INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **57**:1421–1434
Published online 2 June 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1796

Simplex space–time meshes in finite element simulations

Marek Behr*, †

Chair for Computational Analysis of Technical Systems (CATS), Center for Computational Engineering Science (CCES), RWTH Aachen University, 52056 Aachen, Germany

SUMMARY

A straightforward method for generating simplex space–time meshes is presented, allowing arbitrary temporal refinement in selected portions of space–time slabs. The method increases the flexibility of space–time discretizations, even in the absence of dedicated space–time mesh generation tools. The resulting tetrahedral (for 2D problems) and pentatope (for 3D problems) meshes are tested in the context of advection–diffusion equation, and are shown to provide reasonable solutions, while enabling varying time refinement in portions of the domain. Copyright © 2008 John Wiley & Sons, Ltd.

Received 12 October 2007; Revised 7 February 2008; Accepted 8 February 2008

KEY WORDS: space–time finite elements; advection–diffusion equation; pentatope discretization

1. INTRODUCTION

Finite element (FE) method has been typically applied to time-dependent problems in conjunction with a finite difference (FD) temporal discretization, such as θ-family or Runge–Kutta family of methods. Regardless of the sequence of discretization steps, i.e. whether the FE method in space is followed by an FD in time (method of lines or semi-discretization) or this order is reversed (method of Rothe), one utilizes FE interpolation, with its flexibility to admit completely unstructured meshes with varying levels of refinement, purely on a spatial domain.

1.1. Space–time discretization

The space–time approach applies the FE method to the space–time domain in a single discretization step. The idea can be traced to the works by Jamet and Bonnerot [1, 2], Lynch and Gray [3],

*Correspondence to: Marek Behr, Chair for Computational Analysis of Technical Systems (CATS), Center for Computational Engineering Science (CCES), RWTH Aachen University, 52056 Aachen, Germany.
†E-mail: behr@cats.rwth-aachen.de

Contract/grant sponsor: German Science Foundation; contract/grant numbers: GSC 111, EXC 128, SFB 540, SFB 401, SPP 1253, SPP 1273

Copyright © 2008 John Wiley & Sons, Ltd.

and Frederiksen and Watts [4]; Jamet [5] recognized discontinuous-in-time interpolation as an advantageous approach for a model parabolic problem. This concept was then developed further for multi-dimensional advective–diffusive systems [6, 7], elastodynamics [8], Navier–Stokes equations [9, 10], and Navier–Stokes problems involving deforming domains [11–13]. Since those early gestation phases, the space–time approach has been steadily gaining new understanding and new applications. Yet, it can be argued that the full potential of the space–time FEs has been only partially exploited. The extraordinary flexibility of FEs when dealing with varying resolution in complicated domains is not yet commonly harnessed when the time dimension is considered. Although novel space–time meshing concepts appear occasionally (and are listed below), the difficulties ingrained in perceiving and computationally analyzing higher-dimensional geometric entities limit somewhat these concepts' appeal in realistic 3D applications.

In order to avoid unmanageable number of degrees of freedom that must be solved for at any given time, the space–time approach is typically applied to subsets of the temporal domain called space–time slabs, which are roughly comparable with time steps in the semi-discrete approach.[‡] Moreover, in most space–time implementations to date, the meshes for the space–time slabs are simply extruded in the temporal direction from a spatial mesh, resulting in reference element domains that are always Cartesian products of spatial and temporal domains. Such an approach is best described as semi-unstructured (unstructured in space, structured in time) and does not leave the option of increasing temporal refinement in portions of the domain. In such a case, the space–time slab exactly corresponds to a time step of a semi-discrete procedure. In fact, many stencils of semi-discrete methods may be re-derived by using the semi-unstructured space–time approach with appropriate weighting and interpolation functions.

1.2. Unstructured space–time meshes for 1D and 2D problems

Hughes and Hulbert [8] introduced the idea of adaptively refined space–time mesh for a 1D elastic rod problem and discussed the potential of the unstructured space–time meshes as a more flexible and rigorous alternative to subcycling. Maubach in his thesis [15] applied and analyzed unstructured space–time meshes for 2D advection–diffusion equations. Idesman *et al.* [16] obtained deformation history of 2D viscoelastic plate by using an adaptively refined space–time mesh. A similar approach, dubbed 'single mesh', was used by Sathe [17] to solve 2D advection–diffusion problems.

1.3. Hierarchical space–time meshes for 3D problems

Sathe [17] notes the 4D mesh generation difficulties inherent in extending the 'single mesh' approach to 3D problems and proposes a 'multiple mesh' alternative, where structured nested grids, presumably as simple to generate in 4D as in 3D, are superimposed over the standard semi-unstructured mesh. That approach is then used to solve a 2D fluid–structure interaction problem.

In the following sections, we first describe a robust and simple procedure to generate 3D or 4D simplex-based space–time meshes in Section 2. Other aspects of implementation of fully unstructured space–time discretization are discussed in Section 3. As a numerical example, we

[‡]Note that the recent advent of highly scalable parallel computing architectures, and the associated need for parallelism, motivates *increasing* the coupling between available degrees of freedom, by means of simulating larger portions of the temporal domain in a single step [14].

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

examine 1D advection of a Gaussian hill, solved in 2D and 3D with varying levels of temporal refinement in Section 4. This is followed by comments on performance in Section 5 and by concluding remarks and directions of future studies in Section 6.

2. SPACE–TIME MESH GENERATION

The key to space–time computations that allow varying degrees of temporal, as well as spatial, refinement is a straightforward and robust algorithm for the generation of simplex-based space–time meshes. The necessary steps are outlined in the subsections that follow. For simplicity, we restrict ourselves to just linear continuous interpolation functions for all variables, although this aspect is not crucial to the proposed approach.

2.1. Spatial mesh

As a starting point, we consider a spatial mesh in n_{sd} dimensions, generated using any of the freely or commercially available mesh generators. We restrict ourselves to n_{sd}-simplex-based meshes, such as triangular meshes in 2D and tetrahedral meshes in 3D. The Delaunay approach used here to generate space–time connectivity always produces simplex-type elements; therefore, a generalization of the proposed algorithm to quadrilateral or hexahedral spatial meshes is not attempted.

2.2. Prism formation

Similar to that in traditional space–time implementation, the spatial mesh is at first extruded in the time dimension to fill the space–time slab contained between time levels t_n and t_{n+1}. The extruded mesh is composed of prisms—6-noded space–time elements for 2D problems and 8-noded space–time elements for 3D problems. These elements, referred to as 3d6n and 4d8n, respectively, and illustrated in Figure 1(a), are the basis of the traditional space–time approach considered here as a reference. Note that the 'faces' extending in the temporal dimension are non-simplical, having 4 nodes for the 3d6n element and 6 nodes for the 4d8n element. We will aim at subdividing these prism-type elements into simplex-type elements 3d4n (familiar tetrahedrons) and 4d5n (referred to as pentatopes), as shown in Figure 1(b).

2.3. Temporal refinement

The initial space–time mesh contains only two nodes for each of the nodes in the spatial mesh—e.g. in1 located at the bottom of the slab and in2 located at the top of the slab, as shown in Figure 2(a). The temporal refinement is accomplished by adding, in parts of the domain where temporal accuracy is to be increased, one or more nodes along the line connecting the original nodes, as shown in Figure 2(b).

2.4. Coordinate perturbation

The space–time faces of the prism-type elements will be subsequently divided into $(n_{\mathrm{sd}}-1)$-node simplices according to the Delaunay criteria, *independently* for each prism. As (in the case of stationary spatial mesh) these faces are regular (rectangles for 2D and *right* prisms for 3D), the Delaunay mesh will not be unique. Therefore, there is no guarantee that the direction of the

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

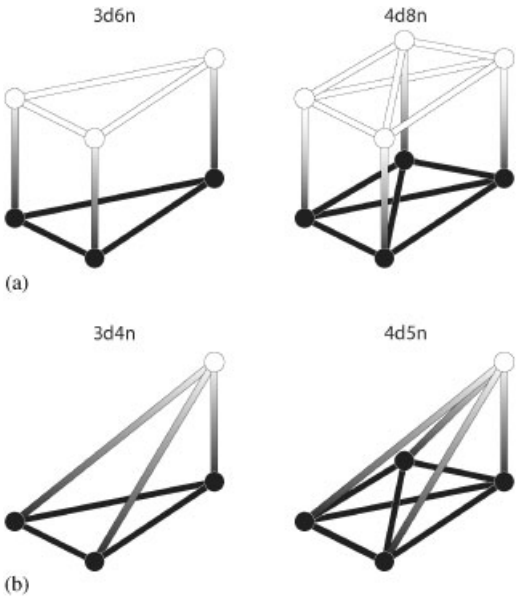

Figure 1. Comparison of prism- and simplex-type space–time elements. Black nodes correspond to t_n and white nodes correspond to t_{n+1}: (a) prism-type space–time elements and (b) simplex-type space–time elements.

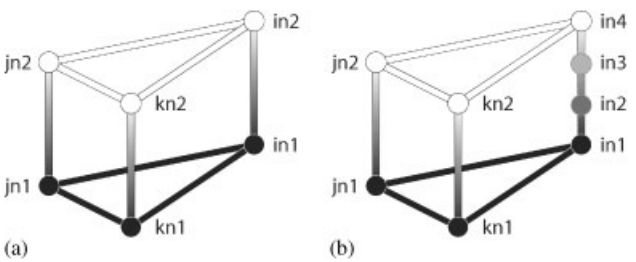

Figure 2. Temporal refinement of a 3d6n space–time prism: (a) original and (b) refined.

diagonal line (2D) or diagonal triangle (3D) will be *compatible* across the neighboring space–time prisms (see Figure 3(a)). A simple solution is to randomly perturb the time coordinates of some or all nodes, as illustrated in Figure 3(b). This perturbation, the same for each prism sharing a particular node, ensures the uniqueness of Delaunay process and guarantees the compatibility of the $(n_{\mathrm{sd}}-1)$-simplices between the neighboring space–time prisms. After the connectivity of the unstructured space–time mesh is established, the time coordinates can be restored to their original values. In practice, the perturbation needs to be applied only to the space–time nodes corresponding to times greater than t_n.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

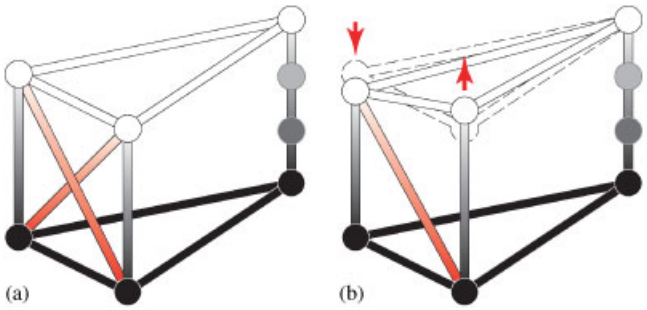

Figure 3. Perturbation of temporal coordinates of a 3d6n space–time prism: (a) before and (b) after.

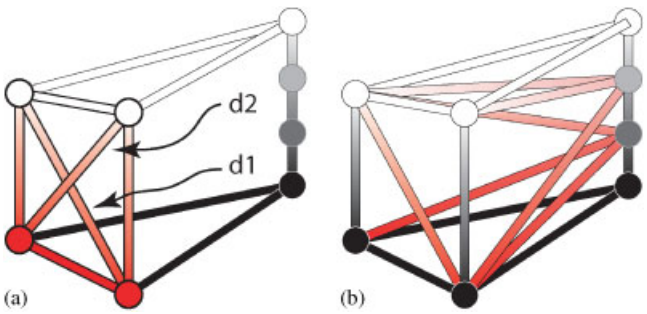

Figure 4. Sliver element elimination in a 3d6n space–time prism: (a) rejected sliver and (b) final connectivity.

2.5. *Delaunay triangulation*

The Delaunay method of generating simplex tesselations, although commonly used in 2D and 3D, is in principle applicable to any number of dimensions. The procedure relies only on point-sphere distance computations, which are easily expressed in four or higher number of dimensions. Freely available implementations of the n-dimensional Delaunay algorithm are available, e.g. in the `qhull` [18] package. The Delaunay approach alone finds limited application in engineering mesh generation, because it is limited to tesselating convex regions only and because it routinely generates sliver (nearly zero-volume) elements. The second issue will be addressed shortly; the first limitation is not a problem in our application, because each space–time prism is *convex*. The non-convexity of the spatial domain has been entirely accounted for by the spatial mesh alone.

2.6. *Sliver elimination*

As previously mentioned, the Delaunay approach, when applied to node sets where more than n_{sd} nodes may lie on a single plane (as is the case with non-simplical space–time prism faces), usually produces sliver, or nearly zero-volume, simplices, as shown in Figure 4. Although the inner diagonal (marked d1) is unique due to the perturbation of time coordinates, an additional element defined by both diagonals d1 and d2 may also be produced. An additional test is needed here to detect and remove generated elements that have nearly zero volume. Note that, e.g. perturbation

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

of node spatial coordinates would remove this ambiguity in the Dealunay procedure, but it would also *guarantee* near-sliver creation in *one* of the two prisms sharing each face. Therefore, *a posteriori* sliver elimination is always necessary and easily accomplished by computing 3D or 4D element volumes and by rejecting elements whose volume falls below a specified threshold.

2.7. Connectivity generation

Having constructed connectivity information inside each space–time prism that (a) incorporates additional nodes placed between the slab-delimiting time levels and (b) is compatible between neighboring prisms, it is trivial to convert it into a global connectivity information that connects all the space–time slab nodes in a network of $(n_{\mathrm{sd}}+1)$-simplex elements. Although this network is difficult to visualize in 4D, the computer program that generates it is largely identical to the program that generates 3D space–time mesh from a 2D spatial mesh, which can be validated using available visualization and verification tools.

3. IMPLEMENTATION ASPECTS

Although it is safe to say that pentatope FEs are not widely used, they are remarkably similar to their lower-dimensional counterparts such as triangles and tetrahedrons. Table I summarizes the differences between a tetrahedron and a pentatope from the point of view of FE implementation. One non-standard issue is the 4D quadrature scheme by Cools and Rabinowitz [19] and Stroud [20].

The use of pentatope elements in place of extruded space–time prisms brings about a significant increase in the number of FEs in each slab. Even if no localized temporal refinement is used (defeating somewhat the purpose of using simplex space–time elements), the ratio of 4D prisms to pentatopes can be of the order of 5–6, which renders any computational scheme based on

Table I. Geometric and implementation details concerning 3D and 4D prismatic and simplex finite elements.

Element type	Tetrahedron 3d4n	Pentatope 4d5n
Physical coordinates	x, y, t	x, y, z, t
Reference coordinates	ξ, η, θ	ξ, η, ζ, θ
Number of element nodes	4	5
Number of element faces	4	5
N_1	ξ	ξ
N_2	η	η
N_3	θ	ζ
N_4	$1-\xi-\eta-\theta$	θ
N_5		$1-\xi-\eta-\zeta-\theta$
Reference volume	$\frac{1}{6}$	$\frac{1}{24}$
Number of quadrature points	4	5
Quadrature weights [19]	$w_i=\frac{1}{4}, \quad i=1,\ldots,4$	$w_i=\frac{1}{5}, \quad i=1,\ldots,5$
Quadrature positions [19]	$\xi_1=0.585410200000000$	$\xi_1=0.526598632371090503$
	$\eta_1=0.138196600000000$	$\eta_1=0.118350341907227374$
	$\theta_1=0.138196600000000$	$\zeta_1=0.118350341907227374$
		$\theta_1=0.118350341907227374$

Notes: In each case, linear shape functions N_i and third-order quadrature rules are shown. Only first quadrature point position is given and remaining points are obtained by permuting the coordinates.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

element-based storage impractical. However, many FE codes use node-based storage schemes, with the system matrix stored either in compressed sparse row (general case) or in block sparse row (multi-degree-of-freedom case with equal-order interpolation); there is no increase in storage costs in such a scheme due to simply increasing the number of elements. The cost of assembly of the system matrix and residual is expected to increase of course, but in our experience this cost is relatively small compared with the cost of the subsequent solution to that linear system.

4. PROPAGATION OF A GAUSSIAN HILL

The numerical example involves the 1D propagation of a Gaussian hill by means of a convection–diffusion equation. The example is based on the benchmark problem from [21, Section 5.6.1]. The initial profile is described as follows:

$$u(x,0)=\frac{5}{7}\exp\left\{-\left(\frac{x-x_0}{\ell}\right)^2\right\} \tag{1}$$

with boundary condition $u(0,t)=0$ and $u(1,t)=0$ for $t\geqslant 0$, where $x_0=\frac{2}{15}$ and $\ell=7\sqrt{2}/300$.

The convection–diffusion equation of the following form:

$$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}-\nu\frac{\partial^2 u}{\partial x^2}=0 \tag{2}$$

is used, with $a=1$ and $\nu=1/30\,000$. As this example serves as the initial validation of the unstructured space–time mesh solver for problems in two- and three-space dimensions, the 1D problem is solved in a 2D domain discretized with a regular triangular mesh, as shown in Figure 5. The division of the unit interval along the x direction into 150 uniform elements results in a Péclet number of 100.

The hill was first propagated for $t=0.6$ using the Crank–Nicolson time stepping as well as the usual discontinuous-in-time Galerkin time stepping (prismatic space–time elements, linear-in-time interpolation). The latter formulation can be summarized as follows: Find $u^h(x,t)\in\mathscr{S}$, such that for all $v^h(x,t)\in\mathscr{V}$

$$\int_Q v^h\frac{\partial u^h}{\partial t}\,\mathrm{d}x\,\mathrm{d}t+\int_Q v^h a\frac{\partial u^h}{\partial x}\,\mathrm{d}x\,\mathrm{d}t+\int_Q\frac{\partial v^h}{\partial x}\nu\frac{\partial u^h}{\partial x}\,\mathrm{d}x\,\mathrm{d}t+\int_\Omega (v^h)^+((u^h)^+-(u^h)^-)\,\mathrm{d}x$$
$$+\sum_e\int_{Q^e}\left[\frac{\partial v^h}{\partial t}+a\frac{\partial v^h}{\partial x}-\nu\frac{\partial^2 v^h}{\partial x^2}\right]\tau\left[\frac{\partial u^h}{\partial t}+a\frac{\partial u^h}{\partial x}-\nu\frac{\partial^2 u^h}{\partial x^2}\right]\mathrm{d}x\,\mathrm{d}t=0 \tag{3}$$

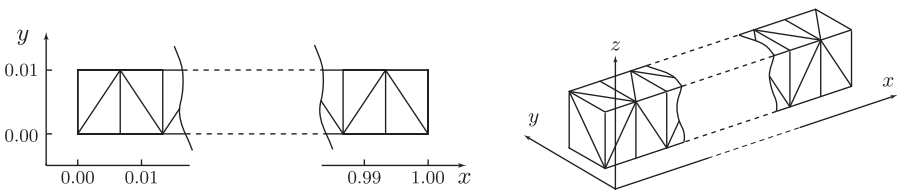

Figure 5. Gaussian hill: domain definition and the spatial discretization in 2D (left) and 3D (right).

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

where the space–time slab for a given time interval I is denoted as $Q=\Omega\times I$, decomposed into element subdomains Q^e. Moreover, the spatial domain is $\Omega=[0,1]$ and $\mathscr{S}=\mathscr{V}=\{u|u\in H^1(Q), u(0,t)=0, u(1,t)=0\}$ are the standard interpolation and weighting function spaces. The notations $(u^h)^+$ and $(u^h)^-$ in the weak-continuity term denote the upper and lower values, respectively, of the discontinuous variable at the lower surface Ω of the space–time slab Q. The stabilization term takes a doubly asymptotic form of [22]. When using piecewise-linear interpolations, as is the case in this paper, the second derivatives in (3) are identically zero. The discretization using Crank–Nicolson time stepping is essentially identical to (3), with $\partial u^h/\partial t$ terms approximated by a difference stencil, $\partial v^h/\partial t$ and weak-continuity term dropped, and all integrations taking place over the spatial domain Ω only.

The time step size was at first chosen as $\Delta t=0.02/3$ corresponding to a Courant number $C=1$ and then increased threefold resulting in $C=3$. Figures 6 and 7 show the results, with high dispersion errors apparent in the Crank–Nicolson solution. These standard results were then

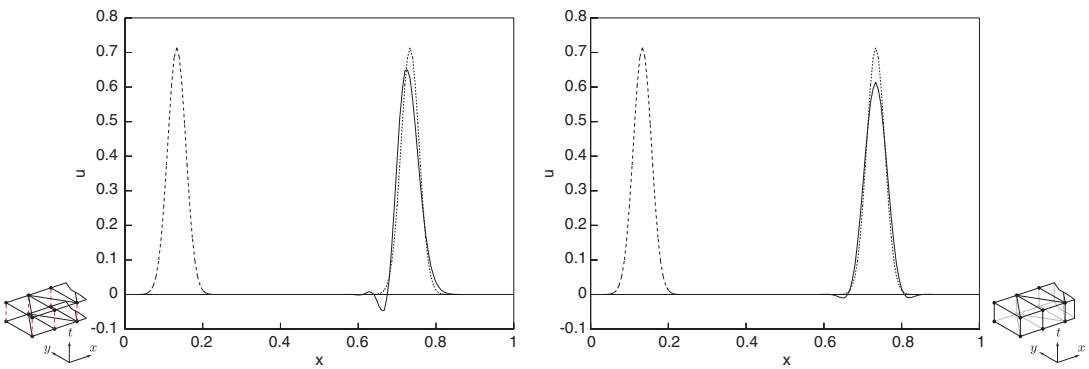

Figure 6. Gaussian hill in 2D: Galerkin/least-squares solution at $t=0.6$ obtained using Crank–Nicolson (left) and discontinuous Galerkin (right) time stepping with $C=1$. The initial condition and the exact solution are also shown.

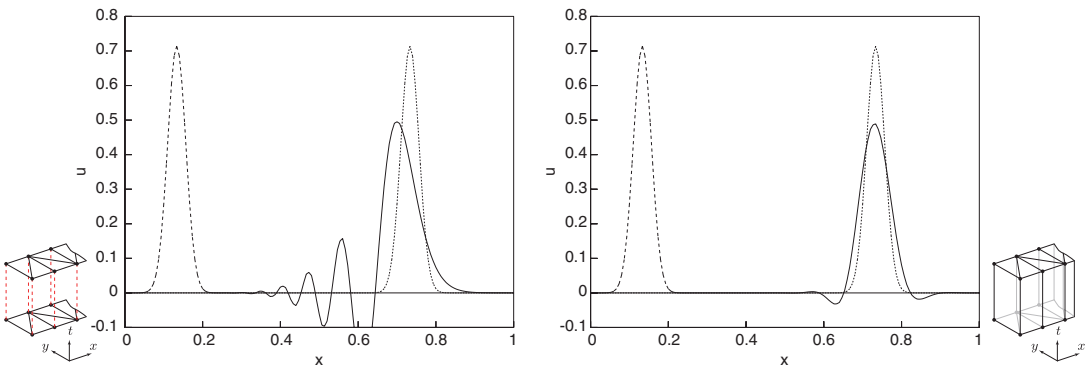

Figure 7. Gaussian hill in 2D: Galerkin/least-squares solution at $t=0.6$ obtained using Crank–Nicolson (left) and discontinuous Galerkin (right) time stepping with $C=3$. The initial condition and the exact solution are also shown.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

compared with the propagation results obtained with the tetrahedral space–time mesh discretization of a slab $\Delta t = 0.02$ thick, with three and one elements in time, resulting again in $C = 1$ and 3, respectively. The results are shown in Figure 8.

The experiment was then repeated in 3D again for $t = 0.6$ using the Crank–Nicolson time stepping as well as the usual discontinuous-in-time Galerkin time stepping (prismatic space–time elements, linear-in-time interpolation). The time step size was again chosen as $\Delta t = 0.02/3$ corresponding to a Courant number $C = 1$, and then increased threefold resulting in $C = 3$. Figures 9 and 10 show the results, with high dispersion errors apparent in the Crank–Nicolson solution. These standard results were then compared with the propagation results obtained with the pentatope-based space–time mesh discretization of a slab $\Delta t = 0.02$ thick, with three and one elements in time, resulting again in $C = 1$ and 3, respectively. The results are shown in Figure 11.

Finally, a hybrid mesh was generated using the technique presented in Section 2, with each of the 30 time slabs being 0.02 thick and discretized differently with one to three elements in the

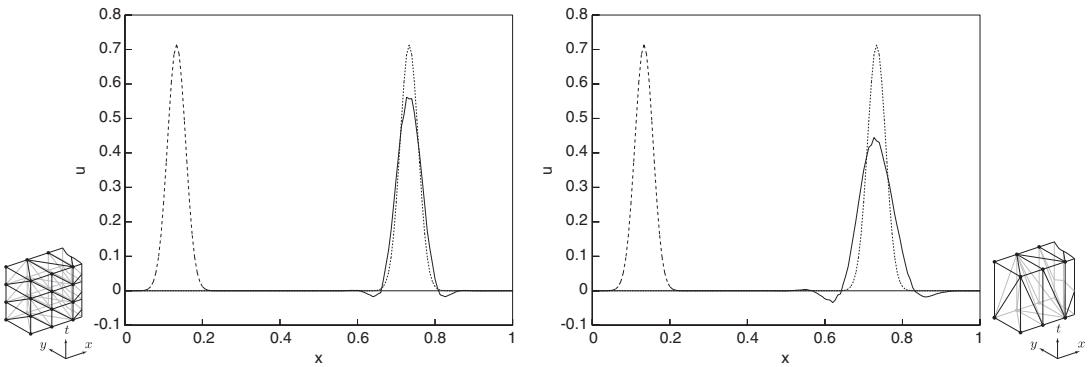

Figure 8. Gaussian hill in 2D: Galerkin/least-squares solution at $t = 0.6$ obtained using tetrahedral space–time discretization with $C = 1$ (left) and $C = 3$ (right). The initial condition and the exact solution are also shown.

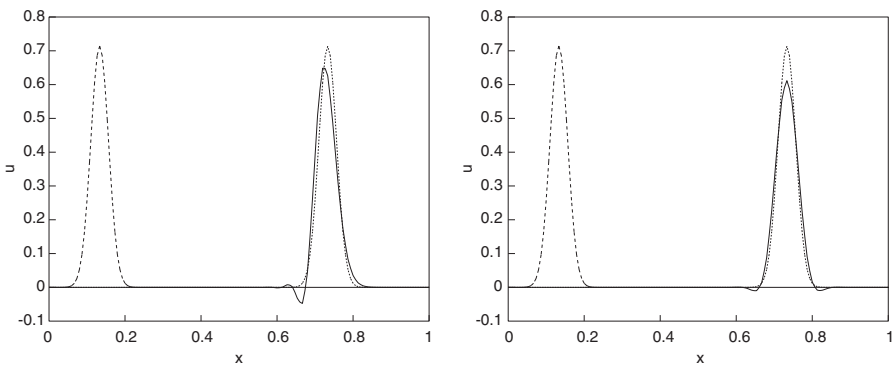

Figure 9. Gaussian hill in 3D: Galerkin/least-squares solution at $t = 0.6$ obtained using Crank–Nicolson (left) and discontinuous Galerkin (right) time stepping with $C = 1$. The initial condition and the exact solution are also shown.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

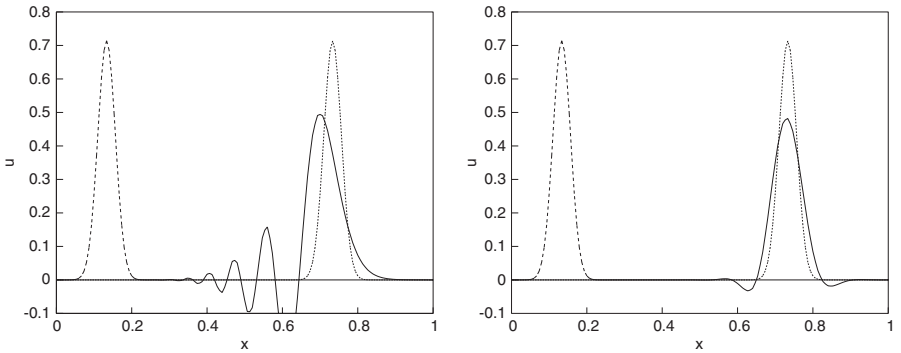

Figure 10. Gaussian hill in 3D: Galerkin/least-squares solution at $t=0.6$ obtained using Crank–Nicolson (left) and discontinuous Galerkin (right) time stepping with $C=3$. The initial condition and the exact solution are also shown.

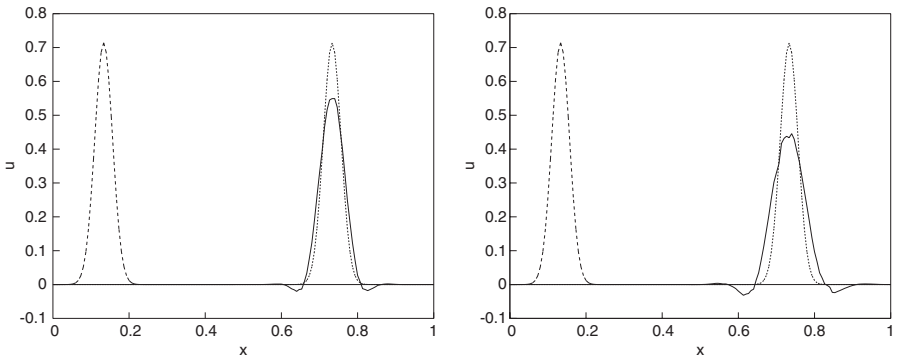

Figure 11. Gaussian hill in 3D: Galerkin/least-squares solution at $t=0.6$ obtained using pentatope-based space–time discretization with $C=1$ (left) and $C=3$ (right). The initial condition and the exact solution are also shown.

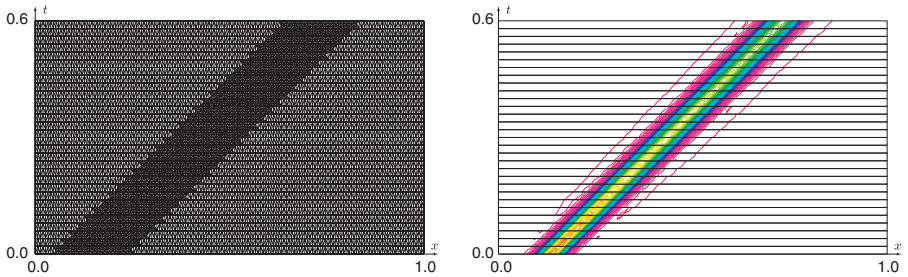

Figure 12. Gaussian hill in 3D: Galerkin/least-squares solution at $t=0.6$ obtained using pentatope-based space–time discretization with $C=1$–3. The space–time mesh corresponding to the spatial domain *edge* $(x,0,0)$ is shown on the left. The contour plot of the advected quantity for that edge is shown on the right. See Figure 13 for an elevation plot.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

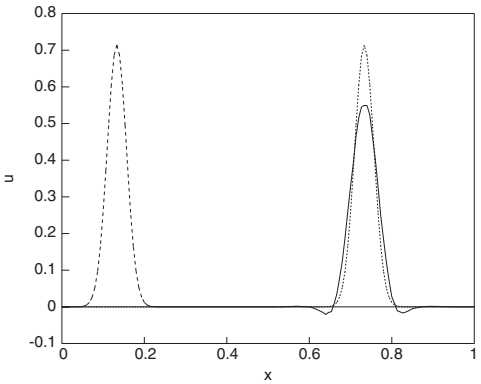

Figure 13. Gaussian hill in 3D: Galerkin/least-squares solution at $t=0.6$ obtained using pentatope-based space–time discretization with $C=1$–3. The elevation plot of the advected quantity corresponding to the spatial domain *edge* $(x,0,0)$ is shown. The initial condition and the exact solution are also shown.

time direction, resulting also in $C=1$–3. The resulting mesh is a fully unstructured pentatope-based space–time mesh, with the 4D grid projected onto 2D plot by selecting all triangular faces corresponding to the spatial *edge* $(x,0,0)$ as shown in Figure 12. The contour plot of the advected quantity ($-0.05\leqslant u\leqslant 0.75$) for that edge is also shown in that figure. In Figure 13, the elevation plot along that edge is shown, with the dissipative behavior comparable to that obtained with $C=1$, as already shown in Figure 11 (left), but with significantly fewer pentatope elements (4200 *versus* 9000). The result is slightly worse than that obtained with prismatic space–time elements with $C=1$ shown in Figure 9 (right), but certainly better than the result for prismatic space–time elements with $C=3$ shown in Figure 10 (right).

5. EFFICIENCY ASPECTS

The example in Section 4 can be considered too small to provide reliable timing measurements; nevertheless, the typical performance behavior is summarized in Table II. Table II lists the number of time steps, equation system size, as well as the time required to form and to solve the equation systems (summed over all time steps required to solve the benchmark problem) for (a) prism-based space–time elements with small time step ($C=1$), as used in Figure 9 (right), (b) prism-based space–time elements with large time step ($C=3$), as used in Figure 10 (right), and (c) pentatope-based space–time elements with variable time step ($C=1$–3), as used in Figure 13.

It is expected that, assuming fixed linear solver parameters, the time required to obtain a solution with $C=1$ is thrice that required to obtain a solution with $C=3$. Two further aspects are apparent from the above table. The number of equations (nodes) in the pentatope mesh with variable temporal refinement is only 20% higher than the number of equations for a prismatic mesh with the same maximum time step size. Therefore, the solution time is also 20% higher, when using iterative solvers with linear scaling properties. This time is still significantly lower than 200%-higher-time resulting from high uniform temporal refinement. The conditioning of the linear systems arising from both types of elements was not examined in detail; the convergence of the iterative GMRES solver used was observed to be similar for all cases considered.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

Table II. Typical performance behavior of the prism- and pentatope-based calculations.

	Time steps	Equations per step	System formation (s)	System solution (s)	Comments
Prisms, $C=1$	30	1192	5.38	20.93	Low dissipation, slow
Prisms, $C=3$	10	1192	1.89	7.00	High dissipation, fast
Pentatopes, $C=1$–3	10	1432	3.99	8.57	Low dissipation, slow formation fast solution

On the other hand, as the number of elements in a pentatope-based mesh is significantly higher than that in the prism-based meshes (4200 *versus* 750), even though the elements are simpler, the system formation time is also significantly higher—in fact, it is more than twice. It remains lower, however, than the time required to repeatedly form the system with a uniformly refined prism-based mesh, which would be necessary in a non-linear case. This aspect is not seen as critical to the efficiency of the new approach; in our experience, the system formation time is not significant compared with the solution time for realistic problems solved using implicit time stepping.

It can also be concluded that using pentatopes *without* exploiting their potential for variable temporal refinement is not an attractive option; the accuracy properties are not improved, the equation system size is the same compared with prism-based mesh with the same time step size, and the number of elements is massively increased leading solely to increased system formation times.

6. CONCLUDING REMARKS

A straightforward method for generating simplex space–time meshes has been introduced, allowing arbitrary temporal refinement in selected portions of space–time slabs. The goal is to increase the flexibility of space–time discretizations, bypassing partially the limitations of mesh generation tools which are universally limited to three spatial dimensions. Implementation aspects arising from the use of unusual pentatope elements have been described; they are shown to be relatively minor. The resulting tetrahedral (for 2D problems) and pentatope (for 3D problems) meshes were tested in the context of advection–diffusion equation and were shown to provide reasonable solutions, while enabling varying time refinement in portions of the domain. Although for the presented example it may be possible to generate the required connectivities *manually*, taking advantage of the simple geometry, that was purposefully not done; any geometry-specific mesh generation algorithm will not be in general applicable to realistic problems of engineering interest, and our effort concerns *automatic* connectivity generation. In this approach, the only user input is the spatial mesh and the number of temporal nodes required for each spatial node. The extension of the method to problems of more engineering significance, such as incompressible Navier–Stokes equations, is straightforward, although the numerical behavior of stabilized FE formulations on such highly unstructured space–time meshes still needs to be examined. Similarly, cost benefits and best-case and worst-case scenarios are yet to be analyzed for more complex geometries. It remains to be seen whether the mesh generation scheme could be possibly extended to connect disparate spatial meshes at the bottom and top levels of the space–time slab.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

ACKNOWLEDGEMENTS

The author gratefully acknowledges the computing resources provided by the RWTH Aachen University Computing and Communication Center and by the Jülich Supercomputer Center. The author also wishes to acknowledge the support of the German Science Foundation under programs GSC 111, EXC 128, SFB 540 and 401, and SPP 1253 and 1273. The author would like to thank the Chair for Computational Mechanics (LNM) at the Technical University of Munich for the creative atmosphere that led to the first version of this article, and then to a finished version during a subsequent visit.

REFERENCES

1. Jamet P, Bonnerot R. Numerical solution of the Eulerian equations of compressible flow by a finite element method which follows the free boundary and the interfaces. *Journal of Computational Physics* 1975; **18**: 21–45.
2. Bonnerot R, Jamet P. Numerical computation of the free boundary for the two-dimensional Stefan problem by space–time finite elements. *Journal of Computational Physics* 1977; **25**:163–181.
3. Lynch DR, Gray WG. Finite element simulation of flow in deforming regions. *Journal of Computational Physics* 1980; **36**:135–153.
4. Frederiksen CS, Watts AM. Finite-element method for time-dependent incompressible free surface flows. *Journal of Computational Physics* 1981; **39**:282–304.
5. Jamet P. Galerkin-type approximations which are discontinuous in time for parabolic equations in a variable domain. *SIAM Journal on Numerical Analysis* 1978; **15**:912–928.
6. Hughes TJR, Franca LP, Mallet M. A new finite element formulation for computational fluid dynamics. VI. convergence analysis of the generalized SUPG formulation for linear time-dependent multi-dimensional advective–diffusive systems. *Computer Methods in Applied Mechanics and Engineering* 1987; **63**:97–112.
7. Hughes TJR, Franca LP, Hulbert GM. A new finite element formulation for computational fluid dynamics. VIII. the Galerkin/least-squares method for advective–diffusive equations. *Computer Methods in Applied Mechanics and Engineering* 1989; **73**:173–189.
8. 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.
9. Shakib F. Finite element analysis of the compressible Euler and Navier–Stokes equations. *Ph.D. Thesis*, Stanford University, Department of Mechanical Engineering, 1988.
10. Hansbo P, Szepessy A. A velocity–pressure streamline diffusion finite element method for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1990; **84**:175–192.
11. 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**:339–351.
12. 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**:353–371.
13. Hansbo P. The characteristic streamline diffusion method for the time-dependent incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1992; **99**:171–186.
14. Farhat C, Chandesris M. Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications. *International Journal for Numerical Methods in Engineering* 2003; **58**:1397–1434.
15. Maubach JML. Iterative methods for non-linear partial differential equations. *Ph.D. Thesis*, University of Nijmegen, 1991.
16. Idesman A, Niekamp R, Stein E. Finite elements in space and time for generalized viscoelastic Maxwell model. *Computational Mechanics* 2001; **27**:49–60.
17. Sathe SV. Enhanced-discretization and solution techniques in flow simulations and parachute fluid–structure interactions. *Ph.D. Thesis*, Rice University, Department of Mechanical Engineering and Materials Science, 2004.
18. Barber CB, Dobkin DP, Huhdanpaa H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software* 1996; **22**:469–483.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld

19. Cools R, Rabinowitz P. Monomial cubature rules since 'Stroud': a compilation. *Journal of Computational and Applied Mathematics* 1993; **48**:309–326.
20. Stroud AH. *Approximate Calculation of Multiple Integrals*. Prentice-Hall: Englewood Cliffs, NJ, 1971.
21. Donea J, Huerta A. *Finite Element Methods for Flow Problems*. Wiley: New York, 2003.
22. 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.

Copyright © 2008 John Wiley & Sons, Ltd.

DOI: 10.1002/fld