086, Los Alamos, NM 87545, U.S.A.
†E-mail: jwaltz@lanl.gov

Contract/grant sponsor: National Research Council

Received 8 April 2002
Revised 4 November 2003

Although incompressible microfluidics does at first glance appear to be relatively free of computational difficulties, the issue of grid resolution quickly arises. Due to the low Reynolds numbers involved—generally 10^{-1} or less—boundary layer grids are not required. However, most problems of practical interest contain a very large variation in feature size: the largest feature may be hundreds or thousands of microns across, whereas the smallest feature may be less than 10 μm across. Numerical experience has indicated that across any feature, 5–10 grid points are generally needed to allow proper development of velocity gradients. Therefore the smallest edge will be on the order of 1 μm or less, while the largest edge will be on the order of tens of microns.

A uniform grid size of 1 μm or less is obviously quite excessive. Therefore, if the problem is to be kept at a reasonable size, the numerical method must be able to accommodate large variations in edge length. Furthermore, an adaptive method is preferred: although one could generate an appropriate mesh *a priori* to the calculation, the use of mesh adaption allows for a more precise and efficient placement of additional mesh points.

The combination of mesh adaption and non-uniform mesh size leads quite naturally to unstructured grid methods, and an adaptive unstructured Finite Element (FE) scheme forms the basis of the methodology described in this work. To offset the computational costs associated with incompressible flow solvers and unstructured grid methods, the entire procedure has been parallelized for shared memory architectures. Further improvements in computational efficiency are achieved by integrating the viscous terms implicitly. Although this approach is hardly revolutionary, the impact on microfluidics is particularly dramatic due to the overwhelming dominance of viscous effects.

1.2. Validity of continuum approximation

The incompressible Navier–Stokes equations are used as the governing model for the flows considered in this work. One advantage of this approach is that an equation of state is not required. Therefore, the same mathematical model can be used for a wide range of both liquid and gaseous flows. An important question is whether or not the Navier–Stokes equations are still valid for the problems under consideration.

For gases, this question is easily answered via the Knudsen number, defined as $Kn = \bar{\lambda}/L$, where $\bar{\lambda}$ is the mean free path of the flow and L is the size of the smallest feature. Generally, $Kn = 0.01$ is accepted as the upper limit of the full continuum approximation, with extension into the range $0.01 \leqslant Kn \leqslant 0.1$ possible if modified boundary conditions are used [2]. For air at standard temperature and pressure, $\bar{\lambda} \approx 6 \times 10^{-6}$ cm, which yields a lower limit on the feature size of $L \approx 6$ μm.

For liquid flows, a typical Leonnard–Jones fluid has a model potential of the form

$$U(r) = k\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{1}$$

where r is the intermolecular distance, and k and σ are constants. An infinite system—and hence bulk properties—can be successfully approximated with particle-based methods and periodic boundary conditions if the size of the computational domain is on the order of 6σ in each direction [3]. For water, $\sigma \approx 3.5$ Å and hence the computational domain must be ≈ 21 Å, or 0.0021 μm, in each direction. A minimum feature size of 6 μm corresponds to $\approx 2800\sigma$,

which is equivalent to several hundred 'computational boxes' of size 6σ. The continuum approximation is clearly reasonable in this case.

1.3. Summary of paper

This paper builds on previous work [4] to include additional details of the numerical method, the mesh adaption procedure, and more complex applications which better represent real engineering-type problems. The emphasis of this work is not new methods but rather improvements to and combinations of existing methods which facilitate accurate and efficient numerical simulation of the problems of interest.

The remainder is organized as follows. Spatial and temporal discretization are discussed in Sections 2 and 3, respectively. Section 4 presents the mesh adaption procedure, and Section 5 discusses parallelization. A verification problem is presented in Section 6, while Section 7 presents sample applications. Lastly, concluding remarks are given in Section 8.

2. SPATIAL DISCRETIZATION

2.1. General finite element formulation

The FE approach used in this work is an edge-based Galerkin method. Edge-based approaches date back to the early 1990s for compressible flows [5–8], with more recent work exploring edge-based schemes in the context of incompressible flow solvers [9, 10]. The underlying idea in all such approaches is to store geometric parameters at the edge level rather than the element level. In 3D, this storage approach results in a substantial savings in indirect addressing and hence overall computational cost [11]. Furthermore, the edge-based approach forms a natural framework for upwind schemes on unstructured grids [11–13]. The edge-based scheme presented here is very similar to those previously described but has the property that it does not assume linear basis functions. The derivation is included below for completeness.

Given a generic field variable $u(x)$ and a FE approximation Ω_h of the continuous domain Ω, the gradient of $u(x)$ at the nodes of the mesh can be written in the familiar Galerkin form

$$\int_{\Omega_h} N^\alpha N^\beta (\nabla u)^\beta \, \mathrm{d}\Omega_h = \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h \tag{2}$$

where α and β represent nodal indices and N^α denotes the finite element basis function at node α. In the usual FE fashion, this expression has two implied summations: an outer summation over all elements surrounding node α, and, for each such element, and inner summation over the nodes β of the element. The equivalent matrix form is, with a lumped mass approximation,

$$M_L (\nabla u)^\alpha = \underline{G}^{\alpha\beta} u^\beta \tag{3}$$

The lumped mass approximation will be used throughout.

What is desired is an antisymmetric edge-based representation of the right-hand side (RHS) of (2). As a first step in the procedure, the inner summation can be split into the terms $\alpha = \beta$ and $\alpha \neq \beta$. With the summations written out explicitly (and no implied summation convention),

the result is

$$\sum_{\Omega_h \in \alpha} \sum_{\beta \in \Omega_h} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h = \sum_{\Omega_h \in \alpha} \int_{\Omega_h} N^\alpha \nabla N^\alpha u^\alpha \, \mathrm{d}\Omega_h + (1 - \delta^{\alpha\beta}) \sum_{\Omega_h \in \alpha} \sum_{\beta \in \Omega_h} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h \tag{4}$$

where $\Omega_h \in \alpha$ denotes the set of elements which surround node α, and, for each such element, $\beta \in \Omega_h$ denotes the set of nodes which form element Ω_h.

The integral in the above expression can be split in half, with the second half integrated by parts, to yield the following identity:

$$\begin{aligned} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h &= \frac{1}{2} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h + \frac{1}{2} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h \\ &= \frac{1}{2} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h \\ &\quad - \frac{1}{2} \int_{\Omega_h} \nabla N^\alpha N^\beta u^\beta \, \mathrm{d}\Omega_h + \frac{1}{2} \int_{\Gamma_h} N^\alpha N^\beta u^\beta \underline{n} \, \mathrm{d}\Gamma_h \\ &= \frac{1}{2} \int_{\Omega_h} (N^\alpha \nabla N^\beta - \nabla N^\alpha N^\beta) u^\beta \, \mathrm{d}\Omega_h + \frac{1}{2} \int_{\Gamma_h} N^\alpha N^\beta u^\beta \underline{n} \, \mathrm{d}\Gamma_h \end{aligned} \tag{5}$$

For $\alpha = \beta$, the first term is identically zero so that

$$\int_{\Omega_h} N^\alpha \nabla N^\alpha u^\alpha \, \mathrm{d}\Omega_h = \frac{1}{2} \int_{\Gamma_h} N^\alpha N^\alpha u^\alpha \underline{n} \, \mathrm{d}\Gamma_h \tag{6}$$

Substitution of (5) and (6) into (4) gives

$$\begin{aligned} \sum_{\Omega_h \in \alpha} \sum_{\beta \in \Omega_h} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\beta \, \mathrm{d}\Omega_h &= \frac{1}{2}(1 - \delta^{\alpha\beta}) \sum_{\Omega_h \in \alpha} \sum_{\beta \in \Omega_h} \int_{\Omega_h} (N^\alpha \nabla N^\beta - \nabla N^\alpha N^\beta) u^\beta \, \mathrm{d}\Omega_h \\ &\quad + \frac{1}{2}(1 - \delta^{\alpha\beta}) \sum_{\Gamma_h \in \alpha} \sum_{\beta \in \Gamma_h} \int_{\Gamma_h} N^\alpha N^\beta u^\beta \underline{n} \, \mathrm{d}\Gamma_h \\ &\quad + \frac{1}{2} \sum_{\Gamma_h \in \alpha} \int_{\Gamma_h} N^\alpha N^\alpha u^\alpha \underline{n} \, \mathrm{d}\Gamma_h \end{aligned} \tag{7}$$

The above expression contains one term for the interior of the domain and two additional terms for the boundary. The interior term is antisymmetric in the elemental integral, but it is not antisymmetric in the unknown. It can be made antisymmetric in the unknown if the term

$$(1 - \delta^{\alpha\beta}) \sum_{\Omega_h \in \alpha} \sum_{\beta \in \Omega_h} \int_{\Omega_h} N^\alpha \nabla N^\beta u^\alpha \, \mathrm{d}\Omega_h \tag{8}$$

is added and subtracted from the RHS, with (5) applied to the additive part. After minor rearrangement of terms, the result is

$$\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h} N^\alpha\nabla N^\beta u^\beta\,\mathrm{d}\Omega_h = \frac{1}{2}(1-\delta^{\alpha\beta})\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h}(N^\alpha\nabla N^\beta - \nabla N^\alpha N^\beta)u^{\alpha\beta}\,\mathrm{d}\Omega_h$$
$$+\frac{1}{2}(1-\delta^{\alpha\beta})\sum_{\Gamma_h\in\alpha}\sum_{\beta\in\Gamma_h}\int_{\Gamma_h} N^\alpha N^\beta u^{\alpha\beta}\underline{n}\,\mathrm{d}\Gamma_h$$
$$+\frac{1}{2}\sum_{\Gamma_h\in\alpha}\int_{\Gamma_h} N^\alpha N^\alpha u^\alpha\underline{n}\,\mathrm{d}\Gamma_h$$
$$-(1-\delta^{\alpha\beta})\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h} N^\alpha\nabla N^\beta u^\alpha\,\mathrm{d}\Omega_h \tag{9}$$

where $u^{\alpha\beta}=u^\alpha+u^\beta$.

While the first term in (9) is now antisymmetric, a new domain term has been introduced which is not antisymmetric. However, since the basis functions must be able to represent a constant field, they must satisfy

$$\sum_{\beta\in\Omega_h}\nabla N^\beta = 0 \Longrightarrow (1-\delta^{\alpha\beta})\sum_{\beta\in\Omega_h}\nabla N^\beta = -\nabla N^\alpha \tag{10}$$

With this relationship and (6), the last term in (9) can be rewritten as

$$(1-\delta^{\alpha\beta})\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h} N^\alpha\nabla N^\beta u^\alpha\,\mathrm{d}\Omega_h = -\sum_{\Omega_h\in\alpha}\int_{\Omega_h} N^\alpha\nabla N^\alpha u^\alpha\,\mathrm{d}\Omega_h$$
$$= \frac{1}{2}\sum_{\Gamma_h\in\alpha}\int_{\Gamma_h} N^\alpha N^\alpha u^\alpha\underline{n}\,\mathrm{d}\Gamma_h \tag{11}$$

Expression (9) therefore becomes

$$\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h} N^\alpha\nabla N^\beta u^\beta\,\mathrm{d}\Omega_h = \frac{1}{2}(1-\delta^{\alpha\beta})\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h}(N^\alpha\nabla N^\beta - \nabla N^\alpha N^\beta)u^{\alpha\beta}\,\mathrm{d}\Omega_h$$
$$+\frac{1}{2}(1-\delta^{\alpha\beta})\sum_{\Gamma_h\in\alpha}\sum_{\beta\in\Gamma_h}\int_{\Gamma_h} N^\alpha N^\beta u^{\alpha\beta}\underline{n}\,\mathrm{d}\Gamma_h$$
$$+\sum_{\Gamma_h\in\alpha}\int_{\Gamma_h} N^\alpha N^\alpha u^\alpha\underline{n}\,\mathrm{d}\Gamma_h \tag{12}$$

This final form contains a single domain term which is antisymmetric—and hence naturally conservative—upon interchange of α and β.

The dual summations in the above expression consists of an outer sum over all elements surrounding node α, and, for each such element, an inner sum for all points $\beta\neq\alpha$. This dual summation can be reinterpreted as an outer sum over all pairs of points (i.e. edges) for which

α is part of the pair, and, for each such pair, an inner sum over elements which contain the pair. Mathematically, this reinterpretation implies that

$$(1-\delta^{\alpha\beta})\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}[\cdots]\equiv\sum_{\alpha\beta\in\alpha}\sum_{\Omega_h\in\alpha\beta}[\cdots] \tag{13}$$

where $\alpha\beta\in\alpha$ denotes the set of edges which surround node α, and, for each such edge, $\Omega_h\in\alpha\beta$ denotes the elements which contain the edge. That the two forms of summation are equivalent can be seen from direct inspection.

This reinterpretation, along with the observation that the integrand does not depend on the unknowns u, allows (12) to be rewritten as

$$\begin{aligned}\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h}N^{\alpha}\nabla N^{\beta}u^{\beta}\,\mathrm{d}\Omega_h=&\sum_{\alpha\beta\in\alpha}\left[\frac{1}{2}\sum_{\Omega_h\in\alpha\beta}\int_{\Omega_h}\left(N^{\alpha}\nabla N^{\beta}-\nabla N^{\alpha}N^{\beta}\right)\mathrm{d}\Omega_h\right]u^{\alpha\beta}\\&+\sum_{\alpha\beta\in\alpha}\left[\frac{1}{2}\sum_{\Gamma_h\in\alpha\beta}\int_{\Gamma_h}N^{\alpha}N^{\beta}\underline{n}\,\mathrm{d}\Gamma_h\right]u^{\alpha\beta}\\&+\sum_{\alpha\beta\in\alpha}\left[\sum_{\Gamma_h}\in\alpha\beta\int_{\Gamma_h}N^{\alpha}N^{\alpha}\underline{n}\,\mathrm{d}\Gamma_h\right]u^{\alpha}\end{aligned} \tag{14}$$

The following vector quantities can be defined to simplify the notation:

$$\underline{D}^{\alpha\beta}=\frac{1}{2}\sum_{\Omega_h\in\alpha\beta}\int_{\Omega_h}(N^{\alpha}\nabla N^{\beta}-\nabla N^{\alpha}N^{\beta})\,\mathrm{d}\Omega_h \tag{15}$$

$$\underline{B}^{\alpha\beta}=\frac{1}{2}\sum_{\Gamma_h\in\alpha\beta}\int_{\Gamma_h}N^{\alpha}N^{\beta}\underline{n}\,\mathrm{d}\Gamma_h;\quad\underline{B}^{\alpha}=\sum_{\Gamma_h\in\alpha}\int_{\Gamma_h}N^{\alpha}N^{\alpha}\underline{n}\,\mathrm{d}\Gamma_h \tag{16}$$

With these definitions, (14) becomes

$$\sum_{\Omega_h\in\alpha}\sum_{\beta\in\Omega_h}\int_{\Omega_h}N^{\alpha}\nabla N^{\beta}u^{\beta}\,\mathrm{d}\Omega_h=\sum_{\alpha\beta\in\alpha}\underline{D}^{\alpha\beta}u^{\alpha\beta}+\sum_{\alpha\beta\in\alpha}\underline{B}^{\alpha\beta}u^{\alpha\beta}+\underline{B}^{\alpha}u^{\alpha} \tag{17}$$

The equivalent matrix form with implied summations is

$$\underline{G}^{\alpha\beta}u^{\beta}=\underline{D}^{\alpha\beta}u^{\alpha\beta}+\underline{B}^{\alpha\beta}u^{\alpha\beta}+\underline{B}^{\alpha}u^{\alpha} \tag{18}$$

Note that the last two terms apply only to edges and points, respectively, on the boundary of the domain, while the first term applies to all edges in the domain. Additionally, no assumptions about the dimensionality or the order of the elements have been made.

Lastly, note that for a constant field,

$$\underline{G}^{\alpha\beta}u^{\beta}=0\Longrightarrow 2\sum_{\alpha\beta\in\alpha}\underline{D}^{\alpha\beta}+2\sum_{\alpha\beta\in\alpha}\underline{B}^{\alpha\beta}+\underline{B}^{\alpha}=0\quad\forall\alpha \tag{19}$$

This relation constitutes the geometric conservation condition for any node α in the domain.

Notation (18) will be implied wherever the gradient operator is used. Similar expressions can be written for the divergence, curl, and tensor product of a vector field variable $\underline{u}(x)$:

$$\underline{G}^{\alpha\beta} \cdot \underline{u}^{\beta} = \underline{D}^{\alpha\beta} \cdot \underline{u}^{\alpha\beta} + \underline{B}^{\alpha\beta} \cdot \underline{u}^{\alpha\beta} + \underline{B}^{\alpha} \cdot \underline{u}^{\beta} \tag{20}$$

$$\underline{G}^{\alpha\beta} \times \underline{u}^{\beta} = \underline{D}^{\alpha\beta} \times \underline{u}^{\alpha\beta} + \underline{B}^{\alpha\beta} \times \underline{u}^{\alpha\beta} + \underline{B}^{\alpha} \times \underline{u}^{\beta} \tag{21}$$

$$\underline{G}^{\alpha\beta} \otimes \underline{u}^{\beta} = \underline{D}^{\alpha\beta} \otimes \underline{u}^{\alpha\beta} + \underline{B}^{\alpha\beta} \otimes \underline{u}^{\alpha\beta} + \underline{B}^{\alpha} \otimes \underline{u}^{\beta} \tag{22}$$

The Laplacian operator can be interpreted as a natural edge-based quantity. With a similar Galerkin procedure and the assumption of linear elements, the Laplacian of $u(\underline{x})$ at the nodes of the mesh is written as (with no implied summation)

$$M_L(\nabla^2 u)^{\alpha} = L^{\alpha\beta} u^{\beta} = C^{\alpha} u^{\alpha} + \sum_{\alpha\beta \in \alpha} C^{\alpha\beta} u^{\beta} \tag{23}$$

with

$$C^{\alpha\beta} = -\sum_{\Omega_h \in \alpha\beta} \int_{\Omega_h} \nabla N^{\alpha} \cdot \nabla N^{\beta} \, \mathrm{d}\Omega_h; \quad C^{\alpha} = -\sum_{\alpha\beta \in \alpha} C^{\alpha\beta} \tag{24}$$

where the last step follows from the requirement to represent a constant field.

In a general sense, expressions (17) and (23) represent generic Galerkin forms of the discrete gradient and Laplacian operators, respectively, and therefore are used to obtain discrete weak solutions to the specific partial differential equations of interest, i.e. the Navier–Stokes equations. Note that for both sets of operators, the finite element integrals are still performed at the element level; the only difference in the edge-based approach is the way in which the results of those integrations are stored. The number of finite element integrals performed is exactly the same, and hence no additional computational cost is incurred other than the assembly of the element-based integrals at the edge-based level. This additional cost is more than offset, however, by the reduction in indirect addressing costs.

2.2. Navier–Stokes equations

The governing equations for the flows considered in this work are the incompressible Navier–Stokes equations written in differential flux-conservative form

$$\nabla \cdot \underline{v} = 0 \tag{25}$$

$$\rho \frac{\partial \underline{v}}{\partial t} + \nabla \cdot \underline{\underline{F}} + \nabla p - \mu \nabla^2 \underline{v} = 0 \tag{26}$$

where $\underline{v}$ is the velocity vector, ρ is the density, μ the viscosity, p the pressure, and $\underline{\underline{F}} = \rho \underline{v} \otimes \underline{v}$. For simplicity, only the homogeneous case with constant μ is considered.

Discretization of the Navier–Stokes equations via a weak Galerkin procedure leads to

$$\int_{\Omega_h} N^{\alpha} \nabla N^{\beta} \cdot \underline{v}^{\beta} \, \mathrm{d}\Omega_h = 0 \tag{27}$$

$$\rho \int_{\Omega_h} N^\alpha N^\beta \Delta_t \underline{v}^\beta \, \mathrm{d}\Omega_h + \int_{\Omega_h} N^\alpha \nabla N^\beta \cdot \underline{\underline{F}}^\beta \, \mathrm{d}\Omega_h$$
$$+ \int_{\Omega_h} N^\alpha \nabla N^\beta p^\beta \, \mathrm{d}\Omega_h + \mu \int_{\Omega_h} \nabla N^\alpha \cdot \nabla N^\beta \underline{v}^\beta \, \mathrm{d}\Omega_h = 0 \tag{28}$$

where Δ_t is the discrete time derivative. The corresponding matrix form is, in the notation of the previous section,

$$\underline{G}^{\alpha\beta} \cdot \underline{v}^\beta = 0 \tag{29}$$

$$\rho M_L \Delta_t \underline{v}^\alpha + \underline{G}^{\alpha\beta} \cdot \underline{\underline{F}}^\beta + \underline{G}^{\alpha\beta} p^\beta - \mu L^{\alpha\beta} \underline{v}^\beta = 0 \tag{30}$$

2.3. Convective stabilization

A Roe-type scheme is used to stabilize the convective fluxes. Specifically, the stabilized convective fluxes on an edge are written as

$$\underline{\underline{F}}^{\alpha\beta} = \underline{\underline{F}}^\alpha + \underline{\underline{F}}^\beta - \underline{\tilde{D}}^{\alpha\beta} \otimes |\underline{v}^{\alpha\beta}| \left[\underline{v}^\beta - \underline{v}^\alpha + \frac{l}{2} \cdot (\nabla \underline{v}^\alpha + \nabla \underline{v}^\beta) \right] \tag{31}$$

where $\underline{\tilde{D}}^{\alpha\beta}$ is the unit vector along $\underline{D}^{\alpha\beta}$ and $\underline{v}^{\alpha\beta} = \underline{\tilde{D}}^{\alpha\beta} \cdot (\underline{v}^\alpha + \underline{v}^\beta)/2$. The scheme is fourth-order if the gradient terms are included, and second order if they are omitted.

3. TEMPORAL DISCRETIZATION

3.1. Fractional step scheme

The basic method for temporal discretization of the system of Equations (29)–(30) is a semi-implicit fractional step (FS) scheme, with stabilization terms of the form developed by Codina [14]. This recent work explored stabilized FS schemes in the context of a 2D FE formulation. A similar stabilization term was developed for the momentum equations. The overall technique was element-based, although alternative implementations have been examined [15].

Following the usual FS approach, the momentum equation is split through the introduction of an intermediate velocity variable $\tilde{\underline{v}}$. The splitting used in this work is the semi-implicit form

$$\rho \frac{M_L}{\delta t} (\tilde{\underline{v}}^{n+1} - \underline{v}^n) + \underline{G} p^n + \underline{G} \cdot \underline{\underline{F}}^n - \mu L \tilde{\underline{v}}^{n+\theta} = 0 \tag{32}$$

$$\rho \frac{M_L}{\delta t} (\underline{v}^{n+1} - \tilde{\underline{v}}^{n+1}) + \underline{G}(p^{n+1} - p^n) = 0 \tag{33}$$

where nodal superscripts have been dropped for clarity of presentation and δt is the time step. The implicitness parameter for the viscous term is θ, with

$$\tilde{\underline{v}}^{n+\theta} = \theta \tilde{\underline{v}}^{n+1} + (1 - \theta) \underline{v}^n \tag{34}$$

Application of the divergence operator to (33) yields the Poisson-pressure equation, which is solved as an intermediate step

$$\delta tL(p^{n+1} - p^n) - \rho \underline{G} \cdot \underline{\tilde{v}}^{n+1} = 0 \tag{35}$$

A diagonally preconditioned conjugate gradient method [16] is used to solve the linear symmetric systems (32) and (35).

The time step at a point α is taken to be

$$\delta t^{\alpha} = C_n M_L^{\alpha} \left[\sum_{\alpha\beta \in \alpha} |\underline{D}^{\alpha\beta}| \max(|v^{\alpha\beta}|, v_{\infty}) \right]^{-1} \tag{36}$$

where C_n is the Courant number and v_{∞} is some non-zero reference velocity (typically the freestream value) to prevent singularities at stagnation regions. No viscous terms are used in the time step calculation, which implies that $\theta \geqslant 0.5$ for stability.

3.2. *Pressure stabilization*

Since equal-order basis functions for the pressure and velocity spaces are used, the LBB condition is not satisfied and hence the Poisson-pressure equation (35) will be unstable. This equation can be stabilized via the approach in Reference [14]. The stabilized equation takes the form

$$\delta tL(p^{n+1} - p^n) + \tau(L p^{n+1} - \underline{G} \cdot M_L^{-1} \underline{G} p^n) - \rho \underline{G} \cdot \underline{\tilde{v}}^{n+1} = 0 \tag{37}$$

where τ is a point-wise parameter defined as

$$\tau = \frac{h^2}{4\nu + 2h|\underline{v}|} \tag{38}$$

with h the average edge length at the point in question. In this work the definition of τ is modified to

$$\tau = \frac{h}{2\max(|v|, v_{\infty})} \tag{39}$$

This modification was motivated by the fact that τ is directly analogous to the time step, and the time step definition (36) does not include viscous terms.

The removal of viscous terms from τ and δt has been found to significantly improve the stability of the Poisson-pressure equation in problems where the variation in edge size is large (e.g. a factor of 10 or more). This behaviour is not surprising given that viscous contributions to τ and δt vary with h^2, whereas the convective contributions vary linearly. The stability of the scheme when used with local time stepping also improves when the viscous contributions are omitted (the FS formulation assumes a constant time step). These two observations are based entirely on numerical experience; no formal analysis of the situation has been performed.

4. MESH ADAPTION

4.1. General considerations

The mesh adaption procedure in this work consists of an *h*-refinement scheme [17–21] for tetrahedral meshes. The basic elements of any *h*-refinement scheme are an error indicator and a set of adaption rules: the error indicator specifies *where* the mesh should be modified, and the adaption rules specify *how* the mesh should be modified. In addition, a suite of algorithms are needed to ensure that elements are refined and coarsened in accordance to the adaption rules, to add and remove elements from the mesh, and to maintain the element hierarchy through appropriate data structures. Performance also is an issue for large-scale simulations.

4.2. Error detection

Of interest for this work are error indicators based on approximation theory, which dictates that for a pth order approximation, the approximation error will be proportional to the $(p+1)$-th derivatives of the indicator variable in question. Therefore, for linear elements, the error is proportional to second-order derivatives times h^2. Several error indicators have been successfully built on second-order derivatives, or, in more general terms, the Hessian of the indicator variable [22–24]. The specific error indicator used in this work is a local 1D Hessian on an edge, normalized by the L_∞ norm

$$\varepsilon^{\alpha\beta} = \frac{|\nabla u^\beta \cdot \underline{l} - \nabla u^\alpha \cdot \underline{l}|}{\|\nabla u^\beta \cdot \underline{l} - \nabla u^\alpha \cdot \underline{l}\|_\infty} \tag{40}$$

where u is the indicator variable.

The fundamental adaption operation is to split edges with a 'high enough' error into two equal-length child edges. When the error on the child edges is 'low enough', they are merged into the original parent edge. Since the error is proportional to h^2, the error on an edge will decrease by a factor of four if the edge length is halved, and the error on an edge will increase by a factor of four if the edge length is doubled. Therefore, following the approach of Aftosmis and Berger [25], an edge is refined if its error is greater than four times the mean value, and an edge is derefined if its error is less than one fourth the mean value

$$\varepsilon^{\alpha\beta} \geqslant 4\bar{\varepsilon} \Longrightarrow \text{refine} \tag{41}$$

$$\varepsilon^{\alpha\beta} \leqslant \tfrac{1}{4}\bar{\varepsilon} \Longrightarrow \text{derefine} \tag{42}$$

where $\bar{\varepsilon}$ is the mean error value.

Quite obviously, the flows considered in this work do not contain sharp moving features such as shock waves or even vortices. Therefore, the goal in the adaption procedure is not feature tracking *per se*, but rather equidistribution of error and alleviation of preprocessing requirements. The above adaption criterion reflects this approach.

If only a single indicator variable is desired, the normalization in (40) is not strictly necessary. However, since the normalization yields an error indicator which is both dimensionless and bounded on $[0,1]$, the normalized form facilitates comparison between the errors computed with different indicator variables. In the case of N indicator variables, the final error

assigned to an edge is the maximum of all computed values for that edge, i.e.

$$\varepsilon^{\alpha\beta} = \|\varepsilon_i^{\alpha\beta}\|_\infty, \quad i = 1, N \tag{43}$$

Note that both the error indicator and the adaption criterion are free of tunable parameters.

4.3. Adaption cases

In this work, the allowable refinement cases are the standard subdivisions of a tetrahedron into two, four, or eight child elements. These are referred to as the $1:2, 1:4$, and $1:8$ cases, respectively. Subdivision of $1:2$ and $1:4$ children is not allowed; instead, the parent element is fully refined into eight children. These cases are referred to as the $2:8$ and $4:8$ cases. The resultant children are permitted to refine further in subsequent adaption passes. The intermediate case $2:4$ has been omitted for simplicity. The inverse of these cases form the allowable derefinement cases: $2:1, 4:1, 8:1, 8:2$, and $8:4$. For derefinement the $4:2$ case also is included. No restriction is made on the order in which derefinements can occur, e.g. a fully refined element is allowed to derefine into four, two, or one element as needed.

Refinement of an edge is limited to a maximum number of levels. One or two levels of refinement is generally adequate for the class of problems under consideration. Edges in the original mesh are not allowed to coarsen. An element is flagged for refinement if any of its edges are flagged for refinement. For an element to be flagged for derefinement, all of its edges, as well as all edges of its sibling elements, must be flagged for derefinement.

4.4. Implementation

The basis of the adaption data structures is a 'master' list of elements. This list contains the nodes of all elements, including refined elements. A separate list contains the 'active' elements. The active list contains all unrefined elements and therefore represents the current mesh at any stage in the simulation.

For each master element, the following information is stored:

- active element number,
- refinement case,
- number of children,
- child elements,
- refinement level,
- child number,
- parent element.

Values which do not apply for a particular element are set to zero (for example, an element in the original mesh has no child number or parent element). For the active elements, an additional pointer-like array stores the corresponding master element numbers.

The flow solver proper (along with procedures for derived data structures, I/O, etc.) only operates on the active list of elements; the master list of elements and all other adaption arrays are contained in a separate module which is accessible only by the adaption subroutines. To reduce memory usage, no information for a refined element is retained other than the adaption data structures.

4.5. Performance considerations

After each mesh change, the solution must be interpolated to newly created points with the FE basis functions; the mesh must be renumbered; derived data structures must be rebuilt; and FE integrals must be recalculated. The cost of the mesh adaption is less than the combined cost of these secondary operations. This behaviour is not surprising, since the bulk of the operations in the adaption procedures are integer. Furthermore, several of the loops in the adaption procedure operate on reduced subsets of data rather than the entire mesh. The total cost of the mesh adaption and the secondary operations is generally an order of magnitude less than the cost of a single incompressible time step. Therefore the mesh adaption procedure incurs only a minor increase in computational costs. Dynamic memory is used for all adaption arrays (as well as all flow solver arrays), and all arrays are reallocated after a mesh adaption to minimize requested but unused memory.

5. PARALLELIZATION

Despite rapid advances in computer technology over the last decade, the simulation of 3D incompressible flows on large unstructured grids remains a computationally challenging task. Therefore, a high level of parallel efficiency is needed to study many problems of practical engineering interest.

In this work, parallelization of the numerical methodology is performed with a shared-memory paradigm implemented through the OpenMP API. Although many operations can be parallelized as-is with the OpenMP directives, memory performance will suffer severely if multiple processors attempt to simultaneously update neighbouring or identical pieces of data.

The general approach used in this work to overcome the memory contention problem is a domain decomposition of the mesh at the algebraic level. For the elements, the subdomains are equivalent to colouring groups. Consider a mesh $\mathscr{G}=(\mathscr{V},\mathscr{E})$, with points $\mathscr{V}$ and elements $\mathscr{E}$, and denote by $V_{ij}=E_i\cap E_j$ the subset of points shared between elements E_i and E_j. A colouring group is defined as a subset G of $\mathscr{G}$ for which

$$V_{ij}=\emptyset \quad \forall E_i,\ E_j\in G \tag{44}$$

Since no two members of G share a point, they also cannot share a face or an edge. This property allows the mesh adaption procedure to be parallelized [26, 27], as well as the secondary operations related to derived data structures [28] and construction of the finite element matrices.

The colouring groups themselves can be generated in parallel if the elements are divided evenly among the processors and a separate set of colouring groups is constructed on each processor. Numerical experiments indicate that the number of colours typically increases by 10–20% with each doubling of the number of processors.

For the evaluation of the edge-based gradient and Laplacian operators, which constitutes the overwhelming majority of operations in any simulation, a more precise decomposition is needed to achieve a high level of parallel efficiency. Such approaches for unstructured grids on shared memory architectures were explored in Reference [29], and the technique used in this work is very similar. The basic idea is to first collect the edges into microgroups of length N_v, where N_v is a free (i.e. user-settable) parameter. The microgroups are then collected into

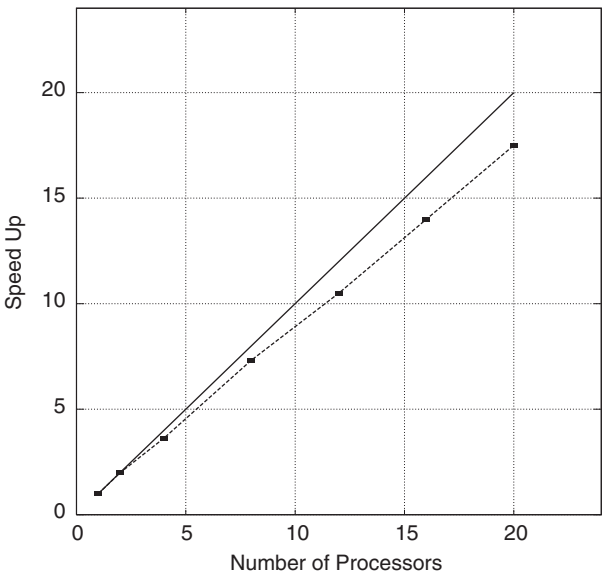

Figure 1. Parallel speed-up as a function of the number of processors.

macrogroups on each processor, and the processors operate on their respective macrogroups in parallel.

To create the micro- and macrogroups, the first step is to renumber the points via a bandwidth-minimization technique (usually a simple advancing wavefront-type procedure), and then sort the edges according to the minimum point number. The edges are sorted a second time according to the maximum point number. With this renumbering, the edges and points will be ordered such that the range of point data indirectly accessed by the edges will vary smoothly as the range of edge data is traversed [30].

Once the renumbering is completed, the microgroups are formed from sequential lists of edges, and the macrogroups—two per processor—are formed from sequential lists of microgroups. If N_{cpu} is the number of processors and N_{edge} is the number of edges, the number of edges in each macrogroup will be approximately $N_{\text{edge}}/2N_{\text{cpu}}$. The macrogroups are assigned sequentially to each processor.

At the inter-processor boundaries between macrogroups, microgroups on different processors will contain overlapping point ranges and the possibility of memory contention exists. Therefore parallelization is performed via a two-pass approach: each processor operates on its first macrogroup in the first pass, and its second in the second pass. Within each parallel loop, the range of edges under consideration by any given processor will be separated in memory by $\approx N_{\text{edge}}/2N_{\text{cpu}}$ from the range of edges under consideration by any other processor. Since the edges are renumbered according to their minimum and maximum point number, the range of point data simultaneously accessed by any two processors will not overlap (without the renumbering this would not be the case). The overall approach generally leads to a near perfect distribution of edges among processors, and hence a uniform workload.

Note that the edges *within* a microgroup do contain overlapping ranges of point data. Therefore, this technique will not perform optimally on a true vector machine. In that case,

the microgroups must not overlap. A non-overlapping ordering scheme suitable for vector machines has been implemented, and the performance of the two approaches was compared for equal values of N_v. For the computer architectures under consideration, specifically the SGI Origin class of servers and similar cache-based machines, the sequential ordering outperformed the non-overlapping ordering by 10–20% due to reduced cache misses within microgroups. The performance of the sequential ordering was found to be only mildly dependent on N_v, with $10^2 \leqslant N_v \leqslant 10^3$ generally giving the best results.

The parallel performance of the solver on a test case of 2.5×10^6 elements is shown in Figure 1. The timing data were obtained on an SGI Origin 2000 system, with a maximum of $20\,R12000$ processors. Ideal performance is indicated by the solid line. The results indicate that a parallel efficiency of $\approx 90\%$ is maintained out to $\approx 10^5$ elements per processor.

6. VERIFICATION STUDY

The flow through a circular pipe was examined as a verification study. The analytical solution in cylindrical coordinates is of the form [31]

$$u(r) = -\frac{\partial p}{\partial z}\frac{a^2 - r^2}{4\mu} \tag{45}$$

where a is the pipe radius. The initial conditions used in the simulation consisted of a uniform pressure p_0 and a radial velocity given by

$$u(r) = u_0 \frac{a^2 - r^2}{a^2} \tag{46}$$

where $u_0 = u(0)$ is the maximum velocity. For the boundary conditions, the inflow velocity profile and the outflow pressure were fixed. Since the pressure gradient is constant, the inflow pressure is given by

$$p(L) = p_0 - L\frac{\partial p}{\partial z} = p_0 + L\frac{4\mu u_0}{a^2} \tag{47}$$

where L is the length of the pipe. The physical parameters were chosen such that the Reynolds number was 10^{-2}, which is consistent with the class of problems under consideration.

The geometry was chosen such that the pipe had length equal to 20 times its radius, and the solution was calculated on a series of four meshes. The coarsest mesh contained approximately five elements across the diameter of the pipe, and the average edge length was successively halved in each finer mesh. The meshes were generated independently and were not nested.

The simulation was run to convergence on each mesh with the second-order form of the stabilized flux in (31). The velocity profile on a vertical line through the centre of the pipe was extracted and compared to the analytical solution. Table I summarizes the number of points, number of elements, the L_1 error in the velocity field, and the L_1 error in the computed inflow pressure for each calculation. The error values decrease by approximately a factor of four with each mesh refinement, which indicates the expected second-order accuracy.

The computed velocity profiles, along with the analytical velocity profile, are shown in Figure 2. Also shown are the normalized convergence histories for the three finest meshes.

Table I. Summary of mesh parameters and errors for validation study.

Mesh	Points	Elements	$\|\varepsilon_u\|/u_0$	$\|\varepsilon_p\|/p_0$
1	2194	9046	5.561	1.241×10^{-3}
2	13885	70067	1.378	0.284×10^{-3}
3	100565	549033	0.354	0.075×10^{-3}
4	777982	4416474	0.083	0.016×10^{-3}

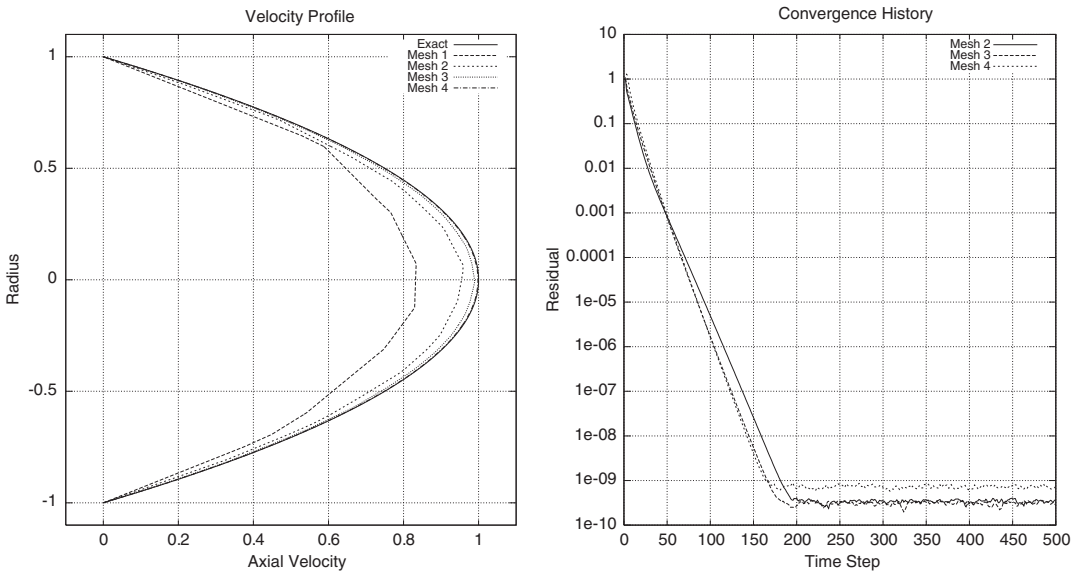

Figure 2. Velocity profiles (left) and convergence histories (right) for verification problem.

The convergence histories indicate that the convergence rate is largely independent of mesh size.

7. SAMPLE APPLICATIONS

7.1. Microsensor

The first sample problem is an air flow past a prototypical microsensor platform. The device is mounted in a square tube. Figure 3 shows a wireframe of the geometry and the initial surface mesh as viewed from the inflow plane. The initial mesh contains approximately 2.1×10^6 tetrahedron. The device has a minimum thickness of 8 μm. The boundary conditions consisted of a fixed inflow velocity profile (biparabolic) with a peak of 10 cm/s; zero velocity on all solid surfaces; and fixed outflow pressure. The Reynolds number based on the minimum thickness is 0.05. Mesh adaption was performed every 100 time steps with the velocity magnitude used as the indicator variable. Adaption was restricted to one level of refinement.

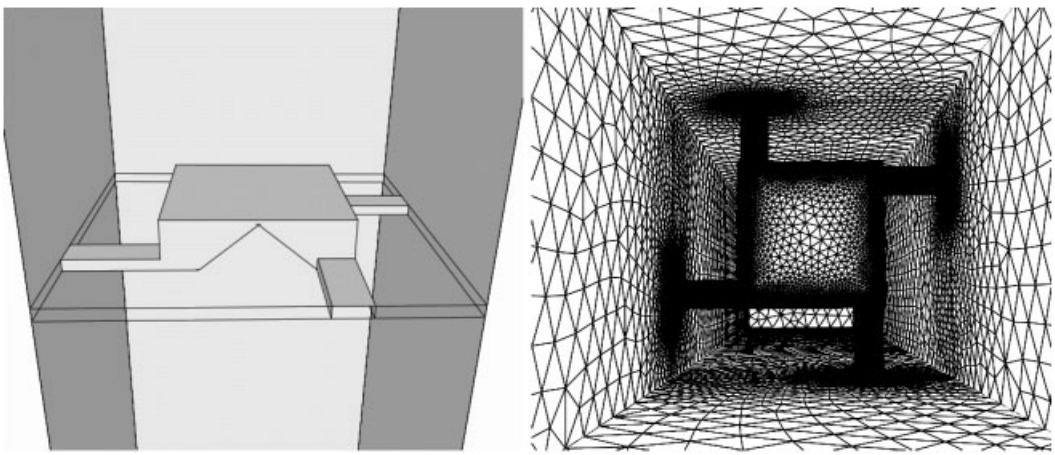

Figure 3. Geometry definition (left) and surface mesh (right) for microsensor.

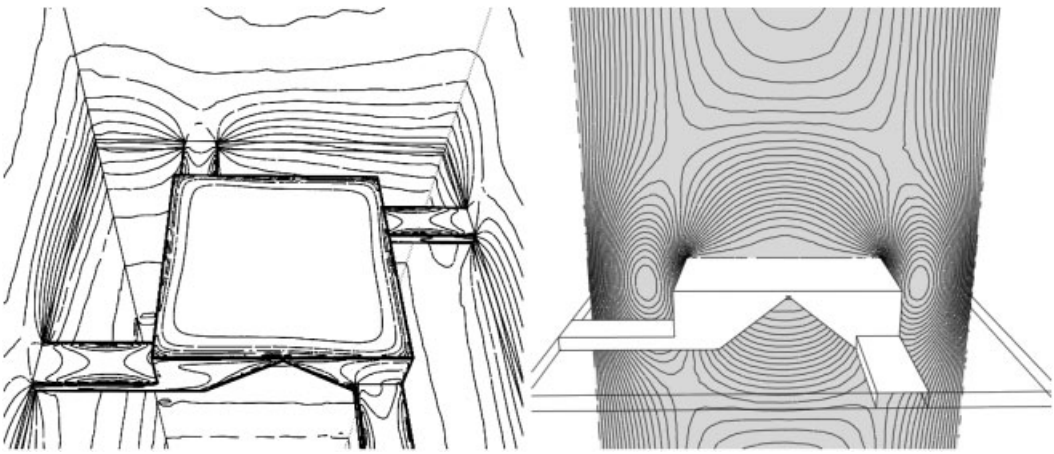

Figure 4. Steady-state pressure (left) and velocity (right) distributions.

Figure 4 shows the steady-state solution. Shown are the pressure contours on the solid surfaces and the velocity contours on a vertical cut-plane through the centre of the computational domain (the shaded region indicates the cut-plane). In Figure 5 are two images of the final adapted surface mesh. On the left is the same view as in Figure 3; note that most of the refinement occurs on the sidewalls. On the right is a close-up of the adapted mesh near one of the support arms. The final adapted mesh contained approximately 3.4×10^6 tetrahedron. No mesh adaption was needed after the third adaption pass.

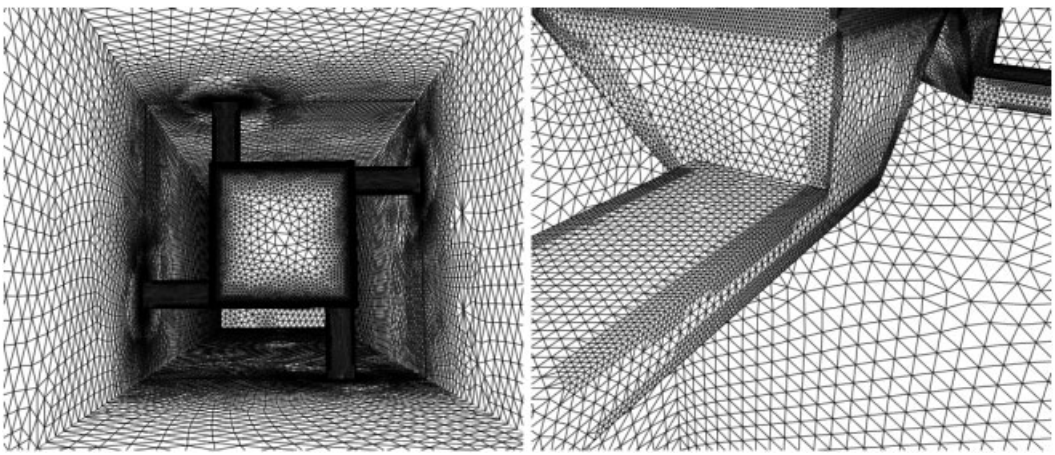

Figure 5. Adapted surface mesh viewed from the inflow plane (left) and near a support arm (right).

7.2. *Microfilter*

The second sample problem is a water flow through a cylindrical microfilter. Figure 6 shows the geometry definition and the initial surface mesh near the membrane. The filter membrane is 5 μm thick, the holes have a diameter of 10 μm, and the membrane diameter is 500 μm. The flow-through area is approximately 17% of the total membrane area. Due to symmetry, only one fourth of the geometry is actually simulated. The initial volume mesh contained approximately 5.6×10^5 tetrahedron. The boundary conditions are similar to those for the microsensor, with the exception of a symmetry condition on the planar side walls. The peak inflow velocity was 100 μm/s. The Reynolds number based on the membrane thickness was 5×10^{-4}. Mesh adaption was performed on the first time step and every 100 subsequent time steps. Two levels of refinement were allowed for this case, with velocity magnitude as the indicator variable.

The results for this problem are shown in Figures 7 and 8. Figure 7 shows the steady-state velocity contours on the symmetry walls (left) and on a vertical cut-plane (right); Figure 8 shows final adapted surface mesh from both external and internal viewpoints. The final volume mesh contained approximately 7.0×10^6 tetrahedron and had a variation in edge size of approximately 10^3. The second level of refinement was needed only in the membrane holes and the immediate surrounding area.

7.3. *Particulate cluster*

The final sample problem consists of a generic liquid flow past a cluster of spherical particles at a Reynolds number of 10^{-1}. The cluster is affixed to the bottom and side walls of the channel. Each particle has a diameter of ≈ 10 μm, and the cluster has a maximum length of ≈ 100 μm. The channel is approximately 150 μm square. The geometry definition and surface mesh are shown in Figure 9. The volume mesh contained approximately 8.3×10^6 tetrahedron. In this case, the discrete surface faces which comprised the particle cluster resulted in a

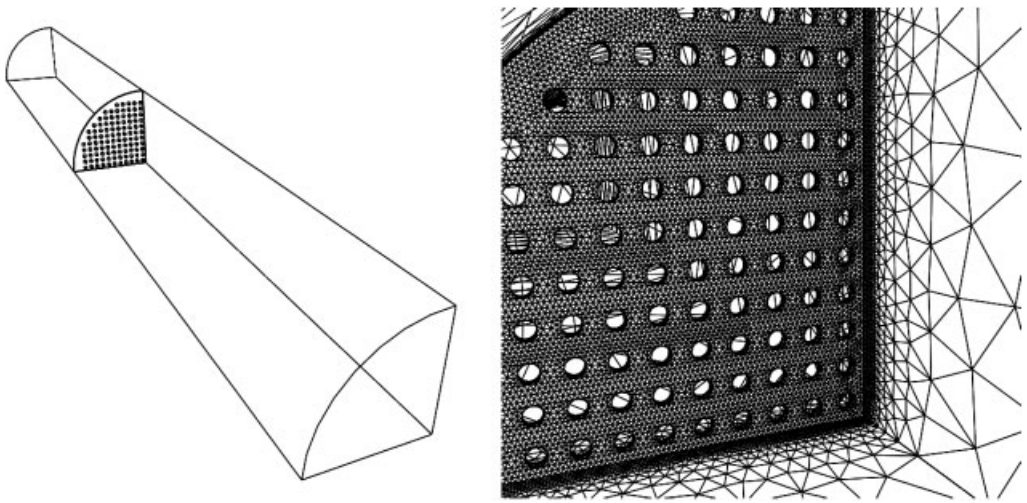

Figure 6. Geometry definition (left) and surface mesh (right) for microfilter.

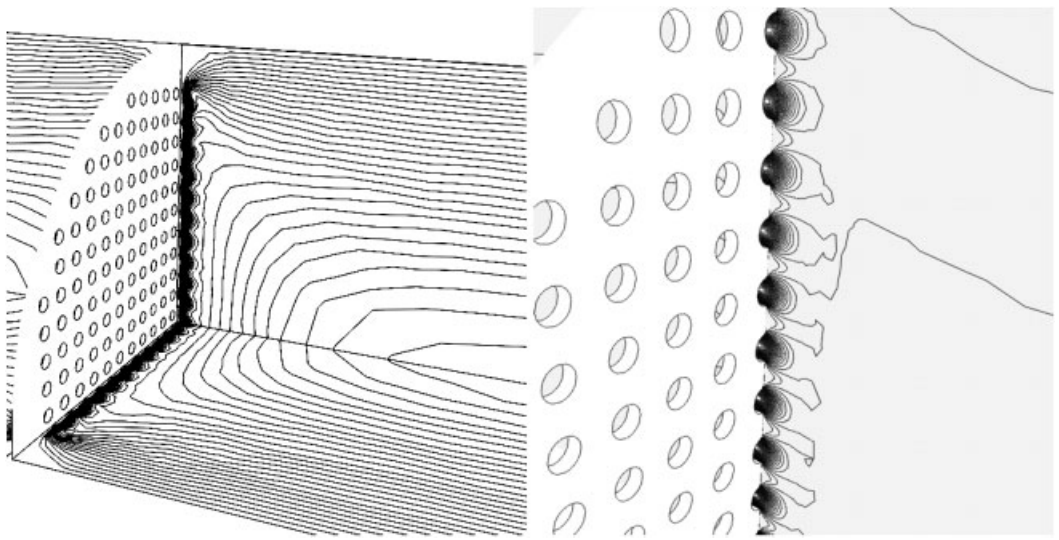

Figure 7. Steady-state velocity contours on symmetry walls (left) and vertical cut-plane (right).

volume mesh of sufficient resolution. Therefore, no mesh adaption was needed. The boundary conditions for this case were similar to the microsensor problem.

Figure 10 shows the steady-state velocity contours on horizontal cut-planes. Figure 11 shows the steady-state contours on vertical cut-planes. Given the complex nature of the particulate geometry, the flow patterns are surprisingly simple. This relative simplicity is clearly due to the overwhelmingly viscous nature of the flow.

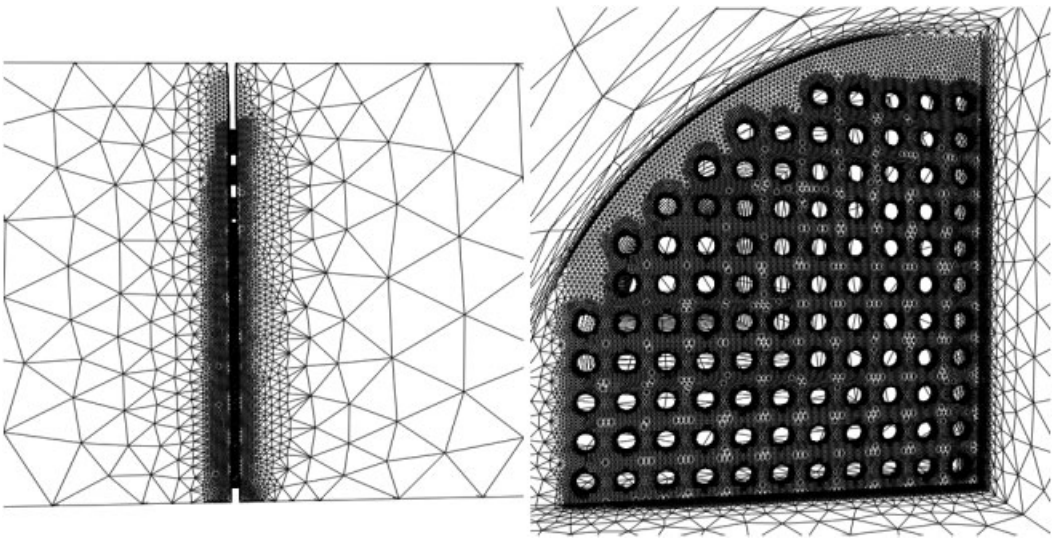

Figure 8. Final adapted mesh on symmetry wall (left) and membrane surface (right).

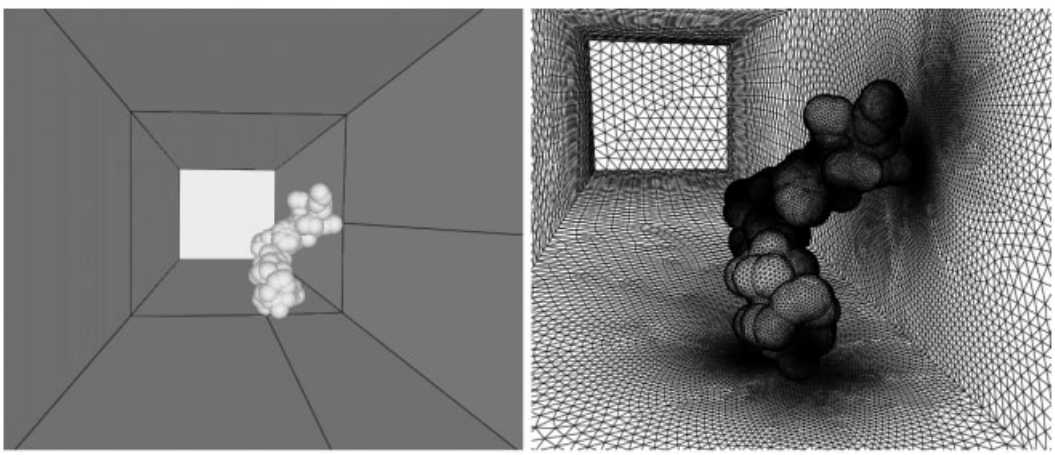

Figure 9. Geometry definition (left) and surface mesh (right) for particulate cluster.

8. CONCLUSIONS

A methodology for the simulation of microfluidics problems has been presented. The methodology is based on the incompressible Navier–Stokes equations and therefore is suitable for both gaseous and liquid flows. The governing equations are solved by combining an adaptive unstructured Finite Element method for spatial discretization with a FS scheme for temporal discretization. Parallelization is achieved for shared memory architectures via an algebraic

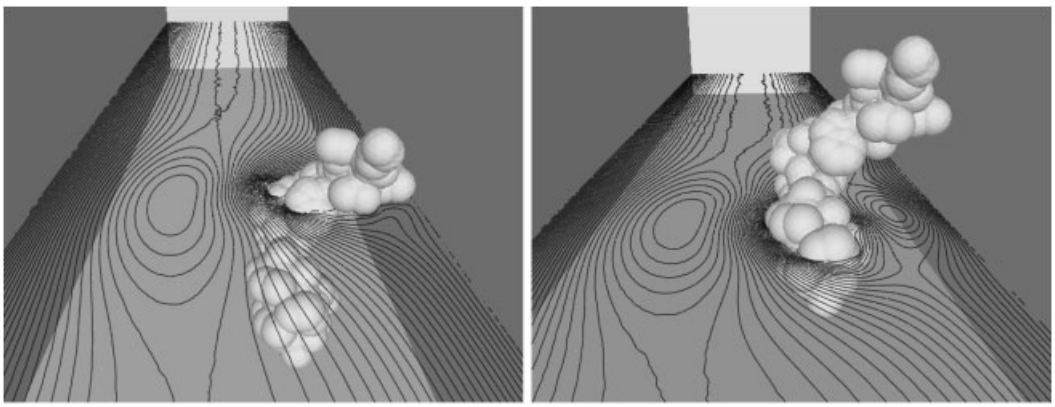

Figure 10. Steady-state velocity contours on horizontal cut-planes.

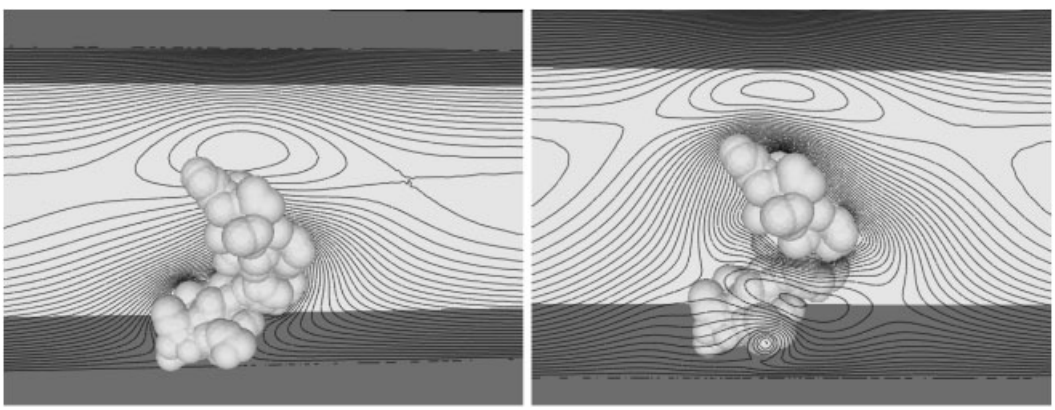

Figure 11. Steady-state velocity contours on vertical cut-planes.

domain decomposition. The ability of the methodology to simulate flows in microscale devices has been demonstrated through a number of sample applications.

What remains for future work are two primary issues. The first of these is computational efficiency. The primary bottleneck in the numerical methodology is the Poisson-pressure equation: upwards of 90% of the CPU time might be spent in its solution. Better preconditioners [32–34] can potentially offer significant gains in performance for relatively modest increases in complexity. Unstructured multigrid methods represent another possibility, and have been explored for the solution of the Poisson-pressure equation on dynamic unstructured grids [35, 36]. While these studies were useful from a proof-of-concept perspective, they nonetheless suggest that perhaps a better approach would be to apply multigrid to the entire system of equations, as is typically done with compressible flow solvers [37]. Lastly, a fully implicit time integration could substantially decrease the number of time steps required for many simulations. A preliminary version of a fully implicit solver has been implemented, but the overall

benefit of such a scheme compared to the semi-implicit scheme for steady state problems is still unclear: the time step for the semi-implicit scheme is already a factor of 10^3 higher than that for a fully explicit scheme. There is no question that a fully implicit scheme will facilitate the simulation of unsteady problems.

A second area for future work is the incorporation of additional flow physics. Numerous MEMS-related problems are multi-disciplinary in nature, and the utility of any simulation which does not involve all of the related physical phenomena is obviously limited. Examples might include surface physics [38], reacting flows [39], or molecular phenomena [3]. For compressible microflows, the continuum approximation may begin to break down. In these cases a coupling to Direct Simulation Monte Carlo methods [40, 41] is needed to ensure validity of the underlying equations throughout the computational domain. Although some work has been done in all of these areas, numerous challenges, from algorithmic issues to boundary conditions to the simple question of computational feasibility, remain.

ACKNOWLEDGEMENTS

This work was supported by a National Research Council Postdoctoral Research Associateship at the Naval Research Laboratory, Washington DC. The assistance of Drs. W.C. Sandberg (NRL), R. Löhner (GMU), and O. Soto (GMU) is gratefully acknowledged.

REFERENCES

1. Maluf N. *An Introduction to Microelectromechanical Systems Engineering*. Artech House: Boston, 2000.
2. Schaff S, Chambre P. *Fundamentals of Gas Dynamics*. Princeton University Press: Princeton, 1958.
3. Allen MP, Tildesley DJ. *Computer Simulation of Liquids*. Clarendon Press: Oxford, 1987.
4. Waltz J. A parallel unstructured finite element solver for incompressible flows over microscale devices. *AIAA Technical Paper* 2002-1055, 2002.
5. Barth TJ. Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes. *AIAA Technical Paper* 1991-0721, 1991.
6. Mavriplis DJ. Three-dimensional unstructured multigrid for the Euler equations. *AIAA Technical Paper* 1991-1549, 1991.
7. Luo H, Baum JD, Löhner R. Edge-based finite element schemes for the Euler equations. *AIAA Journal* 1994; **32**:1183.
8. Peraire J, Peiro J. A 3D finite element multigrid solver for the Euler equations. *AIAA Technical Paper* 1992-0449, 1992.
9. Wright JA, Smith RW. An edge-based method for incompressible Navier–Stokes equations on polygonal meshes. *Journal of Computational Physics* 2001; **169**:24.
10. Thomadakis M, Leshziner M. A pressure correction method for the solution of incompressible viscous flows on unstructured grids. *Journal of Computational Physics* 1996; **22**:581.
11. Löhner R. *Applied Computational Fluid Dynamics Techniques*. Wiley: New York, 2001.
12. Barth TJ, Jespersen DC. The design and application of upwind schemes on unstructured meshes. *AIAA Technical Paper* 1989-0366, 1989.
13. Barth TJ, Frederickson PO. Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. *AIAA Technical Paper* 1990-0013, 1990.
14. Codina R. Pressure stability in fractional step finite element methods for incompressible flows. *Journal of Computational Physics* 2001; **170**:112–140.
15. Codina R. A nodal-based implementation of a stabilized finite element method for incompressible flow problems. *International Journal for Numerical Methods in Fluids* 2000; **33**:711–736.
16. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C*. Cambridge University Press: New York, 1992.
17. Mavriplis DJ. Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes. *AIAA Journal* 1990; **28**:213.
18. Mavriplis DJ. Adaptive meshing techniques for viscous flow solvers on anisotropic unstructured meshes. *International Journal for Numerical Methods in Fluids* 2000; **34**:2.

19. Löhner R, Baum J. Adaptive h-refinement on 3-D unstructured grids for transient problems. *International Journal for Numerical Methods in Fluids* 1992; **14**:1407.
20. Kallinderis Y, Vijayan P. Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes. *AIAA Journal* 1993; **31**:1440.
21. Peraire J, Peiro J, Morgan K. Adaptive remeshing for three-dimensional compressible flow calculations. *Journal of Computational Physics* 1992; **103**:2.
22. Habashi WG, Dompierre J, Bourgault Y, Ait-Ali-Yahia D, Fortin M, Vallet M-G. Anisotropic mesh adaption: towards user-independent, mesh-independent, and solver-independent CFD; Part I: general principles. *International Journal for Numerical Methods in Fluids* 2000; **32**:725.
23. Tam A, Ait-Ali-Yahia D, Robichaud MP, Moore M, Kozel V, Habashi WG. Anisotropic mesh adaption for 3D flows on structured and unstructured grids. *Computational Methods in Applied Mechanics and Engineering* 2000; **189**:1205.
24. Alauzet F, George PL, Mohammadi B, Frey P, Borouchaki H. Transient fixed point based unstructured mesh adaption. *ECCOMAS CFD Conference*, University of Wales, Swansea, UK, 2001.
25. Aftosmis MJ, Berger MJ. Multilevel error estimation and adaptive *h*-refinement for Cartesian meshes with embedded boundaries. *AIAA Technical Paper* 2002-0863, 2002.
26. Waltz J. Parallel adaptive refinement for 3D unstructured grids. *AIAA Technical Paper* 2003-1115, 2003.
27. Waltz J. Parallel adaptive refinement for unsteady flow calculations on 3D unstructured grids. LANL Report LA-UR-05-5637. *International Journal for Numerical Methods in Fluids* 2003, submitted.
28. Waltz J. Derived data structure algorithms for unstructured finite element meshes. *International Journal for Numerical Methods in Engineering* 2002; **54**:9.
29. Löhner R. Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**:95.
30. Löhner R. Some useful renumbering strategies for unstructured grids. *International Journal for Numerical Methods in Engineering* 1993; **36**.
31. White FM. *Viscous Fluid Flow* (2nd edn). McGraw-Hill: New York, 1991.
32. Qin N, Ludlow DK, Shaw ST. A matrix-free preconditioned Newton/GMRES method for unsteady Navier–Stokes solutions. *International Journal for Numerical Methods in Fluids* 2000; **33**:2.
33. Candler GV, Wright MJ, McDonald JD. Data-parallel lower-upper relaxation method for reacting flows. *AIAA Journal* 1994; **32**:12.
34. Martin D, Löhner R. An implicit linelet-based solver for incompressible flows. *AIAA Technical Paper* 1992-0668, 1992.
35. Waltz J, Löhner R. A grid coarsening algorithm for unstructured multigrid applications. *AIAA Technical Paper* 2000-1024, 2000.
36. Waltz J. Unstructured multigrid for time dependent incompressible fluid flow. *Ph.D. Thesis*, George Mason University, 2000.
37. Jameson A. Solution of the Euler equations by a multigrid method. *Applied Mathematics and Computation* 1983; **13**.
38. Mazumder S, Lowry SA. The treatment of reacting surfaces for finite-volume schemes on unstructured meshes. *Journal of Computational Physics* 2001; **173**:2.
39. Oran E, Boris J. *Numerical Simulation of Reactive Flow* (2nd edn). Cambridge University Press: Cambridge, 2001.
40. Bird GA. *Molecular Dynamics and the Direct Simulation of Gas Flow*. Oxford Science Publications: Oxford, 1994.
41. Piekos ES, Bruer KS. Numerical modeling of micromechanical devices using the Direct Simulation Monte Carlo method. *Journal of Fluids Engineering* 1996; **464**:118.
42. Hirsch C. *Numerical Computation of Internal and External Flows*, Vols. 1 and 2. Wiley: New York, 1988.

