High-Order Matching Finite Elements on Recursive Grids

G. Kozlovsky*

Abstract

Uniform rectangular grids, which are used mainly in the academic environment, permit very simple high-degree elements and are the easiest grids to implement, but are limited as far as local adaptation and approximation of boundaries is concerned. Despite of the difficulty of matching triangular high-order elements, triangular unstructured grids dominate the world of engineering computing on the strength of their ability to approximate curved boundaries and to provide variable local resolution. However, the unfavorable rate of convergence of low-degree methods is a major drawback, especially in high-precision computing.

Hierarchical recursive grids require more complex data structures but are attractive for use in dynamic adaptive algorithms. Recently, in his paper [6] the author investigated properties of a subclass of recursive grids called "consistent" grids, which can be viewed in certain sense as an intermediate type between the two previous grid types, and developed a version of the finite element method suited for treatment of the interfaces between computational cells of different sizes. In our current presentation we propose an extension of this method to high-degree Hermite finite elements and discuss ways to treat curved boundaries when using recursive grids. In addition to their matching properties, Hermite elements are uniquely suitable for creating general-purpose software, being uniformly extensible for any order and any number of space dimensions.

In light of all their proved and potential advantages discussed above, it appears that the main reason recursive grids are not widely used is the complexity of their implementation. This problem can be overcome by creating grid-handling program libraries and specialized development and visualization environments [5].

Key words: finite elements, adaptive grids.

*CN Division, CERN, 1211 Geneva 23, Switzerland

ICOSAHOM'95: Proceedings of the Third International Conference on Spectral and High Order Methods.

AMS subject classifications: 65D99, 65M60, 65M50, 65N30, 65N50, 65Y99.

1 Introduction

The choice of grid type is one of the most crucial choices in numerical solution of partial differential equations. An eloquent, if perhaps slightly unfair towards approximation and solution methods, description of the importance of grid choice was presented by J.F. Thomson [10]:

> "The ultimate answer to numerical solution of partial differential equations may well be dynamically-adaptive grids, rather than more elaborate difference representations and solution methods. It has been noted by several authors that when the grid is right, most numerical solution methods work well. Oscillations associated with cell Reynolds number and with shocks in fluid mechanics computations have been shown to be eliminated with adaptive grids. Even the numerical viscosity introduced by upwind differencing is reduced as the grid adapts to regions of large solution variation."

Since the early days of the finite element method the two competing methods of choice were uniform rectangular grids and triangular non-structured grids. Gilbert Strang wrote in his classical book [9]

> "... it is not clear at this writing whether it is more efficient to subdivide the region into triangles or into quadrilaterals. Triangles are obviously better at approximating a curved boundary, but there are advantages to quadrilaterals (and especially to rectangles) in the interior: there are fewer of them, and they permit very simple elements of high degree."

Uniform rectangular grids are widely used in the academic world, probably because, in addition to the advantages described by Strang, they are easy to program using

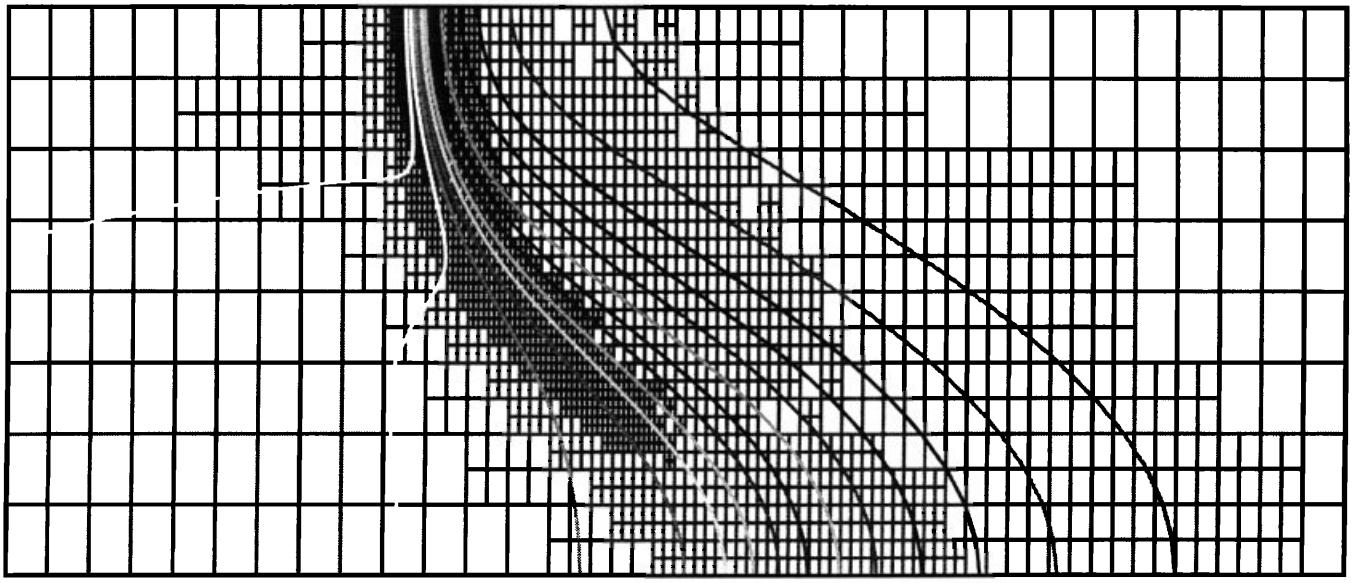

Figure 1: An example of consistent recursive grid with contours of the solution superimposed. The solution represents the temperature distribution of a premixed flame propagating in a shear flow field. The grid results from a dynamic adaptation process where the grid changes as the solution develops from an initial estimate to the state shown in the picture.

the simplest data structure — a Fortran array. In many academic applications, be it testing of new numerical methods, or using numerical solutions to investigate scientific phenomena, the computational domain is rectangular and local adaptivity is not crucial. By transforming the computational domain it is possible to adapt a logically uniform rectangular grid to a non-rectangular computational domain and to provide variable local resolution. However, this transformation requires a solution of a global problem, for many particular geometrical configurations of the area to be refined, the resulting economy is small or non-existent (consider, for example, in two dimensions, a thin ring area to be refined, with diameter close to the diameter of the computational domain), and the numerical error caused by a large condition number of the transformation can be considerable.

Unstructured triangular[1] (sometimes mixed with quadrilaterals) grids have unquestionably won the engineering world on the strength of their ability to approximate curved boundaries and, not less important, their ability to provide variable local resolution. Paradoxically, unstructured grids, when not dynamically modified during the computation process, fit almost as well into simple data structures, because their elements are uniformly related to each other (they don't form a hierarchy). A notable problem with these grids is the difficulty of matching high-order finite elements. Because the local geometry of the grid may differ at every node, each row of the stiffness matrix has to be computed separately, so the calculation of the stiffness matrix is generally more expensive compared with the rectangular grids. Geometrical searches on unstructured grids are more computationally expensive too, compared with the structured grids.

Unlike the two classes of grids discussed above, hierarchical recursive grids (see Figure 1) require more complex data structures [8] which are not straightforward to program in Fortran, and this retarded their use until the early '80s. The main reason for the interest in recursive grids was their use in adaptive algorithms. Recently, in his paper [6], the author investigated properties of a subclass of recursive grids called "consistent" grids and showed that they can be viewed as semi-structured grids, in the sense that the number of possible local geometrical configurations of the grid is limited (as compared to one configuration for uniform rectangular grids and infinity for non-structured grids).

[1] Although we use two-dimensional case in our discussion here, it is relevant to three-dimensional case too.

The treatment of the interface between computational cells of different sizes is a major problem in using composite grids. For recursive grids, the standard nodal formulation of the finite element method can lead to an undesirable unlimited increase in the number of nodes in the discretization stencil. This problem does not arise for finite element formulation with element-based parameters, however the number of unknowns in this case is much greater than for the nodal formulation. To solve the problem of the discretization stencil increase, we developed a special version of the finite element method for consistent recursive grids. Our version uses discontinuous piecewise bilinear basis functions to simplify matrix generation on interfaces of computational cells of different sizes. The solution remains continuous due to additional constraints imposed on the unknowns. With our method, the number of possible discretization stencils is fixed and the stiffness matrix for constant coefficient operators can be preassembled.

Returning to comparison of the main grid types, the major advantage of uniform rectangular grids was their ability to support high-degree finite elements. This ability is of major importance for computing high-precision solutions, because an $O(h)$ method in three dimensions results in about $O(\epsilon^{-3})$ unknowns, where ϵ is the required accuracy of the solution. Given that the solution of implicit equations can rarely be accomplished in linear time, the real rate of growth in computational work versus accuracy is likely to be much higher than the rate quoted above. "High-performance" computing applied to methods with such an unfavorable rate of convergence will not take us far, for with a 1000-fold increase in the computing power we can gain at most 10-fold increase in precision. This 10-fold increase in precision will require also 1000-fold increase in data volume with the associated increase in memory requirements.

Another important criterion for grid comparison is their suitability for creating general purpose software. Undoubtedly, for every grid type there are problems for which this particular type is the most efficient. Although it is difficult to formalize, it is clear that some grid types are less suitable for use in general-purpose solvers, often because of the global dependencies they introduce. Recursive grids possess a very important property of *locality*, meaning that local changes in the required resolution lead to local changes in the grid. This property greatly simplifies adaptive methods and, eventually, adaptive software. The locality property is especially important for dynamic grids, which change frequently as the solution develops.

Composite or nested grids [3] are closely related to recursive grids. Recursive grids can be considered a limiting case of composite grids in which every higher level patch covers one cell of the lower level grid and contains exactly four cells. To the extent that they differ from recursive grids (as patches grow in size), composite grids lose the locality property that distinguishes recursive grids. The non-local nature of composite grids leads to the need for a pattern recognition algorithms to cluster refinement areas into large patches of fine grid [4]. While composite grids are superior for vectorization, it is not clear at present which type of grid is preferable on MIMD distributed memory computers. We believe that the problem of discretization on the interface between coarse and fine grids has not been sufficiently addressed in the literature on composite grids. Many authors avoid this problem by solving their problem on the coarse grid and using the resulting solution as the boundary condition for the fine grid. While this approach can work for demonstrating adaptive grid efficiency or for some selected problems, it is not suitable in the general case. A coarse grid solution obtained without taking into account the influence of the refined area may contain a large error, or make no physical sense at all. For example, in combustion problems [7] the flame front can "fall" between the grid nodes on the coarse grid and become "extinguished," leading to a completely wrong solution.

In this paper we restrict our attention to building an approximation space with any desired degree of local smoothness and its computer implementation. For discussion of grid refinement criteria and the choice of the local degree for the trial functions see [1, ?]. The approach we describe here for the two-dimensional case is trivially extensible to any number of dimensions.

2 Dynamic recursive grids

A dynamic recursive grid is based on a rectangular uniform grid (base grid) covering the computational domain. The grid cells can be refined by dividing them into four equal subcells with lines parallel to the lines of the base grid. This refinement process can be repeated recursively for any of the newly created subcells. The refinement of a cell can later be reversed provided none of its four subcells is currently refined. As can be seen in Figure 1, some of the nodes are surrounded by four cells, possibly of different size, and some by three cells, so that the node lies in the middle of a cell side. We will call the former r-nodes (r from regular) and the latter t-nodes (as suggested by the shape of the adjacent lines). The nodes on the ends of the cell side in the middle of which a t-node is located will be called *controlling nodes* for the t-node.

We impose *consistency* requirement on the grid, that is,

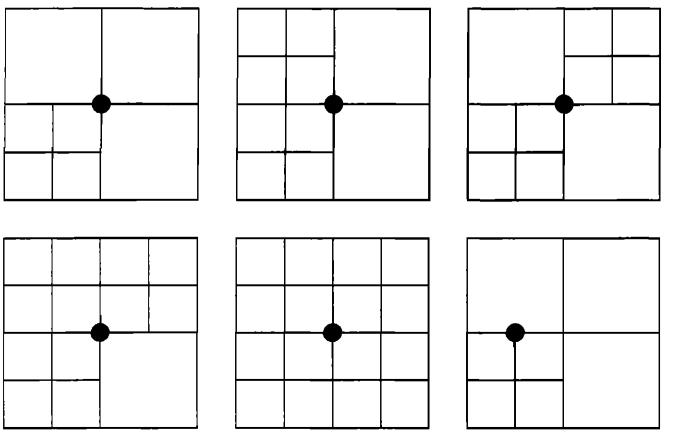

Figure 2: Nontrivial local geometrical configurations around a grid node. The remaining configurations can be obtained by rotation.

any two adjacent (including cells having a common vertex) nonrefined cells differ at most by a factor of two in their linear size. This requirement leads to some non-local dependencies in the grid. However, as we will see below, these non-local dependencies can be treated in a completely automatic way with negligible amount of computational work. In exchange, the consistency requirement brings an important advantage — *semi-regularity* of the grid, meaning that the grid has only a limited number of local geometrical configurations around a grid node (see Figure 2). It is shown in [6] that for an internal terminal r-node in two dimensions this number is fifteen (sixty-three for a three-dimensional grid).

Grid adaptation is done in steps that may be interspersed with numerical computations. One step of grid adaptation is a solution to the following problem.

Definition 2.1 Grid adaptation problem.
Given a grid $\mathcal{G}$, a set of its cells $\mathcal{R}$ to be refined, and a set of its cells $\mathcal{U}$ which are allowed to be unrefined, find a minimal consistent grid in which all the cells of $\mathcal{R}$ are refined and all the cells not belonging to $\mathcal{U}$ are preserved. A minimal grid is a grid from which no cells can be deleted without violating one of the above conditions. □

It is easy to see that because of the consistency requirement and because a cell can be unrefined only if all of its four subcells are marked for deletion, the minimal grid may contain some cells of $\mathcal{U}$ marked for deletion, and some cells not belonging to $\mathcal{R}$ will be refined.

In [6] we prove that the minimal grid is unique and provide a grid adaptation algorithm to construct it. The algorithm is optimal in the sense that no cell refined at its intermediate stage is later unrefined. The algorithm is implemented as part of the grid-handling library. We would like to stress that in our implementation grid adaptation is completely independent from the refinement/unrefinement criteria used by a user of the grid-handling library. The user only communicates the sets $\mathcal{R}$ and $\mathcal{U}$ to the grid adaptation routines.

3 Finite element space

We consider here the construction of a finite element approximation (trial) space on consistent recursive grids. The trial space has variable local degree of approximation and smoothness. Such a space together with the dynamic recursive grid machinery described in the previous section can be used not only to solve partial differential equations by the h-p version of the finite element method, but also to solve problems in many other areas such as image processing and numerical integration, to name a few. A general-purpose computer implementation of this functional space would free people wishing to use adaptive approximation from drudgery involved in working with adaptive grids for loftier tasks in their chosen fields of interest.

A trial function over a recursive grid is taken to be a polynomial over each grid cell. Each polynomial depends on a number of parameters. These parameters may be coefficients of the polynomial, nodal values of the polynomial and its derivatives, or something else. If varying the parameters can produce the whole space of polynomials of degree m, we will say that the local degree of approximation over the corresponding cell is m. To be precise, by a polynomial of degree m of two variables we mean a polynomial in which the sum of the powers of both variables in every monomial is less than or equal to m. In the two-dimensional case the number of dimensions (equal to the number of polynomial coefficients) of the space of polynomials of degree m is $(m+2)(m+1)/2$.

3.1 Element matching problem

A trial function defined above does not satisfy any smoothness requirements on the inter-element boundaries. Constraining the parameter space in such a way that the trial function has continuous derivatives up to a certain order (the function itself is considered the 0-th order derivative) over inter-element boundaries is called *matching* the elements. Problems of matching different kinds of elements

are discussed in [9]. Matching may be required by the nature of the problem to be solved, or it may be desirable because it reduces the number of unknowns under a nodal formulation, without diminishing the degree of approximation. Following [9] we denote C^q the class of functions with continuous derivatives through order q.

The nodal approach to the finite element formulation leads to substantial savings. For example, while for a uniform rectangular grid the number of elements is approximately equal to the number of nodes, a bilinear finite element formulation with cell-based parameters requires keeping four parameters per element, but only one parameter per node is required for node-based formulation.

A convenient way to investigate matching of elements is to take as parameters nodal values of the polynomial and its derivatives. Then, matching the value and derivatives through order q at nodes is a necessary condition for the elements to match up to order q. However, this condition is not sufficient in the general case. Strang [9] quotes the following (unpublished) result of Zenisek concerning the matching of triangular elements: *To achieve piecewise polynomials of class C^q on an arbitrary triangulation, the nodal parameters must include all derivatives at the vertices of order less than or equal to $2q$.* It is conjectured that the minimum possible degree of such an element in n dimensions is $2^n q + 1$. On the other hand, matching Hermite elements on a uniform rectangular grid with the gridlines along the coordinate axis is very easy. It is sufficient to match the value and the derivatives up to order q at the nodes to have the polynomials match up to the same order along the whole inter-element boundary connecting the nodes.

3.2 Choosing trial function support

Nodal formulation on uniform rectangular grids has the following important properties:

- Under condition that a basis function is a single polynomial inside every element, its support is the smallest possible — four elements surrounding the node.
- Inside every element only the basis functions based on its own vertices are non-zero.
- The discretization stencil includes only nodes of the elements adjacent to the central node.
- The smoothness of the trial space does not exceed the smoothness of the basis functions.

An attempt to extend nodal formulation from uniform rectangular grids to consistent recursive grids leads to several possibilities to consider.

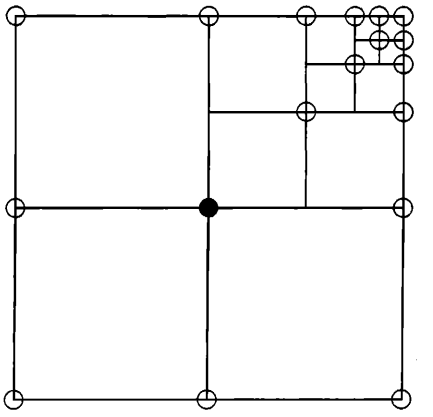

Figure 3: Discretization stencil example for the minimum rectangle basis function support. The central node is marked by a black disk, remaining nodes of the discretization stencil are marked by circles. Note that in this case the discretization stencil may spread outside of the rectangular area on the picture.

The simplest way is to take the smallest possible rectangular area around an r-node as the support of a basis function (see Figure 3). There are no basis functions at t-nodes. Basis functions belonging to the class C^q can be built exactly like for uniform grids. However, if we use the Galerkin formulation with the weight functions identical to the basis functions, the discretization stencil can grow unbounded, depending on the refinement pattern. The number of non-zero basis functions inside an element also becomes unbounded.

The next approach is to reduce the support of a basis function to the smallest number of cells which allow the building of C^q basis functions that are piecewise polynomials inside every cell (we will show how to construct these functions later). Such basis functions are defined in r-nodes (see Figure 4). There still can be non-zero basis functions inside an element that are based at an outside node, but the number of such functions is limited. The discretization stencil can not grow unlimited, but there is an undesirable dependency on the refinement pattern outside of the immediate vicinity of the central node (see Figure 4).

Finally, extending the approach described in [6] to arbitrary degree discontinuous Hermite finite elements, we reduce the support of the basis functions at r-nodes to the four cells adjacent to the node. Basis functions at t-nodes have their support at the two cells sharing vertices at the node (see Figure 6). The trial function inside every cell is then a linear combination of only the basis functions at

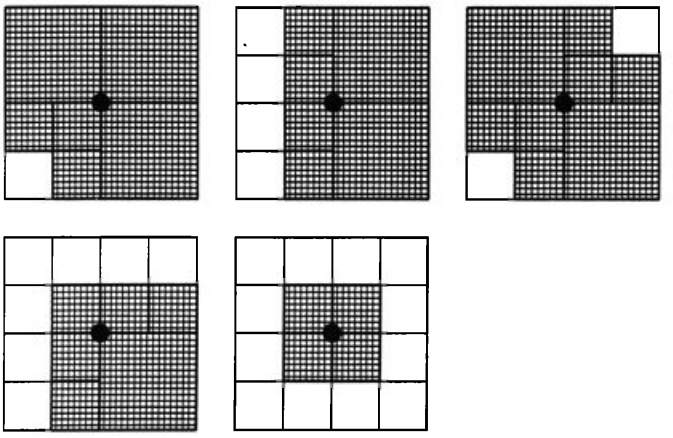

Figure 4: The minimal support of continuous basis functions for the principal geometrical configurations of the grid. The central nodes are marked by black disks.

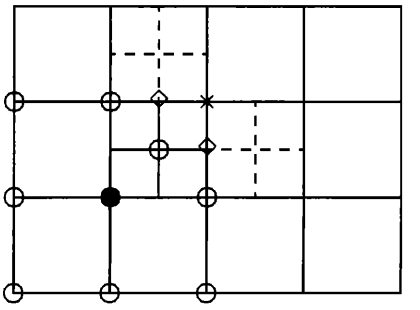

Figure 5: An example of discretization stencil non-local dependence for continuous basis functions with minimal support. If the refinement shown by dashed lines is performed, the nodes marked by $\diamond$ belong to the discretization stencil while the node marked by $\times$ does not. If the said refinement is undone, the node marked by $\times$ enters the discretization stencil and the nodes marked by $\diamond$ leave it.

the nodes of the cell. In the next subsection we develop this approach in detail and show how the smoothness of the trial function can be guaranteed despite the discontinuity of the basis functions. We also show that with proper choice of weight functions for the Galerkin formulation the propagation of the discretization stencil is prevented.

3.3 Discontinuous trial functions with the minimal support

Given local nodal degree of smoothness q_r at every r-node r we shall seek an unknown function in the following form

$$\phi = \sum_{i,j,r} a^{ij}_{q_r r}\phi^{ij}_{q_r r} + \sum_{i,j,t}\left(a^{ij}_{q_t^- t}\phi^{ij}_{q_t^- t} + a^{ij}_{q_t^+ t}\phi^{ij}_{q_t^+ t}\right), \tag{1}$$

where indexes r and t run through the r-nodes and t-nodes, respectively; $q_t^- = q_{t-}$ and $q_t^+ = q_{t+}$ for a t-node t with the controlling r-nodes t^- and t^+; indexes i and j assume independently all the non-negative values less than or equal to the corresponding value of q; and ϕ^{ij}_{qk} are piecewise polynomial basis functions which assume the following values on the grid nodes $p_k = (x_k, y_k)$

$$\frac{\partial^{i'+j'}\phi^{ij}_{qk}}{\partial x^{i'}\partial y^{j'}}(p_{k'}) = \begin{cases} 1 & i = i', j = j', k = k' \\ 0 & otherwise \end{cases} \tag{2}$$

Inside every grid cell of its support, the basis function ϕ^{ij}_{qk} is defined as the product of two one-dimensional polynomials u^i_{qk} and v^j_{qk} of degree $2q+1$

$$\phi^{ij}_{qk} = u^i_{qk}v^j_{qk} \tag{3}$$

where, for example, polynomial u^i_{qk} is determined by the following conditions specified for $0 \le i' \le q$ and $k' = k, \hat{k}_x$, with $\hat{k}_x$ being the node on the opposite side of the grid cell in question from the node k, along the x coordinate

$$\frac{\partial^{i'}u^i_{qk}}{\partial x^{i'}}(x_{k'}) = \begin{cases} 1 & i = i', k = k' \\ 0 & otherwise \end{cases} \tag{4}$$

It is easy to verify that a basis function thus defined has q continuous derivatives everywhere, except at the sides of elements belonging to its support that start at the central node and pass through a t-node.

As in [6] the required smoothness of the solution is assured by imposing additional conditions at the t-nodes. At every inter-element boundary, these matching conditions assure the continuity of q_{min} derivatives of the solution ϕ, where q_{min} is the minimum of nodal continuity at all the nodes at the corners of the elements adjacent to the boundary

$$a^{ij}_{q_k^- k} = \frac{\partial^{i+j}}{\partial x^i \partial y^j}\sum_{i',j'} a^{i'j'}_{q_k^- k^-}\phi^{i'j'}_{q_k^- k^-}(p_k) \tag{5}$$

$$a^{ij}_{q_k^+ k} = \frac{\partial^{i+j}}{\partial x^i \partial y^j}\sum_{i',j'} a^{i'j'}_{q_k^+ k^+}\phi^{i'j'}_{q_k^+ k^+}(p_k) \tag{6}$$

where the nodes k^- and k^+ are the controlling nodes of a t-node k.

After a series of simple transformations using (3) and

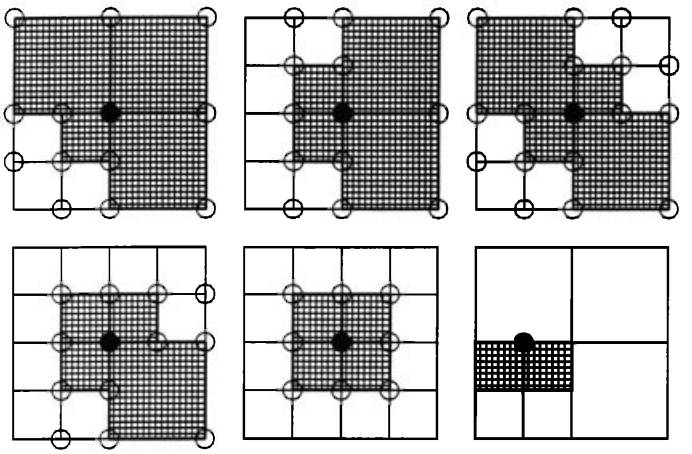

Figure 6: The support of discontinuous basis functions for the principal geometrical configurations of the grid. The central nodes are marked by black disks, the nodes of the discretization stencil for Galerkin formulation with minimal support continuous weight functions are marked by circles.

(4) we obtain a form suitable for computing

$$(7) \qquad a^{ij}_{q_k^- k} = \sum_{i'} a^{i'j}_{q_k^- k^-} \frac{\partial^{i'} u^{i'}_{q_k^- k^-}(x_k)}{\partial x^{i'}}$$

$$(8) \qquad a^{ij}_{q_k^- k} = \sum_{i'} a^{i'j}_{q_k^+ k^+} \frac{\partial^{i'} u^{i'}_{q_k^+ k^+}(x_k)}{\partial x^{i'}}$$

where the coefficients $\partial^{i'} u^{i'}_{q_k^- k^-}(x_k)/\partial x^{i'}$ and $\partial^{i'} u^{i'}_{q_k^- k^+}(x_k)/\partial x^{i'}$ are constants that can easily be computed at the moment the size of the base grid is known.

Incidentally, taking the above basis function at an r-node and adding to it the basis functions at all the adjacent t-nodes with the coefficients defined by the matching conditions (7)-(8) we obtain the $\mathcal{C}^q$ basis function with the minimal support (shown at Figure 4) discussed earlier.

The above continuous basis functions can serve as weight functions for the Galerkin formulation using the minimal support discontinuous basis functions. After the matching conditions (7)-(8) are applied, the resulting solution will be equivalent to the solution obtained using continuous basis functions. The advantage of using the discontinuous basis functions is that the discretization stencil is reduced to the nodes marked by circles on Figure 6 (it has from 9 to 17 nodes) and it is dependent only on the local geometrical configuration of the grid around the central node. As was mentioned in the previous section, the number of possible local geometrical configurations for a two-dimensional grid is fifteen. This property allows to precompute rows of stiffness matrices for differential operators with constant coefficients. It should also simplify writing software for computing the stiffness matrix in more complicated cases.

4 Conclusions

In this paper we have shown how to implement efficiently high-order finite elements on recursive grids, which opens a way to developing efficient and highly accurate adaptive finite element software.

However, no matter how accurate a method is, it can not be successful in the engineering world if it does not treat curved boundaries as triangular grids do. It is fairly easy to see that a domain with curved boundaries can be covered by a rectangular grid and the boundary approximated by means of special elements in the computational cells intersecting with the boundary. For uniform rectangular grids this approach is wasteful, for the grid cells which fall outside of the computational domain are not used. The latter consideration is much less important for recursive grids, there a simple remedy is to keep the outside cells unrefined and, therefore, few in number. The approach outlined above has been successfully used in [11] with trilinear finite elements. Its extension to high-order finite elements will allow to develop engineering finite element packages based entirely on the *recursive technology.*

This recursive technology can be made viable only by developing software tools that will hide its complexity from the end user. These tools will include grid-handling program libraries, finite element libraries, and specialized development and visualization environments. Object-oriented programming paradigm can be expected to be particularly helpful in treating the great variety of finite elements that will be required, especially for treating the boundary conditions. In [5] we present an interactive programming environment for developing adaptive grid solvers that can serve as a prototype for future software development in this area.

References

[1] S. Adjerid, J.E. Flaherty, P.K. Moore, and Y.J. Wang, *High-order adaptive methods for parabolic systems*, Physica D, 60 (1992), pp. 94-111.

[2] I. Babuška, *Feedback, Adaptivity, and a posteriori Estimates in Finite Elements: Aims, Theory, and Expe-*

rience, in I. Babuška et al.(Eds), Accuracy Estimates and Adaptive Refinements in Finite Element Computations, Wiley, New York, 1986.

[3] Berger, M. J. and Oliger, J., *Adaptive mesh refinement for hyperbolic partial differential equations,* J. Comput. Phys., 53 (1984), pp. 484–512.

[4] Berger, M. J. and Rigoutsos, I., *An algorithm for point clustering and grid generation,* IEEE Trans. Systems Man and Cybernet, 21 (1991), pp. 1278–1286.

[5] G. Kozlovsky, *An Interactive Programming Environment for Solving Partial Differential Equations Using Adaptive Grids (with application to flame-flow interaction problems)*, Tech. Report #CCNY-CS1291, Computer Science Department, City College of New York (1991).

[6] G. Kozlovsky, *Solving partial differential equations using recursive grids*, Applied Numerical Mathematics, 14 (1994), pp. 165-181.

[7] G. Kozlovsky and G.I. Sivashinsky, *On Open and Closed Tips of Bunsen Burner Flames*, Theoret. Comput. Fluid Dynamics, 6 (1994), pp. 181-192.

[8] W.C. Rheinboldt and C.K. Mesztenyi, *On a data structure for adaptive finite element mesh refinements*, ACM Trans. Math. Software, 2 (1980), pp. 166–187.

[9] G. Strang and G.J. Fix, *An Analysis of the Finite Element Method*, Prentice Hall, Englewood Cliffs, NJ, 1973.

[10] J.F. Thompson, *Dynamically-adaptive grids and partial differential equations*, Applied Numerical Mathematics, 1 (1985), pp. 3-27.

[11] D.P. Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samant, and J.E. Bussoletti, *A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics*, J. Comput. Phys., 92 (1991), pp. 1–66.