Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 193 (2004) 2033–2049
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Enhanced-approximation linear solution technique (EALST)

Tayfun E. Tezduyar *, Sunil Sathe

Team for Advanced Flow Simulation and Modeling (TAFSM), Mechanical Engineering, Rice University-MS 321, 6100 Main Street, Houston, TX 77005, USA

Received 20 March 2003; received in revised form 10 September 2003; accepted 8 December 2003

Abstract

The enhanced discretization and solution techniques are among the advanced computational methods we rely on in simulation and modeling of complex flow problems, including those with moving boundaries and interfaces. The set of enhanced discretization and solution techniques includes those based on enhancement in spatial discretization, enhancement in time discretization, and enhancement in iterative solution of nonlinear and linear equation systems. The enhanced-approximation linear solution technique (EALST) was introduced to increase the performance of the iterative technique used in solution of the linear equation systems when some parts of the computational domain may offer more of a challenge for the iterative method than the others. The EALST can be used for computations based on semi-discrete or space–time formulations.

Keywords: Flow simulation; Enhanced discretization and solution techniques; Linear equation systems; Iterative solution techniques; Enhanced-approximation

1. Introduction

In simulation and modeling of complex flow problems, including those with moving boundaries and interfaces, we rely on a number of advanced computational methods. These methods include the stabilized finite element techniques such as the streamline-upwind/Petrov–Galerkin (SUPG) [1,2] and pressure-stabilizing/Petrov–Galerkin (PSPG) [3] formulations, interface-tracking and interface-capturing techniques (see [3–6]), and the enhanced discretization and solution techniques (see [4–9]). An earlier version of the pressure-stabilizing formulation for Stokes flows was reported in [10].

The stabilized formulations prevent numerical oscillations and other instabilities in solving problems with high Reynolds and/or Mach numbers and shocks and strong boundary layers, as well as when using equal-order interpolation functions for velocity and pressure and other unknowns. Furthermore, this class of stabilized formulations substantially improve the convergence rate in iterative solution of the large,

* Corresponding author. Tel.: +1-713-348-6051; fax: +1-713-348-5423.
E-mail address: tezduyar@rice.edu (T.E. Tezduyar).
URL: http://www.mems.rice.edu/TAFSM/.

0045-7825/$ - see front matter
doi:10.1016/j.cma.2003.12.045

coupled nonlinear equation system that needs to be solved at every time step of a flow computation. Such nonlinear systems are typically solved with the Newton–Raphson method, which involves, at its every iteration step, solution of a large, coupled linear equation system. It is in iterative solution of such linear equation systems that using a good stabilized method makes substantial difference in convergence, and this was pointed out in [11].

The deforming-spatial-domain/stabilized space–time (DSD/SST) formulation [3], developed for moving boundaries and interfaces, is an interface-tracking technique, where the finite element formulation of the problem is written over its space–time domain. At each time step the locations of the interfaces are calculated as part of the overall solution. 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 [12] (where the motions of the nodes are governed by the equations of elasticity) and full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) as needed. We note that the stabilized space–time formulations were used earlier by other researchers to solve problems with fixed spatial domains (see for example [13]).

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. The enhanced-discretization interface-capturing technique (EDICT) was introduced in [7] to increase accuracy in representing an interface. In the EDICT, we start with the basic approach of an interface-capturing technique such as the volume of fluid (VOF) method [14]. The Navier–Stokes equations are solved over a nonmoving mesh together with the time-dependent advection equation governing the evolution of an interface function marking the location of the interface. In the stabilized formulations of these two sets of equations to be solved, to increase the accuracy, we use function spaces corresponding to 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 will all have two components each: one coming from Mesh-1 and the second one coming from Mesh-2. To further increase the accuracy, we construct a third-level mesh, Mesh-3, 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 will have three components, each coming from one of these three meshes. We re-define the subsets over which we build Mesh-2 and Mesh-3 not every time step but with sufficient frequency to keep the interface enveloped in. We need to avoid this envelope being too wide or too narrow.

We note that, functionally, the EDICT would enable us have the benefits one would draw from adaptive local mesh refinement. The EDICT would allow us accomplish this objective without facing the implementational difficulties associated with elements having variable number of nodes. Furthermore, it would allow us accomplish this enhanced-spatial-discretization objective even when the required increase in mesh refinement is large and locally abrupt. The EDICT was followed by a number of extensions and offshoots. These are based on other forms of enhancement in spatial discretization, enhancement in time discretization, and enhancement in iterative solution of nonlinear and linear equation systems.

An offshoot based on enhancement in spatial discretization, for computation of compressible flows with shocks, was reported in [8]. In this extension, the "interface" becomes the shock front. At and near the shock fronts, we use enhanced discretization to increase the accuracy in representing those shocks. More accurate representation of the shock is accomplished while avoiding local mesh refinement involving elements with variable number of nodes or global mesh refinement involving large increases in computational cost. Another offshoot based on enhancement in spatial discretization, for computation of vortex flows, was reported in [9]. In this case the spatial discretization is enhanced where the vorticity magnitude is larger than a specified value.

The enhanced-discretization space–time technique (EDSTT), which was introduced in [5,6], is an offshoot based on enhancement in temporal 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 we require smaller time steps in certain parts of the fluid domain. For example, in computation of two-fluid interfaces with the DSD/SST method, time-integration of the equation governing the evolution of the interface (i.e. the interface update equation) may require a smaller time step than the one used for the fluid interiors. This requirement might be coming from numerical stability considerations, when time-integration of the interface update equation does not involve any added stabilization terms.

It can promptly be reasoned that the set of enhanced discretization and solution techniques should also include offshoots based on enhancements in iterative solution of nonlinear and linear equation systems. The enhanced-iteration nonlinear solution technique (EINST) [5,6] and the enhanced-approximation linear solution technique (EALST) [5,6] were introduced as natural complements to the offshoots based on 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 nonlinear 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.

In Section 2, we review the governing equations, and describe the stabilized semi-discrete and space–time formulations in Sections 3 and 4. In Section 5, we provide a brief overview of the iterative methods used for solution of nonlinear and linear equation systems. The EINST and EALST are described in Section 6, and numerical tests based on the EALST are reported in Section 7. Concluding remarks are provided in Section 8.

2. Governing equations

Let $\Omega_t \subset \mathbb{R}^{n_{sd}}$ be the spatial fluid mechanics domain with boundary Γ_t at time $t \in (0,T)$, where the subscript t indicates the time-dependence of the spatial domain. The Navier–Stokes equations of incompressible flows can be written on Ω_t 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:

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

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

$$\mathbf{u} = \mathbf{g} \quad \text{on } (\Gamma_t)_g, \quad \mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h} \quad \text{on } (\Gamma_t)_h, \tag{5}$$

where $(\Gamma_t)_g$ and $(\Gamma_t)_h$ are complementary subsets of the boundary Γ_t, $\mathbf{n}$ 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. If the spatial domain does not need to change with respect to time, the subscript t can be dropped from Ω_t and Γ_t.

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

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

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 Eq. (6) are represented as

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

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

3. Stabilized semi-discrete formulations

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

$$\begin{aligned} &\int_\Omega w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) \mathrm{d}\Omega + \int_\Omega \nabla w^h \cdot \nu \nabla \phi^h \, \mathrm{d}\Omega - \int_{\Gamma_h} w^h h^h \, \mathrm{d}\Gamma \\ &\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_{\mathrm{SUPG}} \mathbf{u}^h \cdot \nabla w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h - \nabla \cdot \left(\nu \nabla \phi^h \right) \right) \mathrm{d}\Omega = 0. \end{aligned} \tag{8}$$

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 [15,16].

For the Navier–Stokes equations of incompressible flows, given by Eqs. (1) and (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 $q^h \in \mathscr{V}_p^h$:

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

where

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

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 [15,16].

4. DSD/SST finite element formulation

In the DSD/SST method [3], the finite element formulation of the governing equations is written over a sequence of N space–time slabs Q_n, where Q_n is the slice of the space–time domain between the time levels t_n and t_{n+1}. At each time step, the integrations involved in the finite element formulation are performed over Q_n. The space–time finite element interpolation functions are continuous within a space–time slab, but discontinuous from one space–time slab to another. The notation $(\cdot)_n^-$ and $(\cdot)_n^+$ denotes the function values at t_n as approached from below and above. Each Q_n is decomposed into elements Q_n^e, where $e = 1, 2, \ldots, (n_{\mathrm{el}})_n$. The subscript n used with n_{el} is for the general case in which the number of space–time elements may change from one space–time slab to another. The Dirichlet- and Neumann-type boundary conditions are enforced over $(P_n)_g$ and $(P_n)_h$, the complementary subsets of the lateral boundary of the space–time slab. The finite element trial function spaces $(\mathscr{S}_{\mathbf{u}}^h)_n$ for velocity and $(\mathscr{S}_p^h)_n$ for pressure, and the test function spaces $(\mathscr{V}_{\mathbf{u}}^h)_n$ and $(\mathscr{V}_p^h)_n = (\mathscr{S}_p^h)_n$ are defined by using, over Q_n, first-order polynomials in both space and time. The DSD/SST formulation [3,16] is written as follows: given $(\mathbf{u}^h)_n^-$, find $\mathbf{u}^h \in (\mathscr{S}_{\mathbf{u}}^h)$ and $p^h \in (\mathscr{S}_p^h)_n$ such that $\forall \mathbf{w}^h \in (\mathscr{V}_{\mathbf{u}}^h)_n$ and $q^h \in (\mathscr{V}_p^h)_n$:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h\right) \mathrm{d}Q + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\, \mathrm{d}Q - \int_{(P_n)_h} \mathbf{w}^h \cdot \mathbf{h}^h \, \mathrm{d}P \\
&\quad + \int_{Q_n} q^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho\big((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^-\big) \, \mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \frac{1}{\rho}\left[\tau_{\mathrm{SUPG}} \rho\left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h\right) + \tau_{\mathrm{PSPG}} \nabla q^h\right] \cdot \big[\text{Ł}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h\big] \mathrm{d}Q \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q = 0.
\end{aligned} \tag{11}$$

This formulation is applied to all space–time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, starting with $(\mathbf{u}^h)_0^- = \mathbf{u}_0$. For an earlier, detailed reference on the DSD/SST formulation see [3]. The DSD/SST formulation of Eq. (6) can be written in a similar fashion.

5. Iterative solution methods

The finite element formulations described in the earlier sections fall into two categories: a space–time formulation with moving meshes or a semi-discrete formulation with nonmoving meshes. Full discretizations of these formulations lead to coupled, nonlinear equation systems that need to be solved at every time step of the simulation. Whether we are using a space–time formulation or a semi-discrete formulation, we can represent the equation system that needs to be solved as follows:

$$\mathbf{N}(\mathbf{d}_{n+1}) = \mathbf{F}. \tag{12}$$

Here $\mathbf{d}_{n+1}$ is the vector of nodal unknowns. In a semi-discrete formulation, this vector contains the unknowns associated with marching from time level n to $n+1$. In a space–time formulation, it contains the unknowns associated with the finite element formulation written for the space–time slab Q_n. The time-marching formulations described earlier can also be used for computing a steady-state flow. In such cases time does not have a physical significance, but is only used in time-marching to the steady-state solution.

We solve Eq. (12) with the Newton–Raphson method:

$$\left.\frac{\partial \mathbf{N}}{\partial \mathbf{d}}\right|_{\mathbf{d}^i_{n+1}} \left(\Delta \mathbf{d}^i_{n+1}\right) = \mathbf{F} - \mathbf{N}\left(\mathbf{d}^i_{n+1}\right), \tag{13}$$

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 represented by Eq. (13) needs to be solved at every step of the Newton–Raphson sequence. We can represent Eq. (13) as a linear equation system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{14}$$

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. At each iteration, we need to compute the residual of this system:

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}. \tag{15}$$

This can be achieved in several different ways. The computation can be based on a sparse-matrix storage of **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. After the residual computation, we compute a candidate correction to **x** as given by the expression

$$\Delta \mathbf{y} = \mathbf{P}^{-1}\mathbf{r}, \tag{16}$$

where **P**, the preconditioning matrix, is an approximation to **A**. **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. How to update the solution vector **x** by using $\Delta \mathbf{y}$ is also a major subject in iterative solution techniques. Several update methods are available, and we use the GMRES [17] method. We have been focusing our research related to iterative methods mainly on computing the residual **r** efficiently and selecting a good preconditioner **P**. While moving in this direction, 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, the ''parallel-ready'' methods we designed for the residual computations include those that are element-matrix-based [18] element-vector-based [18], and sparse-matrix-based [19]. The element-vector-based methods were successfully used also by other researchers in the context of parallel computations (see for example [20,21]).

In preconditioning design, we developed some advanced preconditioners such as the clustered-element-by-element (CEBE) preconditioner [22] and the mixed CEBE and cluster companion (CC) preconditioner [22]. We have implemented, with quite satisfactory results, the CEBE preconditioner in conjunction with an ILU approximation [19]. However, our typical computations are based on diagonal and nodal-block-diagonal preconditioners. These are very simple preconditioners, but are also very simple to implement on parallel platforms. More on our parallel implementations can be found in [18].

6. Enhanced solution techniques

Sometimes, some parts of the computational domain may offer more of a challenge for the Newton–Raphson method than the others. This might happen, for example, at the fluid–solid interface in a fluid–structure interaction problem, and in such cases the nonlinear convergence might become even a bigger challenge if the structure is going through buckling or wrinkling. It might also happen at a fluid–fluid interface, for example, if the interface is very unsteady. In the EINST [5,6] as a variation of the Newton–Raphson method, we propose to use sub-iterations in the parts of the domain where we are facing a nonlinear convergence challenge. This could be implemented, for example, by identifying the nodes of the zones where we need enhanced iterations, and performing multiple iterations for those nodes for each

iteration we perform for all other nodes. Or, at every time step, we can let those nodes have a "head start" of several iterations prior to commencing iterations for all other nodes. In time-accurate computations of fluid–structure interactions with the EDSTT, the EINST can be used to allow for a larger number of nonlinear iterations for the structure.

In some challenging cases, using a diagonal or nodal-block-diagonal preconditioners might not lead to a satisfactory level of convergence at some locations, in the parts of the domain posing the challenge. This might happen, for example, in a fluid–structure interaction problem, where the structure or the fluid zones near the structure might be suffering from convergence problems. The situation might become worse if the structure is going through buckling or wrinkling. It might also happen at a fluid–fluid interface. In the EALST [5,6], we propose to use stronger approximations for the parts of the domain where we are facing convergence challenges. This could be implemented, for example, by identifying the elements covering the zones where we need enhanced approximation, and reflecting this in defining the element-level constituents of the approximation matrix. For example, for the elements that need stronger approximations, we can use as the element-level approximation matrix the full element-level matrix, while for all other elements we use a diagonal element-level matrix. This particular EALST can be summarized by first expressing the assembly process for $\mathbf{A}$ and $\mathbf{P}$ as

$$\mathbf{A} = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \mathbf{A}^e, \tag{17}$$

$$\mathbf{P} = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \mathbf{P}^e, \tag{18}$$

where A is the finite element assembly operator, and then defining $\mathbf{P}^e$ for the elements belonging to the enhanced-approximation and diagonal-approximation groups:

$$\mathbf{P}^e = \begin{cases} \mathbf{A}^e & \text{for Enhanced Approximation Group,} \\ \mathrm{DIAG}(\mathbf{A}^e) & \text{for Diagonal Approximation Group.} \end{cases} \tag{19}$$

Here DIAG represents a diagonal or nodal-block-diagonal approximation operator. We note that in factorizing the sub-matrices of $\mathbf{P}$ corresponding to the enhanced-approximation group, we can use a direct solution method, or, as an alternative, a second-level iteration sequence. This second-level iteration sequence would have its own preconditioner (possibly a diagonal or nodal-block-diagonal preconditioner) and its own GMRES vector space (possibly shorter than the GMRES vector space used in the first-level iterations). To differentiate between these two versions of the EALST, we will call them EALST-D and EALST-I.

In EINST and EALST, elements can be selected to the enhanced group (enhanced-iteration group in EINST and enhanced-approximation group in EALST) in a static or dynamic way. In the static way, the elements would be selected to the enhanced group based on what we know in advance about the flow problem. Elements in the parts of the mesh that are expected to offer more of a challenge during a computation would belong to the enhanced group. For example, in a fluid–structure interaction computation, a thin layer of fluid elements around the structure could be defined as the enhanced group. In the dynamic way, elements would be selected to the enhanced group based on identifying the nodes with the highest normalized residuals or lowest residual reduction rates. For example, elements connected to the nodes with the lowest 10% residual reduction rates could be defined as the enhanced group. The residuals being examined are those for the nonlinear equation system in EINST and the linear equation system in EALST. In the dynamic way, the enhanced group would not be re-defined every time step. They would be re-defined frequently enough so that in between re-defining the enhanced group we can expect to maintain substantial

overlap between the elements in the enhanced group and the elements connected to the nodes with the lowest residual reduction rates.

7. Test computations

In this section we present results from 2D test computations for problems governed by the advection–diffusion equation and the Navier–Stokes equations of incompressible flows. The stabilized finite element formulation used is given by Eq. (8) for the advection–diffusion equation and by Eq. (11) for the Navier–Stokes equations. We compare the solutions obtained with EALST-D and EALST-I with the solutions obtained with a diagonal preconditioner. For all test computations with the advection–diffusion equation, the domain size is 1.5×1.5, and the finite element mesh is uniform, consists of triangular elements, and has 61×61 nodes. For the advection–diffusion test computations, the enhanced-approximation group is identified in a static way for the steady cases and in a dynamic way for the unsteady ones. In the dynamic way, elements connected to the nodes with the highest 10% of the initial residuals (in iterative solution of the linear version of Eq. (12)) are selected to the enhanced-approximation group. In advection–diffusion tests, when the problem involves a cosine profile or hill, the corresponding dimensionless wave number ($q = kh$), defined based on the nodal spacing (h), is 0.3142.

7.1. Steady diffusion

The finite element mesh, the enhanced-approximation zone, and a sketch of the exact solution are shown in Fig. 1. The nodal residual distributions for the linear equation system at the end of the computations with the diagonal preconditioner, EALST-D, and EALST-I are shown in Fig. 2. In all three cases, the number of outer and inner GMRES iterations are 1 and 75. For the EALST-I, the number of outer and inner GMRES iterations for the second-level iteration sequence associated with the enhanced-approximation zone are 1 and 20. The EALST results in substantially reduced nodal residuals in the enhanced-approximation zone.

7.2. Steady advection

The finite element mesh, the enhanced-approximation zone, and a sketch of the exact solution are shown in Fig. 3. The flow field is rotational and counter-clock-wise. Along an internal line, the value of ϕ is specified to be a cosine profile. The nodal residual distributions for the linear equation system at the end of the computations with the diagonal preconditioner, EALST-D, and EALST-I are shown in Fig. 4. In all

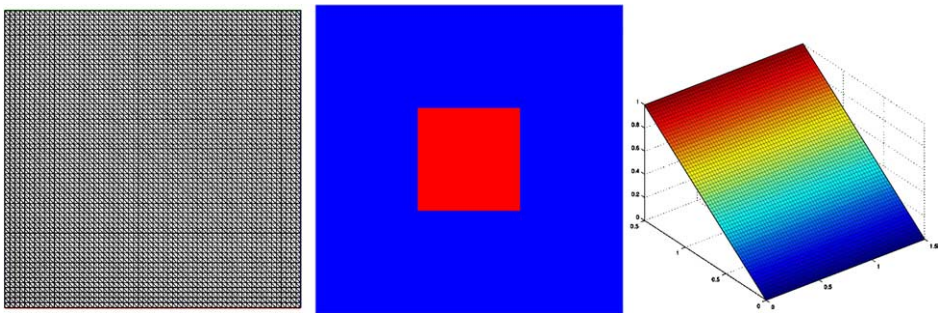

Fig. 1. Steady diffusion. Finite element mesh (left), enhanced-approximation zone (middle), and sketch of the exact solution (right).

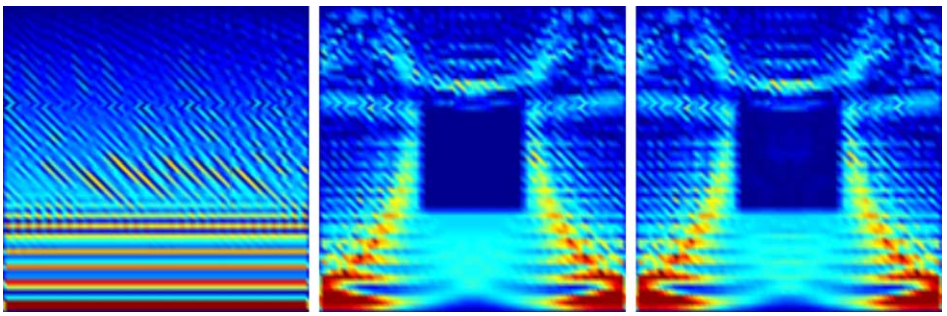

Fig. 2. Steady diffusion. Nodal residual distributions for the linear equation system at the end of the computations with the diagonal preconditioner (left), EALST-D (middle), and EALST-I (right).

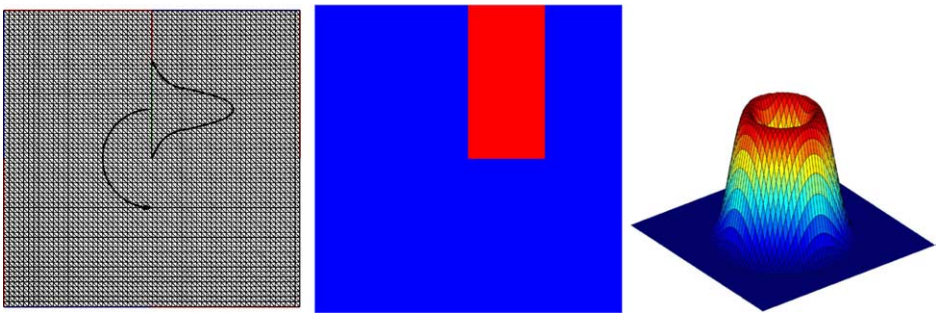

Fig. 3. Steady advection. Finite element mesh (left), enhanced-approximation zone (middle), and sketch of the exact solution. The mesh picture also shows that the rotational flow field is counter-clock-wise and the value of ϕ is specified to be a cosine profile at an internal line.

three cases, the number of outer and inner GMRES iterations are 1 and 110. For the EALST-I, the number of outer and inner GMRES iterations for the second-level iteration sequence associated with the enhanced-approximation zone are 1 and 20. The EALST results in substantially reduced nodal residuals in the enhanced-approximation zone.

7.3. Unsteady diffusion

The finite element mesh, the initial condition in the form of a cosine hill, and a sketch of the exact solution after 10 time steps are shown in Fig. 5. The diffusivity is set to 1.0 and the time-step size to 0.01. We computed this test problem with the diagonal preconditioner, EALST-D, and EALST-I. In all three cases, the number of outer and inner GMRES iterations per time step are 1 and 20. For the EALST-I, the number of outer and inner GMRES iterations for the second-level iteration sequence associated with the enhanced-approximation zone are 1 and 20. At three evaluation instants during the computations with the diagonal preconditioner, EALST-D, and EALST-I, we compare the nodal residual distributions for the linear equation system at the end of the iterations. For the computation with the diagonal preconditioner, the nodal residual distributions at the three evaluation instants are shown in Fig. 6. For the computations with the EALST-D and EALST-I, the enhanced-approximation zones and nodal residual distributions at the three evaluation instants are shown in Figs. 7 and 8. We can see that in computations based on the EALST, the nodal residuals in the enhanced-approximation zones are significantly reduced.

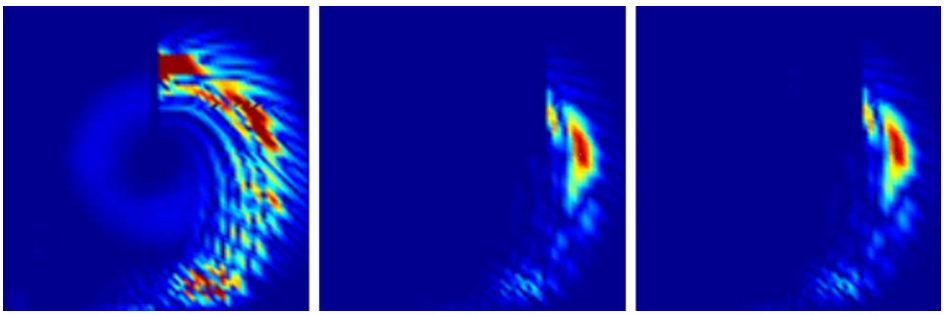

Fig. 4. Steady advection. Nodal residual distributions for the linear equation system at the end of the computations with the diagonal preconditioner (left), EALST-D (middle), and EALST-I (right).

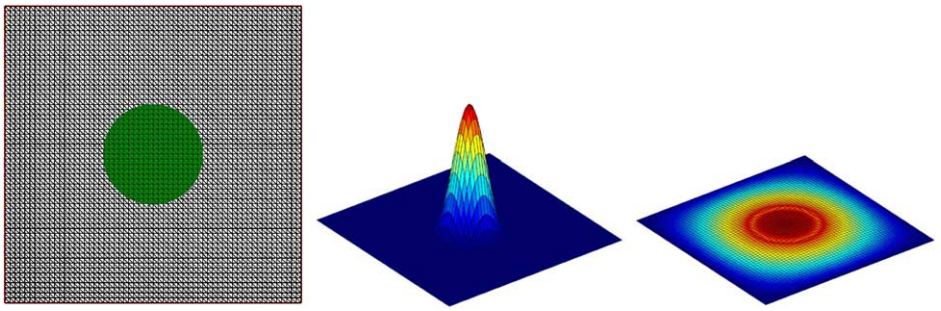

Fig. 5. Unsteady diffusion. Finite element mesh (left), initial condition in the form of a cosine hill (middle), and sketch of the exact solution after 10 time steps. The mesh picture also shows the footprint of the cosine hill.

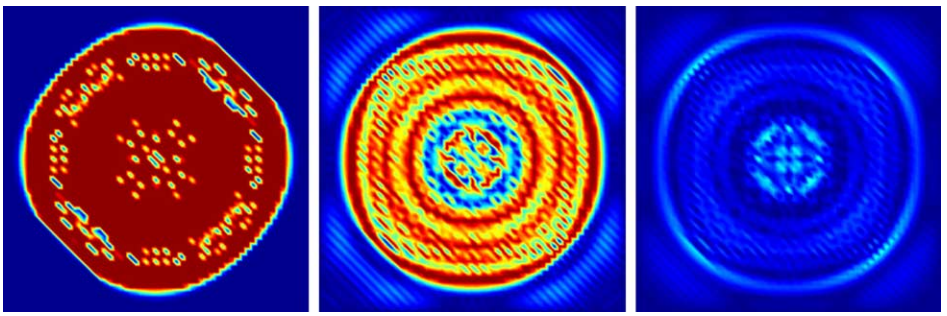

Fig. 6. Unsteady diffusion. Computation with the diagonal preconditioner. Nodal residual distributions for the linear equation system (at the end of the iterations) at the three evaluation instants.

7.4. Unsteady advection

The initial condition in this test problem is also a cosine hill. The flow field is rotational and counter-clock-wise. The finite element mesh, together with the footprint of the cosine hill, and a sketch of the exact solution are shown in Fig. 9. The advection-velocity magnitude is 1.0 at the footprint-center of the cosine hill, and the time-step size is set to 0.025. We computed this test problem with the diagonal preconditioner,

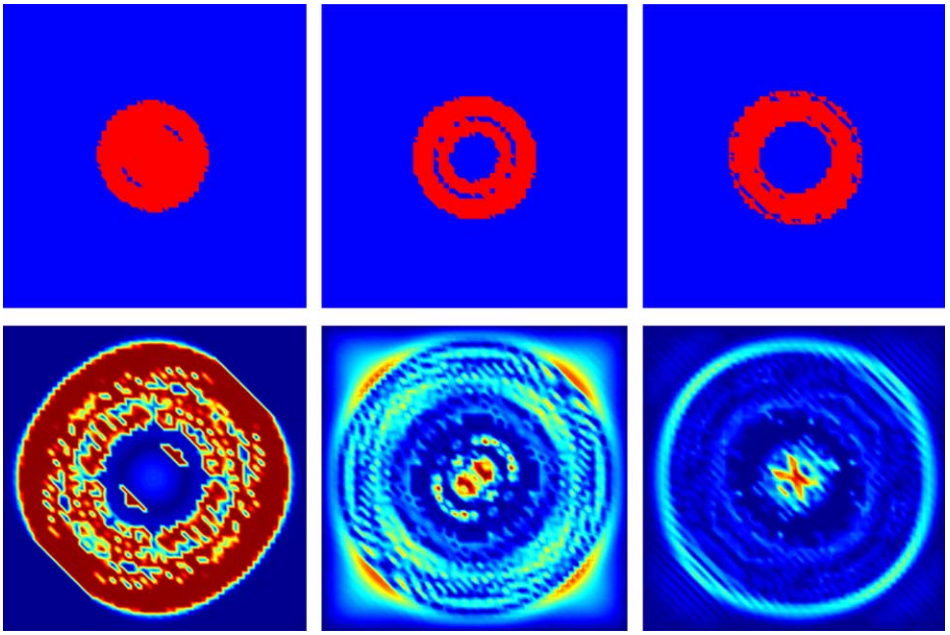

Fig. 7. Unsteady diffusion. Computation with the EALST-D. Enhanced-approximation zones (top) and nodal residual distributions for the linear equation system (at the end of the iterations) (bottom) at the three evaluation instants.

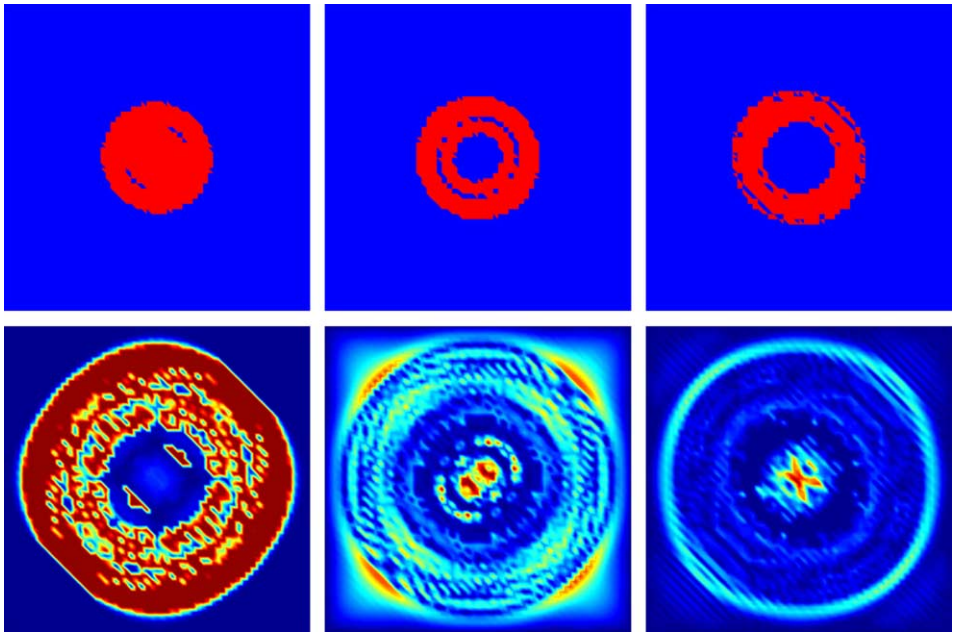

Fig. 8. Unsteady diffusion. Computation with the EALST-I. Enhanced-approximation zones (top) and nodal residual distributions for the linear equation system (at the end of the iterations) (bottom) at the three evaluation instants.

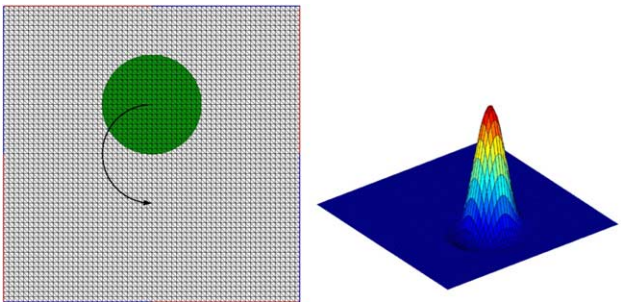

Fig. 9. Unsteady advection. Finite element mesh (left) and a sketch of the exact solution (right). The mesh picture also shows the footprint of the initial condition in the form of a cosine hill.

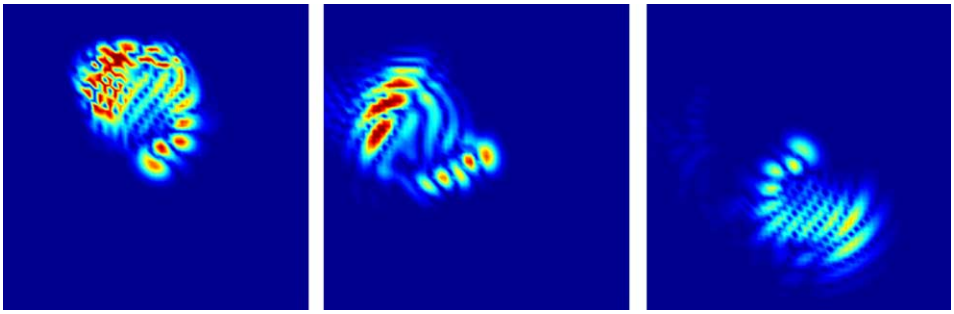

Fig. 10. Unsteady advection. Computation with the diagonal preconditioner. Nodal residual distributions for the linear equation system (at the end of the iterations) at the three evaluation instants.

EALST-D, and EALST-I. In all three cases, the number of outer and inner GMRES iterations per time step are 1 and 7. For the EALST-I, the number of outer and inner GMRES iterations for the second-level iteration sequence associated with the enhanced-approximation zone are 1 and 10. At three evaluation instants during the computations with the diagonal preconditioner, EALST-D, and EALST-I, we compare the nodal residual distributions for the linear equation system at the end of the iterations. For the computation with the diagonal preconditioner, the nodal residual distributions at the three evaluation instants are shown in Fig. 10. For the computations with the EALST-D and EALST-I, the enhanced-approximation zones and nodal residual distributions at the three evaluation instants are shown in Figs. 11 and 12. Again, we can see that in computations based on the EALST, the nodal residuals in the enhanced-approximation zones are significantly reduced.

7.5. Incompressible flow

In this test problem we compute flow past a thin rigid beam. The problem set up is shown in Fig. 13. The fluid density and viscosity are 1000 kg/m^3 and 1.792×10^{-3} N s/m^2. The flow boundary conditions are set to a uniform inflow velocity of 1.0 m/s at the upstream boundary, traction-free conditions at the downstream, slip conditions at the lateral boundaries, and no slip conditions at the beam. The finite element mesh consists of 2770 nodes and 5396 triangular elements. The enhanced-approximation zone is defined in a static fashion and is shown in Fig. 14. The number of elements in the enhanced-approximation zone is 672. The time-step size is set, based on Courant number considerations, to 0.20 s. Fig. 15 shows the flow

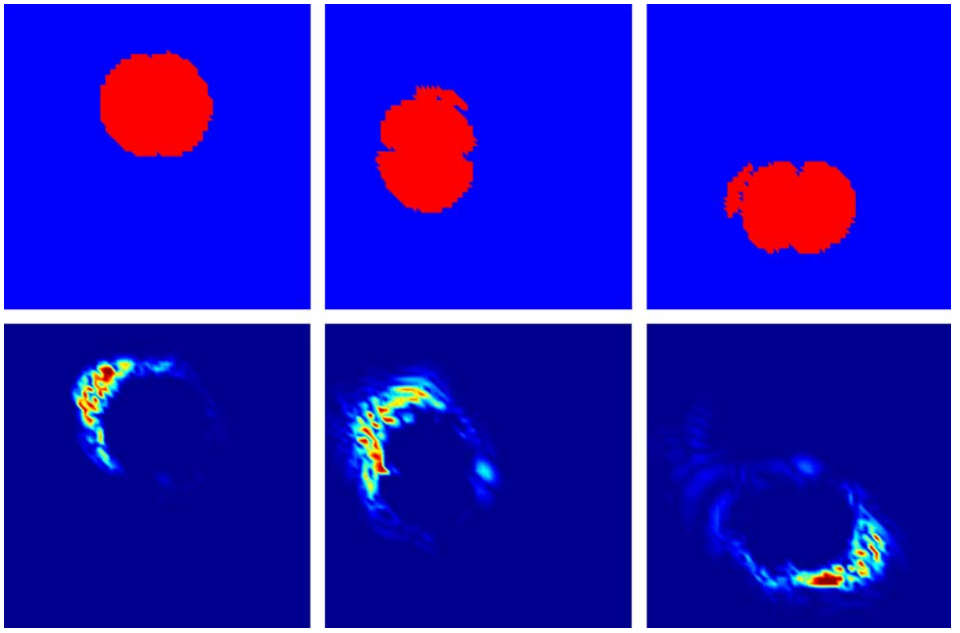

Fig. 11. Unsteady advection. Computation with the EALST-D. Enhanced-approximation zones (top) and nodal residual distributions for the linear equation system (at the end of the iterations) (bottom) at the three evaluation instants.

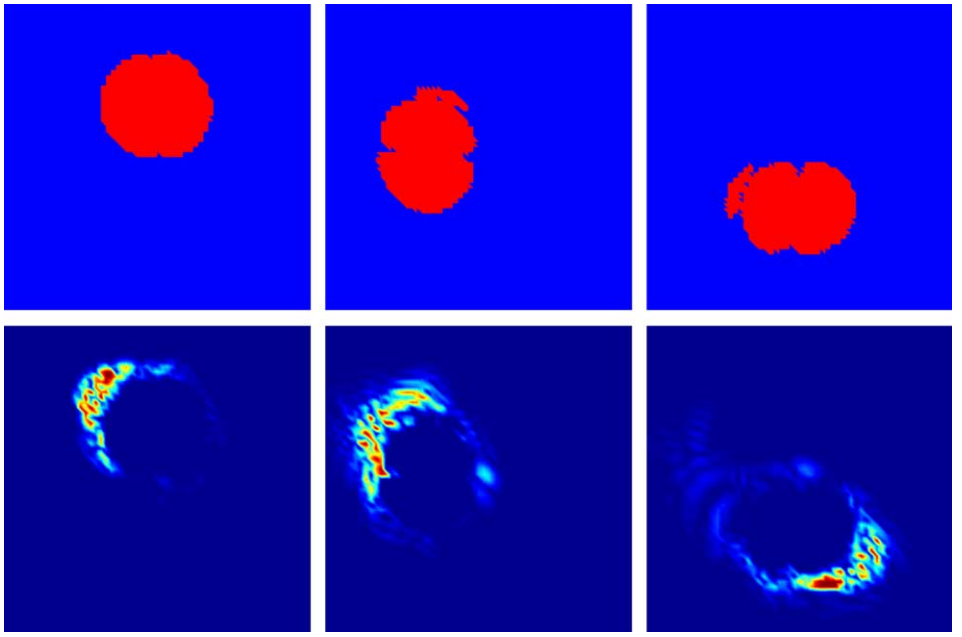

Fig. 12. Unsteady advection. Computation with the EALST-I. Enhanced-approximation zones (top) and nodal residual distributions for the linear equation system (at the end of the iterations) (bottom) at the three evaluation instants.

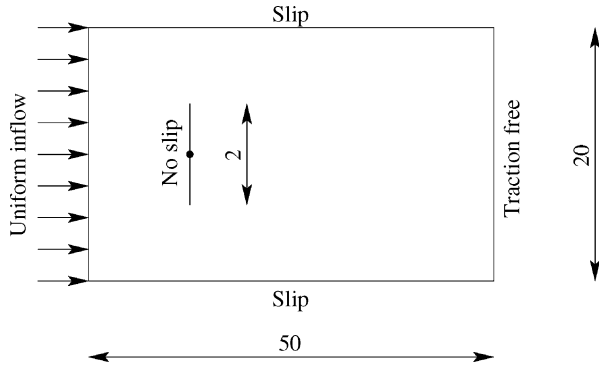

Fig. 13. Incompressible flow. Problem set up for flow past a thin rigid beam. The dimensions indicated are in meters. Flow boundary conditions are indicated next to each boundary.

Fig. 14. Incompressible flow. Enhanced-approximation zone (approximately 12% of the elements).

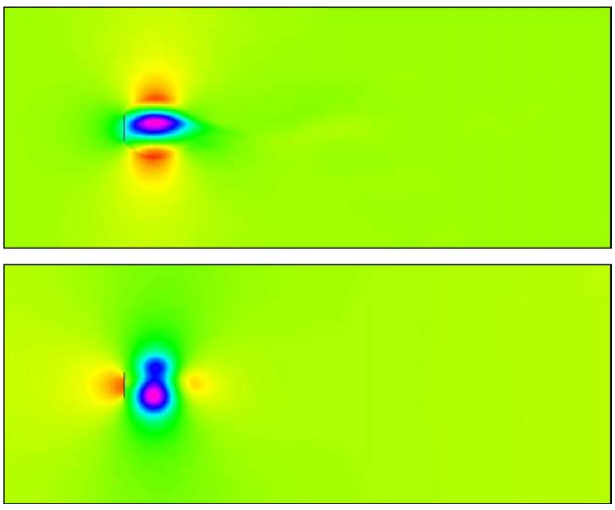

Fig. 15. Incompressible flow. Flow horizontal-velocity (top) and pressure (bottom) at start of computation.

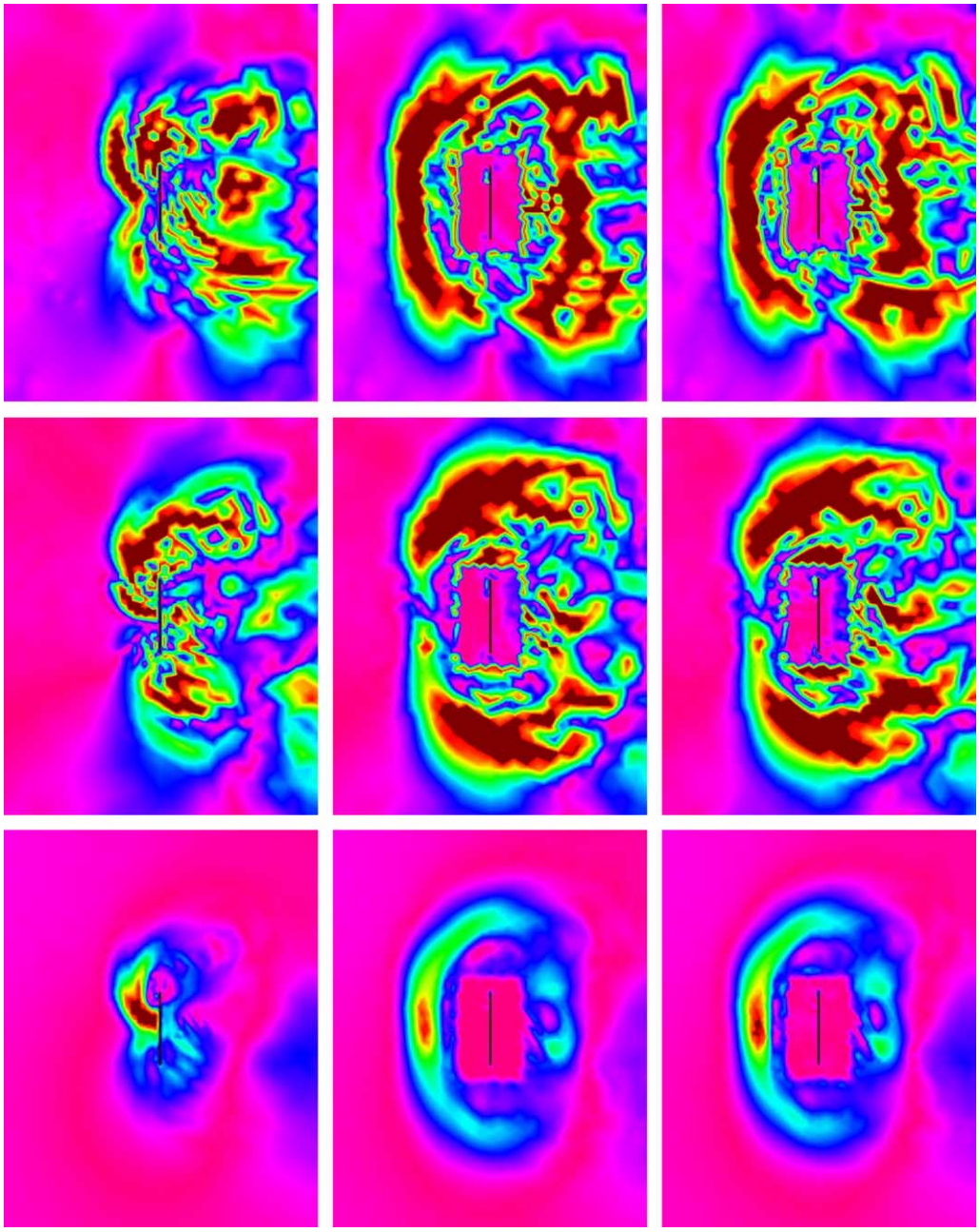

Fig. 16. Incompressible flow. Computations with the diagonal preconditioner (left), EALST-D (middle), and EALST-I (right). Nodal residual distributions for the linear equation system (at the end of the iterations) for the first nonlinear iteration in the first time step. Nodal residual distributions associated with the horizontal-momentum equation (top), vertical-momentum equation (middle), and the incompressibility constraint (bottom).

horizontal-velocity and pressure at the start of the computations. For the first nonlinear iteration in the first time step of the computations with the diagonal preconditioner, EALST-D, and EALST-I, we compare the

nodal residual distributions for the linear equation system at the end of the iterations. The comparisons, based on the nodal residual distributions associated with the horizontal-momentum equation, vertical-momentum equation, and the incompressibility constraint, are shown in Fig. 16. We can clearly see that in the enhanced-approximation zones the nodal residuals are significantly reduced for the computations based on the EALST. We also see in this test, and in a number of tests reported in prior subsections, that, compared to the diagonal preconditioner, the EALST generates some increase in nodal residuals at a few locations outside the enhanced-approximation zone. We believe that this is because the solution update is based on minimizing the norm of the nodal residual vector, and therefore when the nodal residuals are reduced dramatically in the enhanced-approximation zones, they might somewhat increase in a few other regions outside.

8. Concluding remarks

We described the EALST, which was developed to increase the performance of the iterative technique used in solution of the linear equation systems when some parts of the computational domain may offer more of a challenge for the iterative method than the others. The EALST is one of the techniques in the set of enhanced discretization and solution techniques, which includes techniques 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, together with the stabilized finite element formulations and interface-tracking and interface-capturing techniques, help us address some of the major computational challenges involved in simulation and modeling of complex flow problems, including those with moving boundaries and interfaces. With the EALST, in iterative solution of a linear equation system, the approximation matrix (preconditioning matrix) is enhanced, beyond being a diagonal-approximation, to an approximation that more "directly" represents the parts of the domain where we might be facing convergence challenges. The nondiagonal sub-matrices corresponding to the enhanced-approximation zones can be solved with direct solution method (EALST-D), or, as an alternative, a second-level iteration sequence (EALST-I). The enhanced-approximation zones can be identified in a static or dynamic way. In the static way, these zones are identified based on what we know in advance about the flow problem. Elements in the parts of the mesh that are expected to offer more of a challenge during a computation would belong to the enhanced-approximation zones. In the dynamic way, elements would be selected to the enhanced-approximation zones by identifying the nodes with the highest residuals or lowest residual reduction rates. For example, elements connected to the nodes with the highest 10% residuals could be selected to the enhanced-approximation zones.

We presented a number of test computations to demonstrate how the EALST works and how it might help us deal with the parts of the computational domain creating more of a challenge for the iterative method than the other parts.

Acknowledgements

This work was supported by the US Army Natick Soldier Center and NASA Johnson Space Center.

References

[1] T.J.R. Hughes, A.N. Brooks, A multi-dimensional upwind scheme with no crosswind diffusion, in: T.J.R. Hughes (Ed.), Finite Element Methods for Convection Dominated Flows, AMD-Vol. 34, ASME, New York, 1979, pp. 19–35.

[2] T.E. Tezduyar, T.J.R. Hughes, Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations, in: Proceedings of AIAA 21st Aerospace Sciences Meeting, AIAA Paper 83-0125, Reno, Nevada, 1983.
[3] T.E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Adv. Appl. Mech. 28 (1991) 1–44. (1992)
[4] T.E. Tezduyar, Finite element methods for flow problems with moving boundaries and interfaces, Arch. Comput. Methods Engrg. 8 (2001) 83–130.
[5] T. Tezduyar, Finite element interface-tracking and interface-capturing techniques for flows with moving boundaries and interfaces, in: 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.
[6] T. Tezduyar, Interface-tracking and interface-capturing techniques for computation of moving boundaries and interfaces, in: Proceedings of the Fifth World Congress on Computational Mechanics, On-line publication: http://wccm.tuwien.ac.at/, Paper-ID: 81513, Vienna, Austria, 2002.
[7] T. Tezduyar, S. Aliabadi, M. Behr, Enhanced-discretization interface-capturing technique (EDICT) for computation of unsteady flows with interfaces, Comput. Methods Appl. Mech. Engrg. 155 (1998) 235–248.
[8] S. Mittal, S. Aliabadi, T. Tezduyar, Parallel computation of unsteady compressible flows with the EDICT, Comput. Mech. 23 (1999) 151–157.
[9] T. Tezduyar, Y. Osawa, K. Stein, R. Benney, V. Kumar, J. McCune, Computational methods for parachute aerodynamics, in: M. Hafez, K. Morinishi, J. Periaux (Eds.), Computational Fluid Dynamics for the 21st Century, Springer, 2001.
[10] T.J.R. Hughes, L.P. Franca, M. Balestra, 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, Comput. Methods Appl. Mech. Engrg. 59 (1986) 85–99.
[11] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, S. Mittal, Parallel finite-element computation of 3D flows, IEEE Comput. 26 (10) (1993) 27–36.
[12] T.E. Tezduyar, M. Behr, S. Mittal, A.A. Johnson, Computation of unsteady incompressible flows with the finite element methods—space–time formulations, iterative strategies and massively parallel implementations, in: New Methods in Transient Analysis, PVP-Vol. 246/AMD-Vol. 143, ASME, New York, 1992, pp. 7–24.
[13] T.J.R. Hughes, G.M. Hulbert, Space–time finite element methods for elastodynamics: formulations and error estimates, Comput. Methods Appl. Mech. Engrg. 66 (1988) 339–363.
[14] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[15] T.E. Tezduyar, Y. Osawa, Finite element stabilization parameters computed from element matrices and vectors, Comput. Methods Appl. Mech. Engrg. 190 (2000) 411–430.
[16] T. Tezduyar, Stabilization parameters and local length scales in SUPG and PSPG formulations, in: Proceedings of the Fifth World Congress on Computational Mechanics, On-line publication: http://wccm.tuwien.ac.at/, Paper-ID: 81508, Vienna, Austria, 2002.
[17] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[18] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, V. Kalro, M. Litke, High performance computing techniques for flow simulations, in: M. Papadrakakis (Ed.), Solving Large-Scale Problems in Mechanics: Parallel Solution Methods in Computational Mechanics, John Wiley & Sons, 1996, pp. 363–398.
[19] V. Kalro, T. Tezduyar, Parallel iterative computational methods for 3D finite element flow simulations, Comput. Assist. Mech. Engrg. Sci. 5 (1998) 173–183.
[20] Z. Johan, T.J.R. Hughes, F. Shakib, A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids, Comput. Methods Appl. Mech. Engrg. 87 (1991) 281–304.
[21] Z. Johan, K.K. Mathur, S.L. Johnsson, T.J.R. Hughes, A case study in parallel computation: viscous flow around an Onera M6 wing, Int. J. Numer. Methods Fluids 21 (1995) 877–884.
[22] T.E. Tezduyar, M. Behr, S.K. Aliabadi, S. Mittal, S.E. Ray, A new mixed preconditioning method for finite element computations, Comput. Methods Appl. Mech. Engrg. 99 (1992) 27–42.