INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2005; **47**:633–654
Published online 22 December 2004 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.836

Enhanced-discretization successive update method (EDSUM)

Tayfun E. Tezduyar*,† and Sunil Sathe

Team for Advanced Flow Simulation and Modeling (TAFSM), Mechanical Engineering, Rice University–MS 321, 6100 Main Street, Houston, TX 77005, U.S.A.

SUMMARY

The enhanced-discretization successive update method (EDSUM) is a multi-level iteration method designed for computation of the flow behaviour at small scales. As an enhancement in iterative solution of non-linear and linear equation systems, the EDSUM is one of the enhanced discretization and solution techniques developed for more effective computation of complex flow problems. It complements techniques based on enhancement in spatial discretization and based on enhancement in time discretization in the context of a space–time formulation. It is closely related to the enhanced-discretization interface-capturing technique (EDICT), as the function spaces used in the EDSUM are very similar to those used in the EDICT. The EDSUM also has a built-in mechanism for transferring flow information between the large and small scales in a fashion consistent with the discretizations resulting from the underlying stabilized formulations. With a number of test computations for steady-state problems governed by the advection–diffusion equation, we demonstrate that the EDSUM has the potential to become a competitive technique for computation of flow behaviour at small scales. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: flow simulation; enhanced discretization and solution techniques; iterative methods; successive update method; multi-level iteration techniques

1. INTRODUCTION

A number of enhanced discretization and solution techniques were developed in recent years for more accurate and more efficient computation of complex flow problems, including flows with moving boundaries and interfaces. These techniques are based on enhancement in spatial discretization, enhancement in time discretization, and enhancement in iterative solution of non-linear and linear equation systems. The enhanced discretization and solution techniques

*Correspondence to: Tayfun E. Tezduyar, Department of Mechanical Engineering, Rice University–MS 321, 6100 Main Street, TX 77005 Houston, U.S.A.
†E-mail: tezduyar@rice.edu, Web: http://www.mems.rice.edu/TAFSM/

Contract/grant sponsor: U.S. Army Natick Soldier Centre
Contract/grant sponsor: NASA Johnson Space Centre

Received 13 February 2004
Accepted 18 August 2004

are built around the core formulations that we rely on in simulation and modelling complex flow problems. These core methods include the stabilized finite element techniques such as the streamline-upwind/Petrov–Galerkin (SUPG) [1–4] and pressure-stabilizing/Petrov–Galerkin (PSPG) [5] formulations, as well as interface-tracking and interface-capturing techniques (see References [5–11]). The PSPG formulation is based on an earlier pressure-stabilizing formulation [12] that was designed for Stokes flows.

Stabilized formulations prevent numerical instabilities in solving problems with high Reynolds or Mach numbers and shocks or thin boundary layers, as well as when using equal-order interpolation functions for velocity and pressure. The SUPG and PSPG formulations are among the stabilized methods that achieve these objectives without introducing excessive numerical dissipation. Stabilized formulations also substantially improve the convergence rate in iterative solution of the large, coupled non-linear equation system that needs to be solved at every time step of a flow computation.

In an interface-tracking technique the mesh moves to track (follow) the interface. The deforming-spatial-domain/stabilized space–time (DSD/SST) formulation [5, 13, 14] is an interface-tracking technique where the finite element formulation of the problem is written over its space–time domain. As the spatial domain occupied by the fluid changes its shape in time, the mesh needs to be updated. In general, this is accomplished by moving the mesh with the automatic mesh moving technique introduced in Reference [15], where the motions of the nodes are governed by the equations of elasticity. A full or partial remeshing (i.e. generating a new set of elements, and sometimes also a new set of nodes) is performed when the elements are distorted beyond an acceptable level. The stabilized space–time formulations were used earlier by other researchers to solve problems with fixed spatial domains (see for example Reference [16]).

In an interface-capturing technique the mesh does not move to follow the interface, but instead the interface is 'captured' by somehow locating it over the non-moving mesh. An interface function of one kind or another is used in marking the location of the interface. In addition to the flow equations, an advection equation governing the time-evolution of the interface function is solved over the non-moving mesh. The interface-capturing techniques have the advantage of being free from mesh update requirements, but, for comparable levels of spatial discretization, yield less accurate representation of the interface. Independent of how accurately the interface is located, the accuracy in representing the flow field around the interface will be limited by the resolution of the fluid mesh where the interface happens to be located.

As an enhancement in spatial discretization, the enhanced-discretization interface-capturing technique (EDICT) was introduced in Reference [17] to increase accuracy in representing an interface. In the EDICT, in the stabilized formulations of the flow and advection equations to be solved, the function spaces used are based on enhanced discretization at and near the interface. A subset of the elements in the base mesh, Mesh-1, are identified as those at and near the interface. A more refined mesh, Mesh-2, is constructed by patching together second-level meshes generated over each element in this subset. The interpolation functions for velocity and pressure have two components each: one coming from Mesh-1 and the second one coming from Mesh-2. To further increase the accuracy, a third-level mesh, Mesh-3, is constructed for the interface function only. The construction of Mesh-3 from Mesh-2 is very similar to the construction of Mesh-2 from Mesh-1. The interpolation functions for the interface function have three components, each coming from one of these three meshes. The subsets over which

Mesh-2 and Mesh-3 are built are re-defined not every time step but with sufficient frequency to keep the interface always enveloped in.

The EDSTT was introduced in References [7–9] as an enhancement in time discretization in the context of a space–time formulation. The EDSTT was developed to have more flexibility in carrying out time-accurate computations of fluid–structure interactions where we find it necessary to use smaller time steps for the structural dynamics part. In general, EDSTT can be used in time-accurate computations where, for whatever reason, we require smaller time steps in certain parts of the fluid domain. Test computations for the EDSTT were reported in Reference [18].

The set of enhanced discretization and solution techniques also include enhancements in iterative solution of non-linear and linear equation systems. The enhanced-iteration non-linear solution technique (EINST) [7–9, 19] and the enhanced-approximation linear solution technique (EALST) [7–9, 19] were introduced as complements to the enhancements in spatial and temporal discretizations. The EINST and EALST were developed to increase the performance of the iterative techniques used in solution of the non-linear and linear equation systems when some parts of the computational domain may offer more of a challenge for the iterative method than the others. These two techniques can be used for computations based on semi-discrete or space–time formulations. Test computations for the EALST were reported in Reference [20].

The enhanced-discretization successive update method (EDSUM) was proposed in References [6, 9–11] as another enhancement in iterative solution of non-linear and linear equation systems. It is closely related to the EDICT, in the sense that the function spaces used are very similar to those used in the EDICT. The EDSUM is a multi-level iteration method developed for the purpose of being able to compute the flow behaviour at small scales. It has a built-in mechanism for transferring flow information between the large and small scales. The information transfer is achieved in a fashion consistent with the discretizations resulting from the stabilized formulations.

In Sections 2 and 3 we review the governing equations and the standard stabilized formulations. The enhanced-discretization stabilized formulations are described in Section 4, and the construction of the function spaces in Section 5. In Section 6 we provide a brief explanation of the iterative methods used. The multi-level successive update method is described in Section 7. Test computations for problems governed by the advection–diffusion equation are reported in Section 8, and the concluding remarks are presented in Section 9.

2. GOVERNING EQUATIONS

2.1. Navier–Stokes equations of incompressible flows

Let $\Omega \subset \mathbb{R}^{n_{sd}}$ be the spatial domain with boundary Γ, and $(0, T)$ be the time domain. The Navier–Stokes equations of incompressible flows can be written on Ω and $\forall t \in (0, T)$ as

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \mathbf{f}\right) - \nabla\cdot\boldsymbol{\sigma} = 0 \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

where ρ, $\mathbf{u}$ and $\mathbf{f}$ are the density, velocity and the external force, respectively. The stress tensor $\boldsymbol{\sigma}$ is defined as

$$\boldsymbol{\sigma}(p,\mathbf{u}) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\mathbf{u}) \tag{3}$$

Here p is the pressure, $\mathbf{I}$ is the identity tensor, $\mu = \rho\nu$ is the viscosity, ν is the kinematic viscosity, and $\boldsymbol{\varepsilon}(\mathbf{u})$ is the strain-rate tensor:

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}((\boldsymbol{\nabla}\mathbf{u}) + (\boldsymbol{\nabla}\mathbf{u})^{\mathrm{T}}) \tag{4}$$

The essential and natural boundary conditions for Equation (1) are represented as

$$\mathbf{u} = \mathbf{g} \text{ on } \Gamma_g, \quad \mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h} \text{ on } \Gamma_h \tag{5}$$

where Γ_g and Γ_h are complementary subsets of the boundary Γ, $\mathbf{n}$ is the unit normal vector, and $\mathbf{g}$ and $\mathbf{h}$ are given functions. A divergence-free velocity field $\mathbf{u}_0(\mathbf{x})$ is specified as the initial condition.

2.2. Advection–diffusion equation

As a model equation possessing some of the significant features of Equation (1), we consider the following time-dependent advection–diffusion equation, written on Ω and $\forall t \in (0,T)$ as

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\phi - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\phi) = 0 \tag{6}$$

where ϕ represents the quantity being transported (e.g. temperature, concentration), and ν is the diffusivity, which is separate from (but in mathematical significance very comparable to) the ν representing the kinematic viscosity. The essential and natural boundary conditions associated with Equation (6) are represented as

$$\phi = g \text{ on } \Gamma_g, \quad \mathbf{n}\cdot\nu\boldsymbol{\nabla}\phi = h \text{ on } \Gamma_h \tag{7}$$

A function $\phi_0(\mathbf{x})$ is specified as the initial condition.

3. STANDARD STABILIZED FORMULATIONS

3.1. Advection–diffusion equation

For the advection–diffusion equation given by Equation (6), let us assume that we have constructed some suitably defined finite-dimensional trial solution and test function spaces $\mathcal{S}_\phi^h$ and $\mathcal{V}_\phi^h$. The stabilized finite element formulation can then be written as follows: find $\phi^h \in \mathcal{S}_\phi^h$ such that $\forall w^h \in \mathcal{V}_\phi^h$:

$$\int_\Omega w^h \left(\frac{\partial\phi^h}{\partial t} + \mathbf{u}^h\cdot\boldsymbol{\nabla}\phi^h\right) \mathrm{d}\Omega + \int_\Omega \boldsymbol{\nabla}w^h\cdot\nu\boldsymbol{\nabla}\phi^h \,\mathrm{d}\Omega - \int_{\Gamma_h} w^h \mathrm{h}^h \,\mathrm{d}\Gamma$$

$$+\sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_{\mathrm{SUPG}}\mathbf{u}^h\cdot\boldsymbol{\nabla}w^h \left(\frac{\partial\phi^h}{\partial t} + \mathbf{u}^h\cdot\boldsymbol{\nabla}\phi^h - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\phi^h)\right) \mathrm{d}\Omega = 0 \tag{8}$$

Here n_{el} is the number of elements, Ω^e is the domain for element e, and τ_{SUPG} is the SUPG stabilization parameter. For various ways of calculating τ_{SUPG}, see References [21, 22].

3.2. Navier–Stokes equations of incompressible flows

For the Navier–Stokes equations of incompressible flows, given by Equations (1)–(2), let us assume that we have some suitably defined finite-dimensional trial solution and test function spaces for velocity and pressure: $\mathscr{S}_{\mathbf{u}}^h$, $\mathscr{V}_{\mathbf{u}}^h$, $\mathscr{S}_p^h$ and $\mathscr{V}_p^h (= \mathscr{S}_p^h)$. The stabilized finite element formulation can then be written as follows: find $\mathbf{u}^h \in \mathscr{S}_{\mathbf{u}}^h$ and $p^h \in \mathscr{S}_p^h$ such that $\forall \mathbf{w}^h \in \mathscr{V}_{\mathbf{u}}^h$ and $\forall q^h \in \mathscr{V}_p^h$:

$$\int_\Omega \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) \mathrm{d}\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\, \mathrm{d}\Omega - \int_{\Gamma_h} \mathbf{w}^h \cdot \mathbf{h}^h\, \mathrm{d}\Gamma$$
$$+ \int_\Omega q^h \nabla \cdot \mathbf{u}^h\, \mathrm{d}\Omega + \sum_{e=1}^{n_{\text{el}}} \int_{\Omega^e} \frac{1}{\rho} [\tau_{\text{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}^h + \tau_{\text{PSPG}} \nabla q^h] \cdot [Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h]\, \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{n_{\text{el}}} \int_{\Omega^e} \nu_{\text{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h\, \mathrm{d}\Omega = 0 \tag{9}$$

where

$$Ł(q^h, \mathbf{w}^h) = \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) - \nabla \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h) \tag{10}$$

Here τ_{PSPG} and ν_{LSIC} are the PSPG and LSIC (least-squares on incompressibility constraint) stabilization parameters. For various ways of calculating τ_{PSPG} and ν_{LSIC}, see References [21, 22].

4. ENHANCED-DISCRETIZATION STABILIZED FORMULATIONS

To maintain the generality of the formulations, we allow for cases where the enhanced-discretization zone does not necessarily cover the entire domain and its shape and location may change during the computation. A subset of the elements in the base mesh, Mesh-1, are identified as those constituting the enhanced-discretization zone. A more refined mesh, Mesh-2, is constructed by patching together second-level meshes generated over each element in this subset. The trial and test functions will have two components each, one coming from Mesh-1 and the second one coming from Mesh-2.

4.1. Advection–diffusion equation

At a time level n, the trial function space corresponding to ϕ_n^h is denoted by $(\mathscr{S}_\phi^h)_n$, and the test function space corresponding to the advection–diffusion equation by $(\mathscr{V}_\phi^h)_n$. The subscript n indicates that the spatial discretizations corresponding to different time levels may be different. The enhanced-discretization stabilized formulation of Equation (6) is written as follows: given

ϕ_n^h, find $\phi_{n+1}^h \in (\mathscr{S}_\phi^h)_{n+1}$, such that $\forall w_{n+1}^h \in (\mathscr{V}_\phi^h)_{n+1}$:

$$\int_\Omega w_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) \mathrm{d}\Omega + \int_\Omega \nabla w_{n+1}^h \cdot \nu \nabla \phi^h \, \mathrm{d}\Omega - \int_{\Gamma_h} w_{n+1}^h \mathrm{h}^h \, \mathrm{d}\Gamma$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_{\mathrm{SUPG}} \mathbf{u}^h \cdot \nabla w_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h - \nabla \cdot (\nu \nabla \phi^h) \right) \mathrm{d}\Omega = 0 \tag{11}$$

The trial and test functions, at a time level n, are defined as

$$\phi_n^h = \phi_n^1 + \phi_n^2 \tag{12}$$

$$w_n^h = w_n^1 + w_n^2 \tag{13}$$

where superscripts 1 and 2 denote the components of the functions coming from Mesh-1 and Mesh-2, respectively.

4.2. Navier–Stokes equations of incompressible flows

At a time level n, the trial function spaces corresponding to the velocity and pressure are denoted by $(\mathscr{S}_{\mathbf{u}}^h)_n$ and $(\mathscr{S}_p^h)_n$. The test function spaces corresponding to the momentum equation and incompressibility constraint are denoted by $(\mathscr{V}_{\mathbf{u}}^h)_n$ and $(\mathscr{V}_p^h)_n$ $(=(\mathscr{S}_p^h)_n)$. The enhanced-discretization stabilized formulation of Equations (1)–(2) is written as follows: given $\mathbf{u}_n^h$, find $\mathbf{u}_{n+1}^h \in (\mathscr{S}_{\mathbf{u}}^h)_{n+1}$ and $p_{n+1}^h \in (\mathscr{S}_p^h)_{n+1}$, such that $\forall \mathbf{w}_{n+1}^h \in (\mathscr{V}_{\mathbf{u}}^h)_{n+1}$ and $\forall q_{n+1}^h \in (\mathscr{V}_p^h)_{n+1}$:

$$\int_\Omega \mathbf{w}_{n+1}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) \mathrm{d}\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}_{n+1}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \, \mathrm{d}\Omega$$
$$- \int_{\Gamma_h} \mathbf{w}_{n+1}^h \cdot \mathbf{h}^h \, \mathrm{d}\Gamma + \int_\Omega q_{n+1}^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \frac{1}{\rho} [\tau_{\mathrm{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}_{n+1}^h + \tau_{\mathrm{PSPG}} \nabla q_{n+1}^h] \cdot [Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h] \, \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}_{n+1}^h \rho \nabla \cdot \mathbf{u}^h \, \mathrm{d}\Omega = 0 \tag{14}$$

The trial and test functions, at a time level n, are defined as

$$\mathbf{u}_n^h = \mathbf{u}_n^1 + \mathbf{u}_n^2 \tag{15}$$

$$p_n^h = p_n^1 + p_n^2 \tag{16}$$

$$\mathbf{w}_n^h = \mathbf{w}_n^1 + \mathbf{w}_n^2 \tag{17}$$

$$q_n^h = q_n^1 + q_n^2 \tag{18}$$

5. CONSTRUCTION OF THE FUNCTION SPACES

In constructing the function spaces corresponding to a time level n, we start with a base mesh (Mesh-1), with the set of elements and nodal points denoted by ε_n^1 and η_n^1. The subscript n implies that Mesh-1 itself might change from one time level to another.

A second-level and more refined mesh (Mesh-2) is constructed over a subset $(\varepsilon_n^1)_n^2$ of these elements. Mesh-2 is generated by patching together the second-level meshes generated over each of the elements in $(\varepsilon_n^1)_n^2$. The second subscript n implies that for a given Mesh-1, which elements of this mesh are declared to be in $(\varepsilon_n^1)_n^2$ might change from one time level to other. An element which might be declared to be in $(\varepsilon_n^1)_n^2$ at some time level, might fall out of it at some other time, and yet come back in again some time later. For each element in ε_n^1, there will be a unique second-level mesh. Therefore, if an element is declared to be in $(\varepsilon_n^1)_n^2$ for a second time, the refined mesh generated over that element at the earlier declaration can be reused. If an automatic mesh generator is being used to generate these second-level meshes, the cost for that mesh generation will be a one-time cost. The set of elements and nodal points for Mesh-2 are denoted by ε_n^2 and η_n^2.

The function ϕ_n^1 comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^1. The function ϕ_n^2 comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^2, excluding those coinciding with the nodes in η_n^1, and also excluding those at the boundaries of the zones covered by the elements in ε_n^2 unless those boundaries coincide with the boundaries of Ω. The sum of the two trial functions, $\phi_n^1 + \phi_n^2$, needs to satisfy the essential boundary conditions. We construct $\mathbf{u}_n^1$, $\mathbf{u}_n^2$, p_n^1, and p_n^2 in exactly the same way, except for recognizing that for p_n^1 and p_n^2 the references to essential boundary conditions do not apply.

The components of each test function are defined in the same way as we did for the trial functions, except that the test functions need to satisfy the homogeneous form of the essential boundary conditions.

We do not update $(\varepsilon_n^1)_n^2$ every time step. We update it frequently enough to meet our objective of having enhanced discretization at the zones specified by some criteria. How long we can compute without re-defining this subset will depend on how much larger we decide to keep it compared to the level dictated by the criteria used. The more we exceed the level dictated, the longer we can compute before we need to re-define it again. Whenever we re-define this subset, the mesh generation cost will not be a significant one. If we are using an automatic mesh generator for the second mesh, we will be able to use and reuse the meshes which were generated (and stored) the first time these meshes were needed.

6. ITERATIVE SOLUTION METHODS

Full discretizations of the formulations described in the earlier sections lead to coupled, non-linear equation systems that need to be solved at every time step of the simulation. We can represent the equation system that needs to be solved as follows:

$$\mathbf{N}(\mathbf{d}_{n+1}) = \mathbf{F} \tag{19}$$

where $\mathbf{d}_{n+1}$ is the vector that contains the nodal unknowns associated with marching from time level n to $n+1$.

We solve Equation (19) with the Newton–Raphson method:

$$\left.\frac{\partial \mathbf{N}}{\partial \mathbf{d}}\right|_{\mathbf{d}^i_{n+1}} (\Delta \mathbf{d}^i_{n+1}) = \mathbf{F} - \mathbf{N}(\mathbf{d}^i_{n+1}) \tag{20}$$

where i is the step counter for the Newton–Raphson sequence, and $\Delta \mathbf{d}^i_{n+1}$ is the increment computed for $\mathbf{d}^i_{n+1}$. The linear equation system we see in Equation (20) needs to be solved at every step of the Newton–Raphson sequence. We can represent Equation (20) as a linear equation system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{21}$$

In the class of computations we typically carry out, this equation system would be too large to solve with a direct method. Therefore we solve it iteratively. The objective would be to drive the residual,

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x} \tag{22}$$

down to an acceptable level. Iterations involve repeated computations of the form $\mathbf{A}\mathbf{x}$ and

$$\Delta \mathbf{y} = \mathbf{P}^{-1}\mathbf{r} \tag{23}$$

where $\mathbf{P}$, the preconditioning matrix, is an approximation to $\mathbf{A}$. The preconditioner $\mathbf{P}$ has to be simple enough to form and factorize efficiently. However, it also has to be sophisticated enough to yield a desirable convergence rate. A computation of the form $\mathbf{A}\mathbf{x}$ can be performed in several different ways. It can be based on a sparse-matrix storage of $\mathbf{A}$. It can also be based on storing just element-level matrices (element-matrix-based), or even just element-level vectors (element-vector-based). This last strategy is also called the matrix-free technique. How to update the solution vector $\mathbf{x}$ is also a major subject in iterative solution techniques. Several update methods are available, and we use the GMRES [23] method.

We have been focusing our research related to iterative methods mainly on carrying out the computations of the form $\mathbf{A}\mathbf{x}$ efficiently and selecting a good preconditioner $\mathbf{P}$. In these efforts we have always been keeping in mind that the iterative solution methods we develop need to be efficiently implemented on parallel computing platforms. For example, as part of designing 'parallel-ready' iterative solution techniques, for computations of the form $\mathbf{A}\mathbf{x}$, we have made use of the element-matrix-based [24], element-vector-based [24], and sparse-matrix-based [25] techniques. Furthermore, we proposed in References [6, 9, 19] a mixed analytical/numerical element-vector-based (AEVB/NEVB) computation technique as well as a mixed element-matrix-based/element-vector-based (EMB/EVB) technique. The element-vector-based methods were successfully used also by other researchers in the context of parallel computations (see for example References [26, 27]).

In preconditioner design, we developed some advanced preconditioners such as the clustered-element-by-element (CEBE) preconditioner [28] and the mixed CEBE and cluster companion (CC) preconditioner [28]. We have implemented, with quite satisfactory results, the CEBE preconditioner in conjunction with an ILU approximation [25]. However, our typical computations are based on diagonal and nodal–block–diagonal preconditioners. These are simple preconditioners, but are also easy to implement on parallel platforms. More on our parallel implementations can be found in Reference [24].

7. MULTI-LEVEL SUCCESSIVE UPDATE METHOD

In the context of enhanced-discretization, a non-linear equation system of the kind given by Equation (19) involves four classes of nodes. Class-1 consists of all the Mesh-1 nodes. These nodes are connected to each other through the Mesh-1 elements. Class-2E consists of the Mesh-2 edge nodes (but excluding those coinciding with the Mesh-1 nodes). The edge nodes associated with different edges are not connected (except those at each side of an edge, but we could possibly neglect that a side node might be connected to the side nodes of the adjacent edges). Nodes within an edge are connected through Mesh-2 elements. Class-2F contains the Mesh-2 face nodes (but excluding those on the edges). The face nodes associated with different faces are not connected (except those at sides of a face, but we could possibly neglect that those side nodes might be connected to the side nodes of the adjacent face). Nodes within a face are connected through Mesh-2 elements. Class-2I nodes are the Mesh-2 interior nodes. The interior nodes associated with different clusters of Mesh-2 elements are not connected. Nodes within a cluster are connected through Mesh-2 elements. Based on this multi-level decomposition concept, Equation (19) can be re-written, as described in References [6, 9–11], in the following form:

$$\begin{aligned}
\mathbf{N}_1(\mathbf{d}_1, \mathbf{d}_{2E}, \mathbf{d}_{2F}, \mathbf{d}_{2I}) &= \mathbf{F}_1 \\
\mathbf{N}_{2E}(\mathbf{d}_1, \mathbf{d}_{2E}, \mathbf{d}_{2F}, \mathbf{d}_{2I}) &= \mathbf{F}_{2E} \\
\mathbf{N}_{2F}(\mathbf{d}_1, \mathbf{d}_{2E}, \mathbf{d}_{2F}, \mathbf{d}_{2I}) &= \mathbf{F}_{2F} \\
\mathbf{N}_{2I}(\mathbf{d}_1, \mathbf{d}_{2E}, \mathbf{d}_{2F}, \mathbf{d}_{2I}) &= \mathbf{F}_{2I}
\end{aligned} \tag{24}$$

where the subscript '$n+1$' has been dropped to simplify the notation.

In the approximate Newton–Raphson method proposed in References [6, 9–11] to solve Equation (24), at each non-linear iteration step, we successively update the solution vectors corresponding to each class. While updating each class, we use the most recent values of the solution vectors in calculating the vectors $\mathbf{N}_1$, $\mathbf{N}_{2E}$, $\mathbf{N}_{2F}$, and $\mathbf{N}_{2I}$ and their derivatives with respect to the solution vectors. We start with updating the Class-1 nodes, then update the Class-2E, Class-2F, and Class-2I nodes, respectively. The steps followed can be written, as done in References [9–11, 19], in the form shown below, where each class of equations are solved in the order they are written.

$$\begin{aligned}
\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i)} (\Delta \mathbf{d}_1^i) &= \mathbf{F}_1 - \mathbf{N}_1(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i) \\
\left.\frac{\partial \mathbf{N}_{2E}}{\partial \mathbf{d}_{2E}}\right|_{(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i)} (\Delta \mathbf{d}_{2E}^i) &= \mathbf{F}_{2E} - \mathbf{N}_{2E}(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i) \\
\left.\frac{\partial \mathbf{N}_{2F}}{\partial \mathbf{d}_{2F}}\right|_{(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i)} (\Delta \mathbf{d}_{2F}^i) &= \mathbf{F}_{2F} - \mathbf{N}_{2F}(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^i, \mathbf{d}_{2I}^i) \\
\left.\frac{\partial \mathbf{N}_{2I}}{\partial \mathbf{d}_{2I}}\right|_{(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^i)} (\Delta \mathbf{d}_{2I}^i) &= \mathbf{F}_{2I} - \mathbf{N}_{2I}(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^i)
\end{aligned} \tag{25}$$

This sequence would be repeated as many times as needed, and, as an option, as proposed in References [9, 10, 19], we could alternate between this sequence and its reverse sequence:

$$\begin{aligned}
\left.\frac{\partial \mathbf{N}_{2I}}{\partial \mathbf{d}_{2I}}\right|_{(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i)} (\Delta \mathbf{d}_{2I}^i) &= \mathbf{F}_{2I} - \mathbf{N}_{2I}(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i) \\
\left.\frac{\partial \mathbf{N}_{2F}}{\partial \mathbf{d}_{2F}}\right|_{(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^{i+1})} (\Delta \mathbf{d}_{2F}^i) &= \mathbf{F}_{2F} - \mathbf{N}_{2F}(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^{i+1}) \\
\left.\frac{\partial \mathbf{N}_{2E}}{\partial \mathbf{d}_{2E}}\right|_{(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1})} (\Delta \mathbf{d}_{2E}^i) &= \mathbf{F}_{2E} - \mathbf{N}_{2E}(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1}) \\
\left.\frac{\partial \mathbf{N}_{1}}{\partial \mathbf{d}_{1}}\right|_{(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1})} (\Delta \mathbf{d}_{1}^i) &= \mathbf{F}_{1} - \mathbf{N}_{1}(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1})
\end{aligned} \tag{26}$$

Updating the solution vector corresponding to each class would also require solution of a large equation system. These equations systems would each be solved iteratively, with an effective preconditioner, a reliable search technique, and parallel implementation. It is important to note that the bulk of the computational cost would be for Class-1 and Class-2I. It was proposed in References [6, 9–11] that while the Class-1 nodes would be partitioned to different processors of the parallel computer, for the remaining classes, nodes in each edge, face or interior cluster would be assigned to the same processor. Therefore, solution of each edge, face or interior cluster would be local. If the size of each interior cluster becomes too large, then, as proposed in References [6, 9–11], nodes for a given cluster can also be distributed across different processors, or a third level of mesh refinement can be introduced to make the enhanced discretization a tri-level kind.

A variation of the EDSUM was proposed in References [9, 10, 19], for the iterative solution of the linear equation system that needs to be solved at every step of a (full) Newton–Raphson method applied to Equation (24). To describe this variation, we first re-write Equation (21) based on the multi-level decomposition concept:

$$\begin{aligned}
\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12E}\mathbf{x}_{2E} + \mathbf{A}_{12F}\mathbf{x}_{2F} + \mathbf{A}_{12I}\mathbf{x}_{2I} &= \mathbf{b}_1 \\
\mathbf{A}_{2E1}\mathbf{x}_1 + \mathbf{A}_{2E2E}\mathbf{x}_{2E} + \mathbf{A}_{2E2F}\mathbf{x}_{2F} + \mathbf{A}_{2E2I}\mathbf{x}_{2I} &= \mathbf{b}_{2E} \\
\mathbf{A}_{2F1}\mathbf{x}_1 + \mathbf{A}_{2F2E}\mathbf{x}_{2E} + \mathbf{A}_{2F2F}\mathbf{x}_{2F} + \mathbf{A}_{2F2I}\mathbf{x}_{2I} &= \mathbf{b}_{2F} \\
\mathbf{A}_{2I1}\mathbf{x}_1 + \mathbf{A}_{2I2E}\mathbf{x}_{2E} + \mathbf{A}_{2I2F}\mathbf{x}_{2F} + \mathbf{A}_{2I2I}\mathbf{x}_{2I} &= \mathbf{b}_{2I}
\end{aligned} \tag{27}$$

where

$$\mathbf{A}_{\beta\gamma} = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma} \tag{28}$$

with $\beta,\ \gamma = 1, 2E, 2F, 2I$. Then, for the iterative solution of Equation (27), we define two preconditioners:

$$\mathbf{P}_{\mathrm{LTOS}} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & 0 \\ \mathbf{A}_{2E1} & \mathbf{A}_{2E2E} & 0 & 0 \\ \mathbf{A}_{2F1} & \mathbf{A}_{2F2E} & \mathbf{A}_{2F2F} & 0 \\ \mathbf{A}_{2I1} & \mathbf{A}_{2I2E} & \mathbf{A}_{2I2F} & \mathbf{A}_{2I2I} \end{bmatrix} \tag{29}$$

$$\mathbf{P}_{\mathrm{STOL}} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12E} & \mathbf{A}_{12F} & \mathbf{A}_{12I} \\ 0 & \mathbf{A}_{2E2E} & \mathbf{A}_{2E2F} & \mathbf{A}_{2E2I} \\ 0 & 0 & \mathbf{A}_{2F2F} & \mathbf{A}_{2F2I} \\ 0 & 0 & 0 & \mathbf{A}_{2I2I} \end{bmatrix} \tag{30}$$

It was proposed in References [9, 10, 19] that these two preconditioners are used alternatingly during the inner iterations of the GMRES search. We note that this mixed preconditioning technique with multi-level discretization is closely related to the mixed CEBE and CC preconditioning technique [28] we referred to in Section 6. Along these lines, as a mixed preconditioning technique that is more closely related to the mixed CEBE and CC technique, it was proposed in References [9–11, 29] that the following three preconditioners be used in sequence during the inner iterations of the GMRES search:

$$\mathbf{P}_{\mathrm{L}} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & 0 \\ 0 & \mathrm{DIAG}(\mathbf{A}_{2E2E}) & 0 & 0 \\ 0 & 0 & \mathrm{DIAG}(\mathbf{A}_{2F2F}) & 0 \\ 0 & 0 & 0 & \mathrm{DIAG}(\mathbf{A}_{2I2I}) \end{bmatrix} \tag{31}$$

$$\mathbf{P}_{\mathrm{SETOI}} = \begin{bmatrix} \mathrm{DIAG}(\mathbf{A}_{11}) & 0 & 0 & 0 \\ 0 & \mathbf{A}_{2E2E} & 0 & 0 \\ 0 & \mathbf{A}_{2F2E} & \mathbf{A}_{2F2F} & 0 \\ 0 & \mathbf{A}_{2I2E} & \mathbf{A}_{2I2F} & \mathbf{A}_{2I2I} \end{bmatrix} \tag{32}$$

$$\mathbf{P}_{\mathrm{SITOE}} = \begin{bmatrix} \mathrm{DIAG}(\mathbf{A}_{11}) & 0 & 0 & 0 \\ 0 & \mathbf{A}_{2E2E} & \mathbf{A}_{2E2F} & \mathbf{A}_{2E2I} \\ 0 & 0 & \mathbf{A}_{2F2F} & \mathbf{A}_{2F2I} \\ 0 & 0 & 0 & \mathbf{A}_{2I2I} \end{bmatrix} \tag{33}$$

As possible sequences, ($\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SETOI}}$, $\mathbf{P}_{\mathrm{SITOE}}$, ..., $\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SETOI}}$, $\mathbf{P}_{\mathrm{SITOE}}$), as well as ($\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SETOI}}$, ..., $\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SETOI}}$) and ($\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SITOE}}$, ..., $\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SITOE}}$) were proposed in References [9–11, 29]. As a somewhat downgraded version of $\mathbf{P}_{\mathrm{L}}$, as proposed in References [9–11, 29], we can use a preconditioner that is equivalent to not updating $\mathbf{x}_{2E}$, $\mathbf{x}_{2F}$, and $\mathbf{x}_{2I}$, instead of updating them by using DIAG($\mathbf{A}_{2E2E}$), DIAG($\mathbf{A}_{2F2F}$), and DIAG($\mathbf{A}_{2I2I}$). Similarly, as downgraded versions of $\mathbf{P}_{\mathrm{SETOI}}$ and $\mathbf{P}_{\mathrm{SITOE}}$, we can use preconditioners that are equivalent to not updating $\mathbf{x}_1$, instead of updating it by using DIAG($\mathbf{A}_{11}$).

To differentiate between the two variations of the EDSUM we described in this section, we call the non-linear version, described by Equations (25) and (26), EDSUM-N, and the linear version, described by Equations (27)–(33), EDSUM-L.

Remark 1

It was pointed out in References [10, 11] that the EDSUM provides a natural framework for resourceful and selective application of stabilized formulations to multi-scale computations and subgrid-scale modelling. Along these lines, the 'enhanced-discretization selective stabilization procedure (EDSSP)' was proposed in References [10, 11]. In the EDSSP, finite element equations generating different blocks of the non-linear equation system given by Equation (24) would be based on different stabilized formulations. Level-1 equations (generating the first block) would be based on a stabilized formulation more suitable for flow behaviour at larger scales, and the Level-2 equations (generating the second, third and fourth blocks) would be based on a stabilized formulation more suitable for flow behaviour at smaller scales. As a special version of the EDSSP, it was proposed in References [10, 11] to use with the Level-1 equations only the SUPG and PSPG stabilizations, and with the Level-2 equations use additionally the DCDD stabilization [9, 22, 30, 31].

Remark 2

The EDSUM-B was proposed in References [10, 11] as an approximate version of the EDSUM. In the EDSUM-B, it is proposed to solve the Level-2 equations less frequently than the Level-1 equations. For example, as it was pointed in References [10, 11], instead of performing the same number of iterations to solve all four blocks of Equation (24), we can perform one iteration for the Level-2 blocks for every two iterations we perform for the Level-1 block. This would mean that the small-scale data used in solving the large-scale equations is updated at every other iteration. As another example of EDSUM-B, it was proposed in References [10, 11] to use for the Level-2 equations a more dissipative time-integration algorithm and a time-step size that is twice the time-step size we use for the Level-1 equations. This would mean that the small-scale data used in solving the large-scale equations is updated at every other time step. Although the EDSUM-B is a more approximate technique compared to the EDSUM, it will still be superior in accuracy compared to carrying out the computation with the Level-1 discretization alone.

8. TEST COMPUTATIONS

In this section, we present results obtained from 2D test computations of problems governed by the steady-diffusion and steady-advection equations. The stabilized finite element formulations given by Equations (8) and (11) are used for computations based on standard-discretization

('standard') and enhanced-discretization ('EDSUM'), respectively. For all computations, the domain size is 1.5×1.5 and the discretization has 97×97 nodes. For EDSUM computations the enhanced-discretization zone covers the entire domain, and the 97×97 discretization is achieved by the combination of Mesh-1 with 25×25 nodes and Mesh-2 with 97×97 nodes.

8.1. Performance evaluation of the standard and EDSUM function spaces

To investigate if the EDSUM function space is inherently superior in iterative computations to the standard function space, we perform some comparison tests. To have a fair comparison, we use diagonal preconditioners with both function spaces, and keep the number of inner and outer GMRES iterations the same.

8.1.1. Steady diffusion with uniform boundary profile. This is a boundary-value problem with essential boundary conditions, 0.0 and 1.0, imposed at two opposite sides and homogeneous natural boundary conditions at the other sides. Figure 1 shows the exact solution. We perform one outer and 30 inner GMRES iterations. The solutions obtained with the standard and EDSUM function spaces are shown in Figure 2. Clearly, the EDSUM solution is superior to the standard solution.

8.1.2. Steady diffusion with parabolic boundary profile. In this boundary-value problem, an essential boundary condition with parabolic profile is imposed at one side of the domain, and homogeneous essential boundary conditions at all the other sides. Figure 3 shows the exact solution. One outer and 30 inner GMRES iterations are performed. The solutions obtained

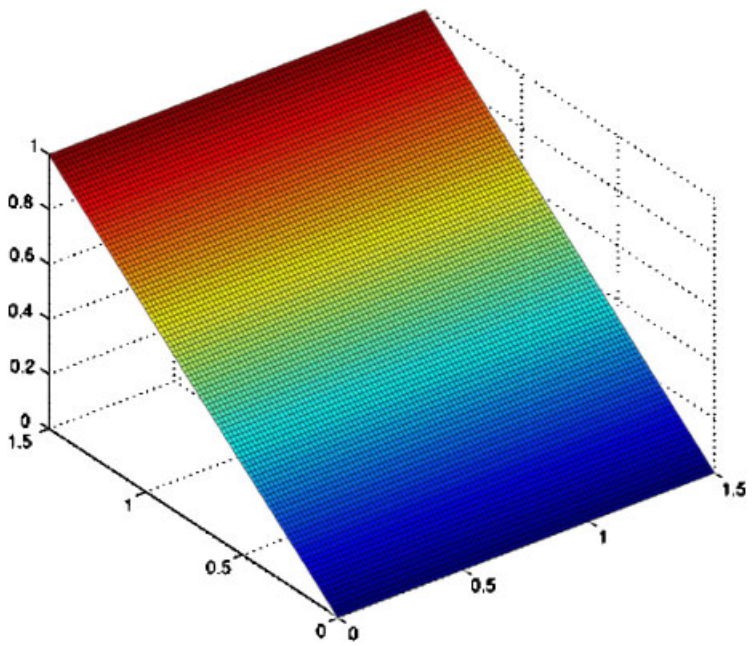

Figure 1. Steady diffusion with uniform boundary profile. Exact solution.

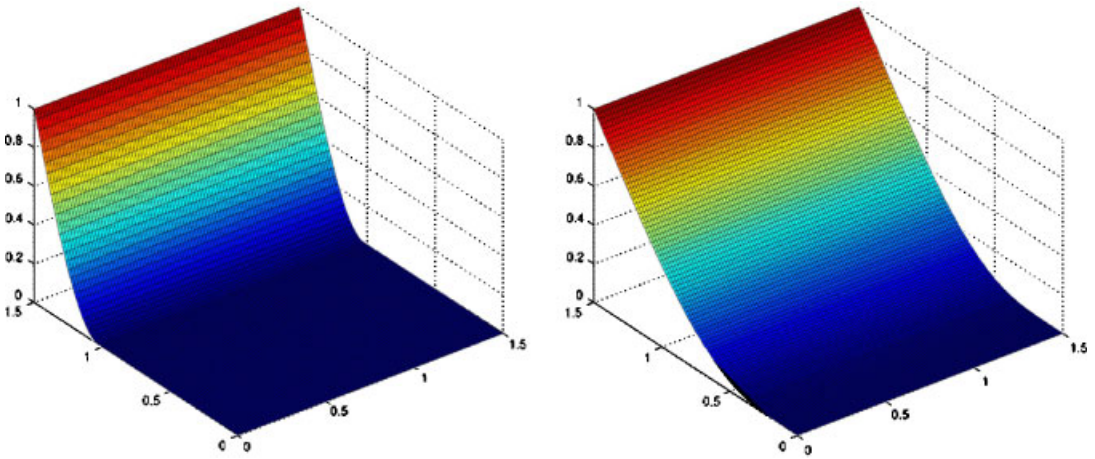

Figure 2. Steady diffusion with uniform boundary profile. Solutions obtained with the standard (left) and EDSUM (right) function spaces.

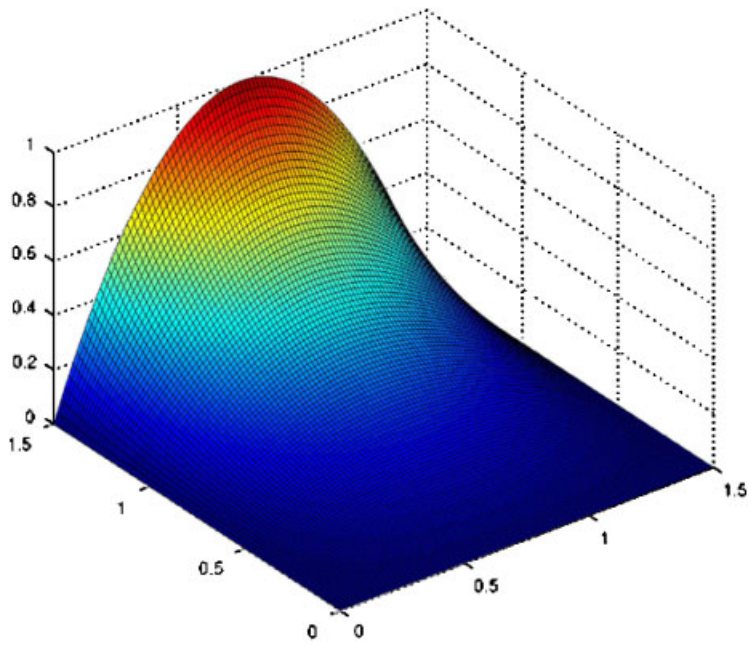

Figure 3. Steady diffusion with parabolic boundary profile. Exact solution.

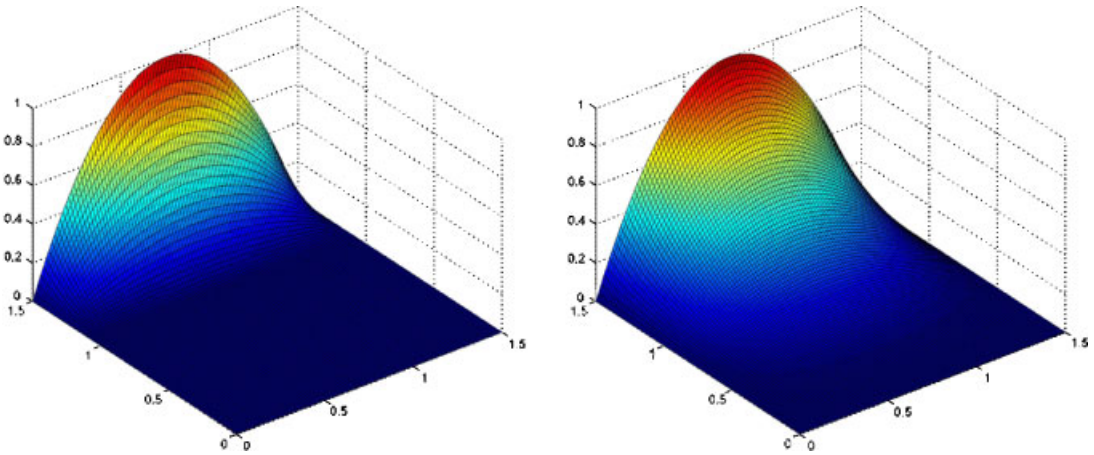

Figure 4. Steady diffusion with parabolic boundary profile. Solutions obtained with the standard (left) and EDSUM (right) function spaces.

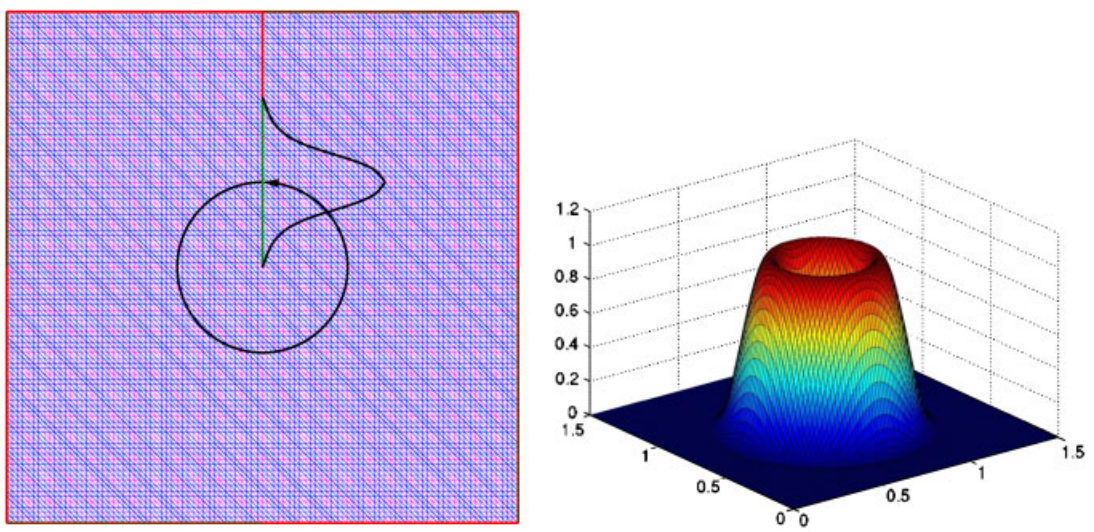

Figure 5. Steady advection. Left: mesh and boundary condition in the form of a cosine hill. Right: exact solution.

with the standard and EDSUM function spaces are shown in Figure 4. We again see that the EDSUM solution is superior to the standard solution.

8.1.3. Steady advection. This is a boundary-value problem with an essential boundary condition in the form of a cosine hill imposed at an internal line (see Figure 5). At each

side of the domain, depending on where the flow is coming in and where it is going out, we impose homogeneous essential boundary condition between the midpoint and one corner and homogeneous natural boundary condition between the midpoint and the other corner. Figure 5 shows also the exact solution. We perform 20 inner GMRES iterations (per outer GMRES iteration), and compare the standard and EDSUM solutions at the end of each outer GMRES iteration, up to a total of 9. The solutions are shown in Figures 6–8. We can clearly see that the EDSUM convergence is superior to the standard convergence.

8.2. Performance evaluations of the EDSUM-L preconditioners

In this subsection, for the EDSUM function space, we comparatively evaluate the performances of some of the EDSUM-L preconditioners described in Section 7. We include in our comparisons the diagonal preconditioner as a starting point, remembering that even with a diagonal preconditioner the EDSUM convergence is already superior to the standard convergence. As test cases, we use the steady diffusion and steady advection problems we considered in Section 8.1. For the two steady diffusion problems we perform one outer and up to 20 inner GMRES iterations, and for the steady advection problem one outer and up to 100 inner GMRES iterations. The EDSUM-L preconditioners tried in test computations are: diagonal, $(\mathbf{P}_{\mathrm{L}})$, $(\mathbf{P}_{\mathrm{L}}, \mathbf{P}_{\mathrm{SETOI}}, \ldots)$, $(\mathbf{P}_{\mathrm{L}}, \mathbf{P}_{\mathrm{SITOE}}, \ldots)$, $(\mathbf{P}_{\mathrm{LTOS}}, \mathbf{P}_{\mathrm{STOL}}, \ldots)$, $(\mathbf{P}_{\mathrm{STOL}})$ and $(\mathbf{P}_{\mathrm{LTOS}})$. Figure 9 shows the convergence histories. All EDSUM-L preconditioners perform far better than the diagonal preconditoner. In the steady diffusion problems, $(\mathbf{P}_{\mathrm{STOL}})$ and $(\mathbf{P}_{\mathrm{LTOS}})$ yield convergence rates that are very comparable to each other but superior to the convergence rates obtained with all the other EDSUM-L preconditioners. In the steady advection problem, again, $(\mathbf{P}_{\mathrm{STOL}})$ and $(\mathbf{P}_{\mathrm{LTOS}})$ convergence rates are comparable to each other but better than the other convergence rates. In this test problem, compared to the diagonal preconditioner, all EDSUM-L preconditioners have dramatically better convergence during the first few iterations.

9. CONCLUDING REMARKS

We described the enhanced-discretization successive update method (EDSUM), a multi-level iteration method designed for being able to compute the flow behaviour at small scales. The EDSUM is a technique based on enhancement in iterative solution of non-linear and linear equation systems. It complements techniques based on enhancement in spatial discretization and based on enhancement in time discretization in the context of a space–time formulation. It is very much related to the enhanced-discretization interface-capturing technique (EDICT). The EDSUM and EDICT are both based on using a two-level function space for the trial and test functions in the finite element formulation. The trial and test functions have two components each, one coming from a first-level mesh (Mesh-1) and the second one coming from a second-level, finer mesh (Mesh-2) constructed over the first-level mesh. In the EDSUM, the large-scale flow information is represented by the interpolation functions associated with Mesh-1, and the small-scale flow information by the interpolation functions associated with Mesh-2. Nodes are classified into Mesh-1 nodes, and Mesh-2 edge, face and interior nodes.

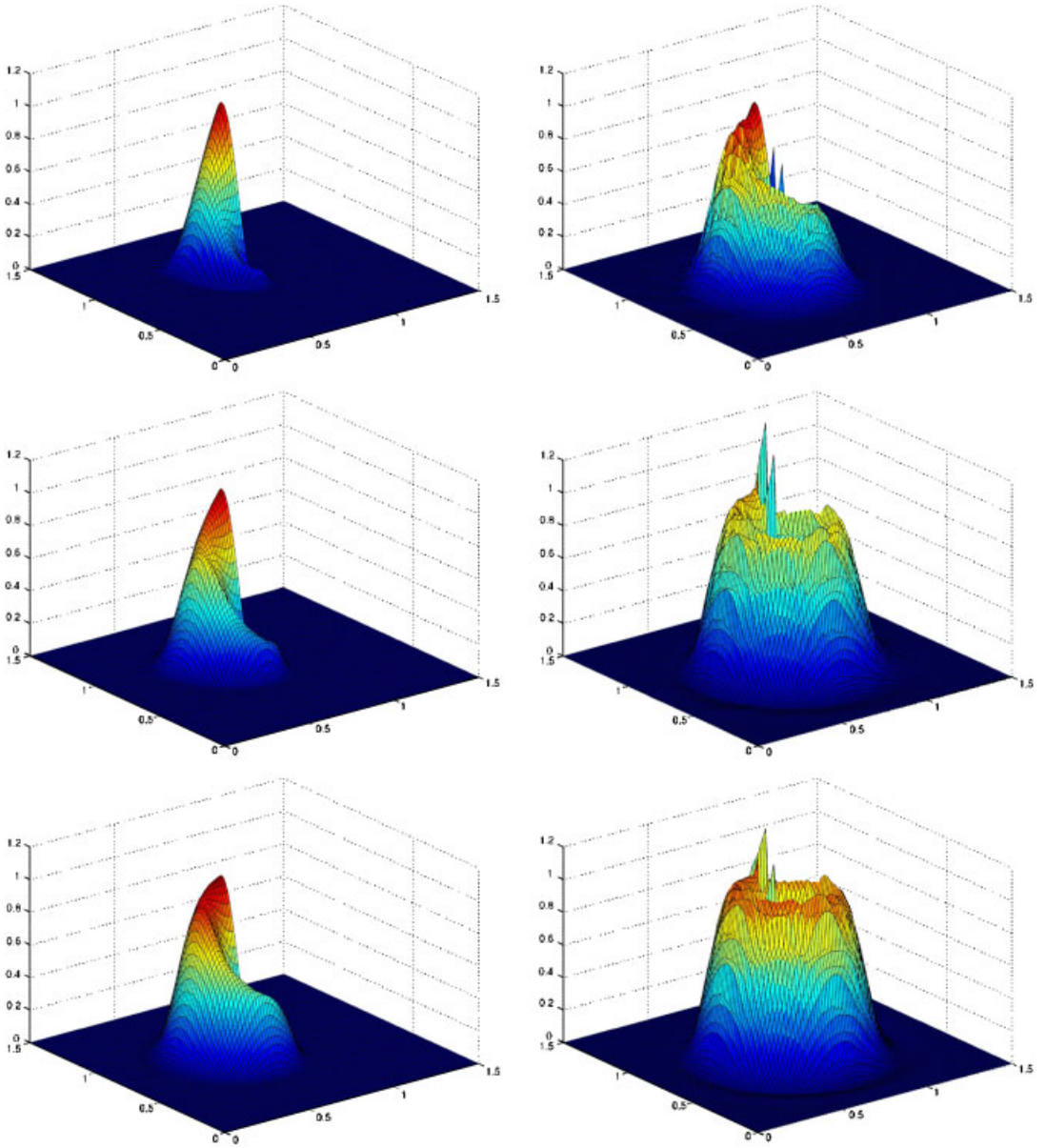

Figure 6. Steady advection. Solutions obtained with the standard (left) and EDSUM (right) function spaces, after 1 (top), 2 (middle), and 3 (bottom) outer GMRES iterations.

In the EDSUM-N, the non-linear equation system is solved by an approximate Newton–Raphson method, where at each non-linear iteration step we successively update the solution vectors corresponding to each class of nodes. While updating each class, we use the most

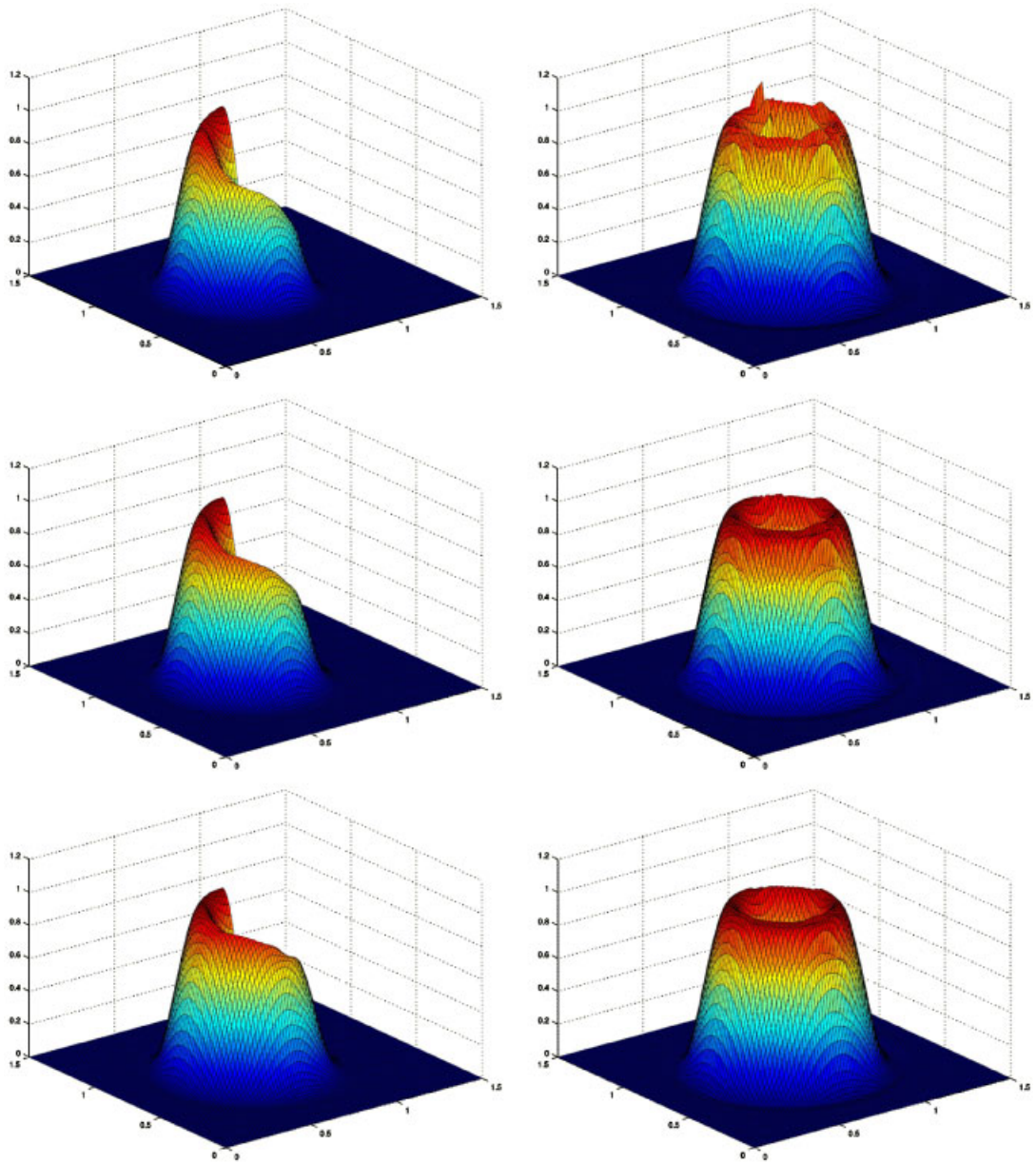

Figure 7. Steady advection. Solutions obtained with the standard (left) and EDSUM (right) function spaces, after 4 (top), 5 (middle), and 6 (bottom) outer GMRES iterations.

recent values of the solution vectors in calculating the non-linear vector functions and their derivatives with respect to the solution vectors. In EDSUM-L, the linear equation system that needs to be solved at every step of a (full) Newton–Raphson method is solved iteratively with

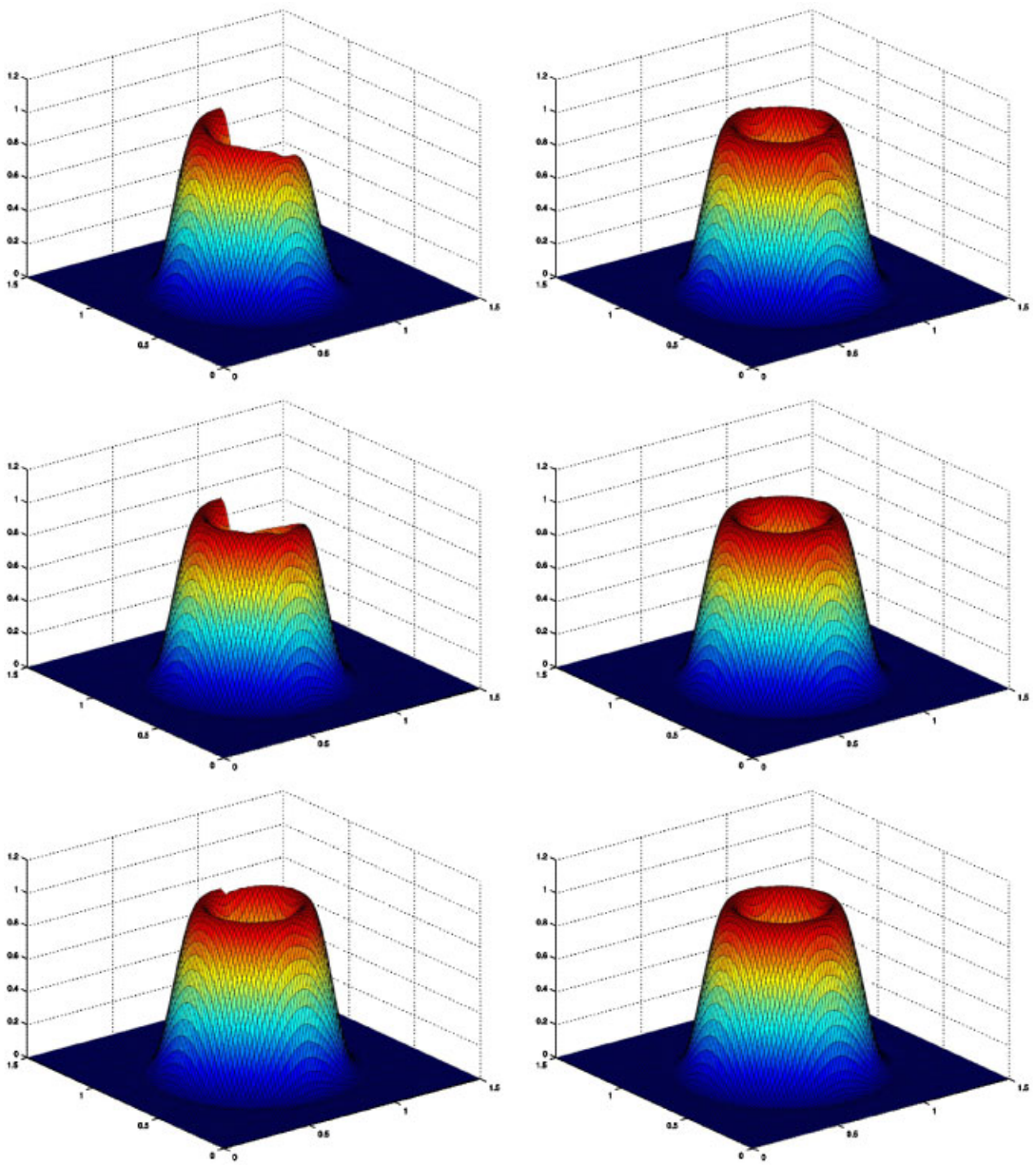

Figure 8. Steady advection. Solutions obtained with the standard (left) and EDSUM (right) function spaces, after 7 (top), 8 (middle), and 9 (bottom) outer GMRES iterations.

preconditioners designed based on the same successive update concept. In computations based on stabilized formulations, during the successive updates, the projection of flow information between the large and small scales is consistent with the discretizations resulting from the

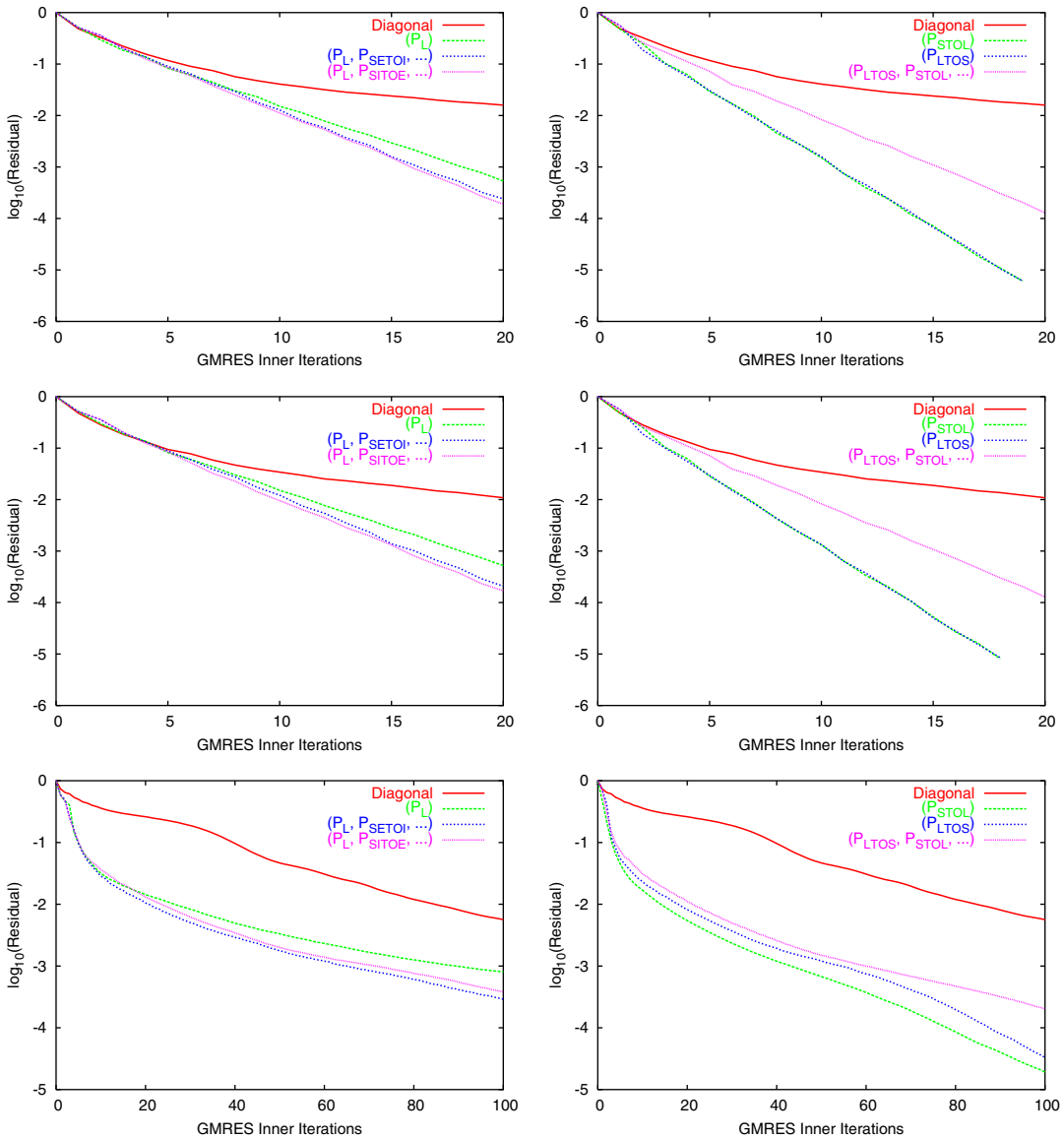

Figure 9. Performance evaluations of EDSUM-L preconditioners. Left: ($\mathbf{P}_{\mathrm{L}}$), ($\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SETOI}}$,...), and ($\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{SITOE}}$,...). Right: ($\mathbf{P}_{\mathrm{LTOS}}$, $\mathbf{P}_{\mathrm{STOL}}$,...), ($\mathbf{P}_{\mathrm{STOL}}$), and ($\mathbf{P}_{\mathrm{LTOS}}$). Test problems: steady diffusion with uniform boundary profile (top), steady diffusion with parabolic boundary profile (middle), and steady advection (bottom).

stabilized formulations. We presented a number of test computations for steady-state problems governed by the advection–diffusion equation to show how the EDSUM works and how it accelerates the convergence of the iterative solutions.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Army Natick Soldier Centre and NASA Johnson Space Centre.

REFERENCES

1. Hughes TJR, Brooks AN. A multi-dimensional upwind scheme with no crosswind diffusion. In *Finite Element Methods for Convection Dominated Flows*, Hughes TJR (ed.), AMD-Vol. 34. ASME: New York, 1979; 19–35.
2. Brooks AN, Hughes TJR. Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**:199–259.
3. Tezduyar TE, Hughes TJR. Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations. *Proceedings of AIAA 21st Aerospace Sciences Meeting*, *AIAA Paper 83-0125*, Reno, Nevada, 1983.
4. Hughes TJR, Tezduyar TE. Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations. *Computer Methods in Applied Mechanics and Engineering* 1984; **45**:217–284.
5. Tezduyar TE. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics* 1991; **28**:1–44. ⟵ 1992 (correct publication year)
6. Tezduyar TE. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering* 2001; **8**:83–130.
7. Tezduyar T. Finite element interface-tracking and interface-capturing techniques for flows with moving boundaries and interfaces. *Proceedings of the ASME Symposium on Fluid-Physics and Heat Transfer for Macro- and Micro-Scale Gas–Liquid and Phase-Change Flows* (*CD-ROM*), *ASME Paper IMECE2001/HTD-24206*. ASME: New York, 2001.
8. Tezduyar T. Interface-tracking and interface-capturing techniques for computation of moving boundaries and interfaces. *Proceedings of the Fifth World Congress on Computational Mechanics*, http://wccm.tuwien.ac.at/, *Paper-ID*: *81513*, Vienna, Austria, 2002.
9. Tezduyar TE. Finite element methods for fluid dynamics with moving boundaries and interfaces. In *Encyclopedia of Computational Mechanics*, *Volume 3*: *Fluids*, Stein E, De Borst R, Hughes TJR (eds), Chapter 17. Wiley: New York, 2004.
10. Tezduyar TE. Stabilized finite element methods for flows with moving boundaries and interfaces. *HERMIS International Journal* 2004; **4**.
11. Tezduyar TE. Moving boundaries and interfaces. In *Finite Element Methods*: 1970's *and Beyond*, Franca LP, Tezduyar TE, Masud A (eds). CIMNE: Barcelona, Spain, 2004; 205–220.
12. Hughes TJR, Franca LP, Balestra M. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška–Brezzi condition: a stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**:85–99.
13. Tezduyar TE, Behr M, Liou J. A new strategy for finite element computations involving moving boundaries and interfaces—the deforming-spatial-domain/space–time procedure: I. The concept and the preliminary tests. *Computer Methods in Applied Mechanics and Engineering* 1992; **94**(3):339–351.
14. Tezduyar TE, Behr M, Mittal S, Liou J. A new strategy for finite element computations involving moving boundaries and interfaces—the deforming-spatial-domain/space–time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Computer Methods in Applied Mechanics and Engineering* 1992; **94**(3):353–371.
15. Tezduyar TE, Behr M, Mittal S, Johnson AA. Computation of unsteady incompressible flows with the finite element methods—space–time formulations, iterative strategies and massively parallel implementations. *New Methods in Transient Analysis*, PVP-Vol. 246/AMD-Vol. 143. ASME: New York, 1992; 7–24.
16. Hughes TJR, Hulbert GM. Space–time finite element methods for elastodynamics: formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering* 1988; **66**:339–363.
17. Tezduyar T, Aliabadi S, Behr M. Enhanced-discretization interface-capturing technique (EDICT) for computation of unsteady flows with interfaces. *Computer Methods in Applied Mechanics and Engineering* 1998; **155**:235–248.
18. Tezduyar TE, Sathe S. Enhanced-discretization space–time technique (EDSTT). *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:1385–1401.
19. Tezduyar TE. Interface-tracking, interface-capturing and enhanced solution techniques. *Proceedings of the First South-American Congress on Computational Mechanics* (*CD-ROM*), Santa Fe-Parana, Argentina, 2002.
20. Tezduyar TE, Sathe S. Enhanced-approximation linear solution technique (EALST). *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:2033–2049.

21. Tezduyar TE, Osawa Y. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:411–430.
22. Tezduyar T. Stabilization parameters and local length scales in SUPG and PSPG formulations. *Proceedings of the Fifth World Congress on Computational Mechanics*, On-line publication: http://wccm.tuwien.ac.at/, *Paper-ID*: *81508*, Vienna, Austria, 2002.
23. Saad Y, Schultz M. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing* 1986; **7**:856–869.
24. Tezduyar T, Aliabadi S, Behr M, Johnson A, Kalro V, Litke M. High performance computing techniques for flow simulations. In *Solving Large-Scale Problems in Mechanics*: *Parallel Solution Methods in Computational Mechanics*, Papadrakakis M (ed.), Chapter 10. Wiley: New York, 1997; 363–398.
25. Kalro V, Tezduyar T. Parallel iterative computational methods for 3D finite element flow simulations. *Computer Assisted Mechanics and Engineering Sciences* 1998; **5**:173–183.
26. Johan Z, Hughes TJR, Shakib F. A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids. *Computer Methods in Applied Mechanics and Engineering* 1991; **87**:281–304.
27. Johan Z, Mathur KK, Johnsson SL, Hughes TJR. A case study in parallel computation: viscous flow around an Onera M6 wing. *International Journal for Numerical Methods in Fluids* 1995; **21**:877–884.
28. Tezduyar TE, Behr M, Aliabadi SK, Mittal S, Ray SE. A new mixed preconditioning method for finite element computations. *Computer Methods in Applied Mechanics and Engineering* 1992; **99**:27–42.
29. Tezduyar TE. *Stabilized Finite Element Methods for Computation of Flows with Moving Boundaries and Interfaces*. Lecture Notes on Finite Element Simulation of Flow Problems (Basic–Advanced Course). Japan Society of Computational Engineering and Sciences: Tokyo, Japan, 2003.
30. Tezduyar TE. Adaptive determination of the finite element stabilization parameters. *Proceedings of the ECCOMAS Computational Fluid Dynamics Conference 2001* (*CD-ROM*), Swansea, Wales, United Kingdom, 2001.
31. Tezduyar TE. Computation of moving boundaries and interfaces and stabilization parameters. *International Journal for Numerical Methods in Fluids* 2003; **43**:555–575.

