Flow modelling using quadtrees and multigrid technique

by Csaba Gáspár
Széchenyi István College, Dept. of Mathematics
P.O.Box 701, H-9007 Győr, Hungary

Abstract

The self-adaptive quadtree algorithm is applied to flow modelling and related problems. The algorithm generates a non-equidistant computational grid with multilevel local refinements and surprisingly low computational cost. On the resulting cell system, proper finite difference schemes can be defined which are simpler that the usual finite element discretisation. The structure of the cell system makes it possible to build up a natural multigrid technique for steady problems. The same grid is used also to perform a scattered data interpolation of the initial data (depth, transmissibility etc.) The procedure is illustrated through flow problems arising in lake modelling.

Introduction

When a flow problem has to be solved by some numerical technique, one should always find an optimal compromise between the spatial resolution of the applied discretisation and the exactness requirements of the simulation. Making the discretisation finer and finer, the number of the introduced unknowns becomes higher and higher, which makes the simulation slower. On the other hand, the nature of the original problem often requires fine discretisation only on a small part of the flow domain, and allows coarser spatial resolution on other subregions.

This problem can, of course, be handled by using curvilinear coordinates or a proper finite element mesh. However, the grid generation results in an additional problem, which may be unconvenient, if the original problem requires several discretisation steps, like in some Lagrangian techniques. At the same time, the transformed differential equations become more complicated, which makes the simulation more difficult. As to the initial data (such as depth, transmissibility), it should be pointed out that they are practically never given on the applied computational structure (grid, mesh), so their interpolation results in another subproblem, which may be often almost as time consuming as the solution of the original problem.

In this paper we present a self-adaptive technique which seems optimal from the above points of view. The computational grid is a non-uniform, non-equidistant but Cartesian grid generated by the so-called quadtree algorithm. This algorithm has been extensively used in certain data structures e.g. in GIS as well as in the "unstructured" grid generation, but not so often in the much simpler finite difference methods. The algorithm is based on the systematic subdivision of an initial square, and makes it possible to create local refinements in an easy and well-controllable way in the subregions of greater interest. Thus, despite the possibly very fine discretisation in some subregions, the number of the introduced unknowns can be kept under an acceptable limit. Based on this type of grids, the data interpolation, the discretisation and the solution can be performed in the same computational structure, which allows also the use of the highly economic multigrid techniques. In the followings, we try to show the use of this technique, overviewing the whole modelling procedure from the initial data handling until the solution methods with illustrations from lake modelling.

1 The Quadtree Grid Generation

First, we briefly recall the fundamental algorithm. For details, see [1, 2].

Let Ω_0 be an initial square (e.g. the unit square) and let S_0 be a finite set of points contained in Ω_0. We call this initial pair (Ω_0,S_0) the 0th level of subdivision. The quadtree algorithm is a recursively defined procedure which operates on a pair (Ω,S) as follows:

Step 1: If the number of the points of S is less than or equal to a prescribed number N_0, or the level of subdivision has reached a prescibed number L_0, then the algorithm is finished at the current level. Otherwise, go to Step 2.

Step 2: Divide Ω into four congruent subsquares Ω_1, Ω_2, Ω_3, Ω_4 and define the subsets S_1, S_2, S_3, S_4 by $S_k:=S\cap\Omega_k$ (k=1,2,3,4).

Step 3: Increase the subdivision level by 1 and repeat the algorithm from Step 1 with the pairs (Ω_1,S_1), (Ω_2,S_2), (Ω_3,S_3) and (Ω_4,S_4).

In Step 3, the algorithm invokes itself, i.e. it is recursive. The implementation is especially simple in the programming languages which allow the recursive subroutine calls.

The algorithm results in a non-uniform, non-equidistant but Cartesian "grid" i.e. cell system. It is clear that the spatial "resolution" of this cell system follows the spatial density distribution of the points of S_0. Thus, the structure of the cell system can be controlled by the set S_0, therefore we refer to S_0 as the set of "controlling points". By the proper definition of S_0, local refinements can be generated in the quadtree- (QT-) grid in a simple way.

In spite of the unusual recursive definition, the computational cost of the algorithm is surprisingly low. A rough estimate for the number of the necessary operations is clearly $O(N \cdot L_0)$, where N is the number of the set of controlling points S_0. However, the maximal level of subdivision is often proportional to the number $\log N$: in such cases, the cost of the grid generation is $O(N \cdot \log N)$ only. This makes it possible to apply the algorithm even in Lagrangian techniques, in order to generate a computational grid in each time step: see [3, 4] for details.

Figure 1 shows two typical QT cell systems: the first one is generated by 10 points with N_0:=1; the second one shows that a single points can also generate a QT cell system (with N_0:=0) which "converges" to this single point. In Figure 2, another QT cell system can be seen generated by 2000 points: the structure of the cell system follows exactly the distribution of these points, therefore it is possible to estimate the point density of S_0 by the cell size of the QT grid as pointed out in [5].

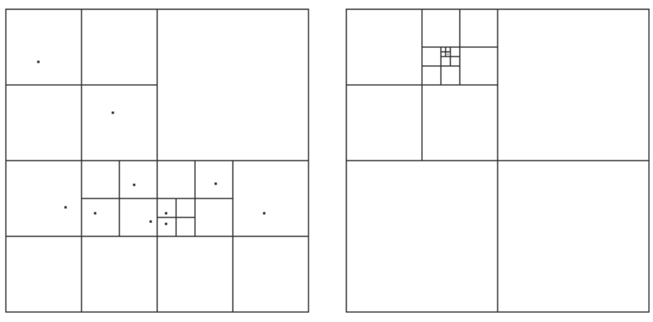

Figure 1. *Two typical QT cell systems*

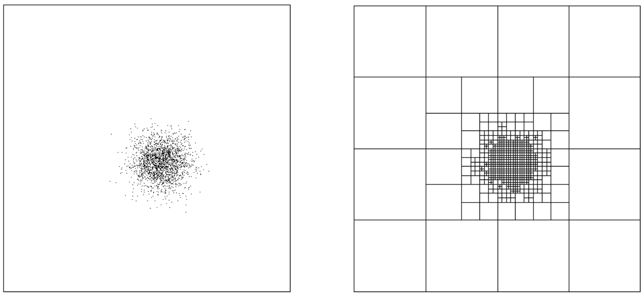

Figure 2. *A QT cell system generated by 2000 points*

Remarks:
1. The algorithm can be obviously extended to arbitrary dimensions as well. In 3D, for example, a cube is to be divided into eight smaller cubes in the second step of the algorithm (*octtree algorithm*). In 1D, the analogous procedure is called *bin-tree* and results in also an ordering algorithm for the points of S_0.
2. In the 2D case, the algorithm can be easily modified to work with *triangles* instead of squares ("tri-tree" structure [6]). The resulting QT cell system consists of triangles, which is often more comfortable in finite element context, see [6] for details.
3. In Step 1, the subdivision criteria can be completed by other conditions as well. This makes it possible to refine the cell systems either in the vicinity of steep gradients of the data (or of the solution itself, which gives a self-adaptive property to the model) or in the neighbourhood of some extreme values. This is usual in shallow water modelling, when the spatial resolution of the discretisation should follow the depth values, see [7].

The QT-grid can be represented by a directed graph with tree-like structure. The root element is the square Ω_0: the remaining elements represent the subsquares and the subdivision procedure is represented by the branches pointing to the subsquares, which are referred to as the *children* of the actually subdivided square. Thus, to any QT cell system, a tree-like data structure can be attached, and usual tree-traversal algorithms can be used for data handling. The leaves of the tree are the cells which are not subdivided further.

We say that the QT-grid is *regular*, if it obeys the "1:2 rule", i.e. the neighbouring leaf-cells have the same size or their ratio is 1:2 (or 2:1). Roughly speaking, this means that the cell structure does not have abrupt changes in cell sizes. Every QT cell system can be made regular either by additional subdivisions or by a slight modification of the recursive algorithm. The structure of the regular QT cell systems is much simpler, and at the same time, still contains a relatively low number of cells. In the followings, therefore, we assume that the applied QT cell systems are always regular.

In the flow modelling it is often necessary to handle more or less complicated geometry. Typically, this is the case in lake or 2D river modelling, when the flow domain is bounded by complicated shorelines and/or it contains islands as well. Consequently, there is a natural need that the applied computational grid is boundary-fitted. The usual technique is often the finite element method or the use of some curvilinear system. However, very comfortable and efficient "quasi-boundary-fitted" grids can be created by the QT algorithm, if it is controlled by the points of the boundary (more procisely, by the points obtained by a sufficiently fine discretisation of the boundary). Figure 3 shows such a boundary-controlled regular QT-grid generated by a circle.

It should be pointed out that in a lot of practical cases, the nature of the problem implies that the solution is much more smooth in the middle of the domain and has rapid changes in the vicinity of the boundary only, e.g. in groundwater flow modelling governed by the Laplace equation. For such problems, the boundary-controlled QT-grids seem especially applicable. We emphasize that though the boundary-controlled QT-grids result in essentially domain-type numerical methods, the grid generation, the discretisation as well as the whole solution procedure can be controlled by the boundary only: in this sense, the QT-based methods can be considered as special *boundary-type methods* as well, like e.g. the traditional Boundary Integral Equation Methods as noted in [8].

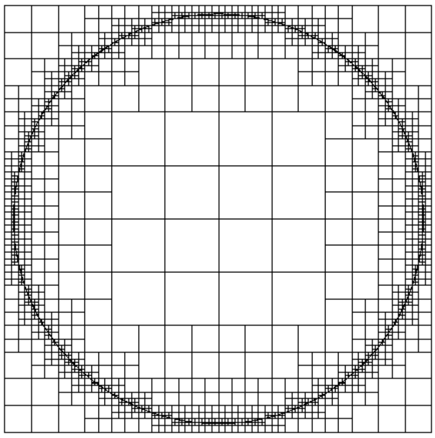

Figure 3. *A regular QT-grid generated by the boundary of a domain*

2 Difference Schemes in QT-Grids

The QT-grids were originally defined in numerical grid generation for constructing finite element meshes [9-11]. Using bilinear trial functions over each cell, it is also possible to derive variational schemes directly on the QT-grids (i.e. without triangularization). However, it is also possible to define much simpler finite difference schemes, where the tegularity of the QT-grid is strongly exploited. The price of simplicity is that these schemes are often of first order only.

In this section we show the construction of the schemes through the example of the simple Laplace operator. The construction can be obviously extended to more general elliplic differential operators as well as to first-order operators. Two main strategies can be distinguished: the *vertex-centered schemes* and the *cell-centered schemes*. An intermediate choice is the use of the *staggered grids* which are introduced later.

Before constructing difference schemes on QT-grids, it should be emphasized that, unlike the traditional uniforms grids, the determination of the *neighbours* of a selected cell is not trivial any more, and requires tree-traversal algorithms based on the graph structure of the grid. However, the computational cost of neighbour finding is relatively low (the determination of all neighbours of all cells requires also $O(N{\cdot}\log N)$ operations only), so that the overall computational work does not increase significantly.

2.1 Vertex-centered schemes

In this approach, the discretized values of the unknown functions to be determined are attached to the vertices of the cells. The schemes are constructed on the basis of Taylor series expansion. Because of the regularity, for each vertex C, only two essentially different cell configurations can occur (see Figure 4) depending on whether or not there exists four congruent cells having C as a common vertex.

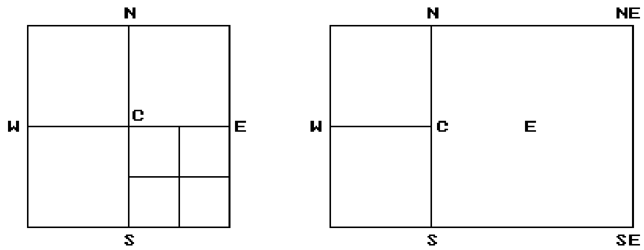

Figure 4. *Vertex-centered schemes, cell configurations*

In the first case, the usual central scheme can be applied:

$$\Delta u_C := \frac{u_N + u_W + u_S + u_E - 4u_C}{h^2} \tag{1}$$

for sufficiently smooth functions u. This can be applied even in the cases when one or more cells have child cells (as shown in Figure 4). The second case is the only "irregular" situation, when one of the neighbouring points taken in the main directions does not exist. As a simple consequence of the regularity, at most one missing point can occur (point E in the example of Figure 4). In this case, let us define a central scheme similar to (1):

$$\Delta u_C := \frac{u_N + u_W + u_S + u_E - 4u_C}{\frac{3}{2}h^2} \tag{2}$$

where the fictitious value u_E is defined by:

$$u_E := \frac{u_N + u_{NW} + u_{SW} + u_S}{4}$$

By a standard Taylor series expansion technique, one can check that the scheme (2) is consistent with the Laplace operator and is at least of order 1.

Remark: In the first case, one can use non-central schemes as well, but they do not remain of second order any longer, and, at the same time, their data structure is a bit more complicated.

In spite of their simplicity, the data structure of the vertex-centered schemes do not fit the graph structure of the QT-grids in a comfortable way since almost every vertex belongs to several cells.

2.2 Cell-centered schemes

Now the discretized values are attached to the cell centers, which perfectly fits the QT-graph structure: each value belongs to one and only one cell. To construct finite difference schemes, two ways seem comfortable:

Cell-centered schemes based on Taylor series expansion

Denote by C an arbitrary central cell. As a consequence of regularity, there are only two essentially different types of cell configurations (see Figure 5):

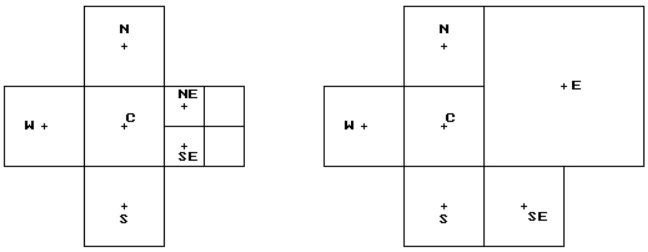

Figure 5. *Cell-centered schemes based on Taylor series expansion: cell configurations*

In the first case, all neighbouring cells are at most as big as the central cell C. Then it is sufficient to take into account these neighbouring cells (the cells N, W, S, NE, SE in Figure 5). Straightforward calculations show that the Laplacian at the center of the cell C can be approximated by the following (at least first order) scheme:

$$\Delta u_C := \frac{20u_N + 24u_W + 20u_S + 16u_{NE} + 16u_{SE} - 96u_C}{21h^2} \tag{3}$$

In the second case, there exists (at least one) neighbouring cell of C which is twice as big as C. Then it is *not* sufficient to take into account the neighbouring cells (the cells N, W, S, E in Figure 5): by this way, the second order mixed derivative cannot be eliminated from the scheme. It is necessary, therefore, to incorporate and additional cell in the scheme, e.g. a neighbour of the bigger cell (SE in Figure 5). Now the following (at least first order) scheme can be obtained:

$$\Delta u_C := \frac{18u_N + 18u_W + 16u_S + 6u_{SE} + 8u_E - 66u_C}{21h^2} \tag{4}$$

For both cases, there exist several additional cell configurations with the same characteristic property. Thanks to the regularity again, only relatively low number of different cases occur:

they are summarized in Figure 6 (not taking into account the geometrically congruent cell configurations). Note that the above schemes are not symmetric in general.

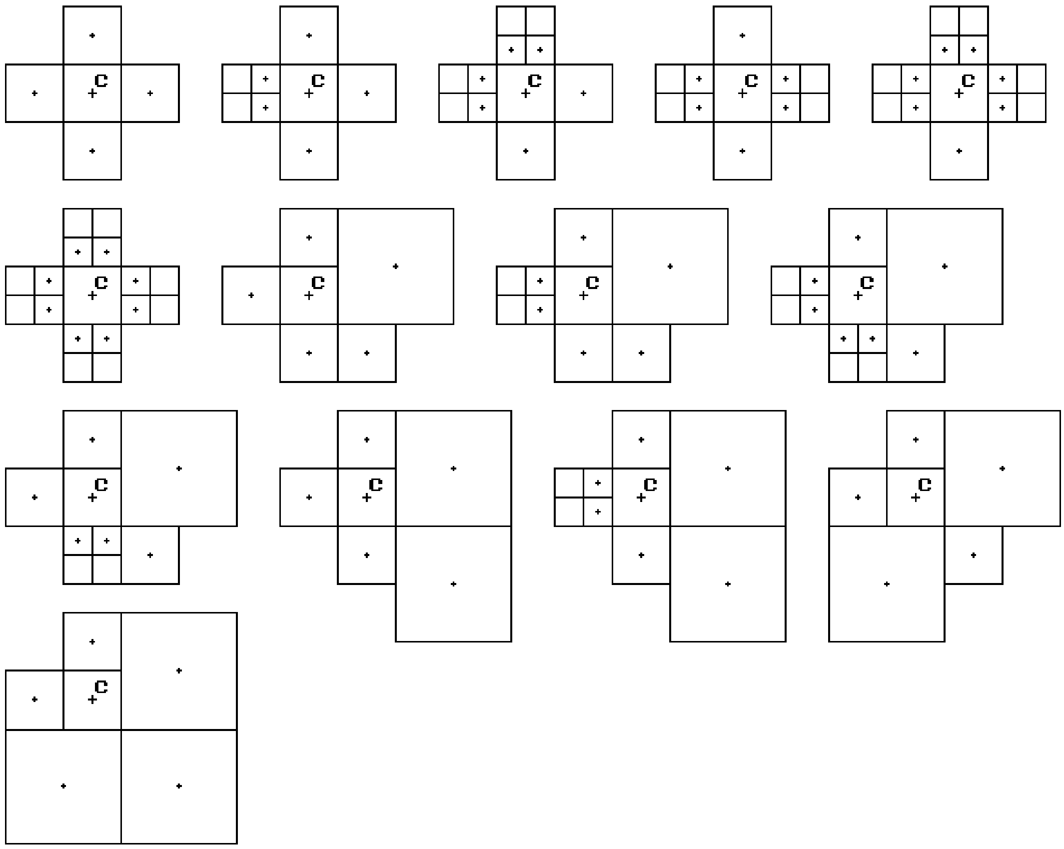

Figure 6. *Cell-centered schemes based on Taylor series expansion: the essentially different cell configurations*

For each cell configuration, a similar difference scheme can be derived: however, the use of this scheme system is not convenient very much, because of the need of recognition of the different cell configurations.

Cell-centered integrated schemes

The simplest schemes can be obtained by integrating the original differential equation (the Laplace equation in our model example) over the cells, which results in *box schemes*. Using Green's formula, we have:

$$\int_C \Delta u \, d\Omega = \int_{\partial\Omega} \frac{\partial u}{\partial n} d\Gamma ,$$

therefore only the fluxes across the cell sides have to be approximated. We show three different approaches, each of them results in symmetric schemes. Denote by h the side length of the cell C.

Scheme 1:
We give an approximation of the flux across the eastern side CE of the cell C: the remaining fluxes can be approximated in an analogous way. Thanks to the regularity again, we have four different cases, but only two of them differ essentially (see Figure 7):

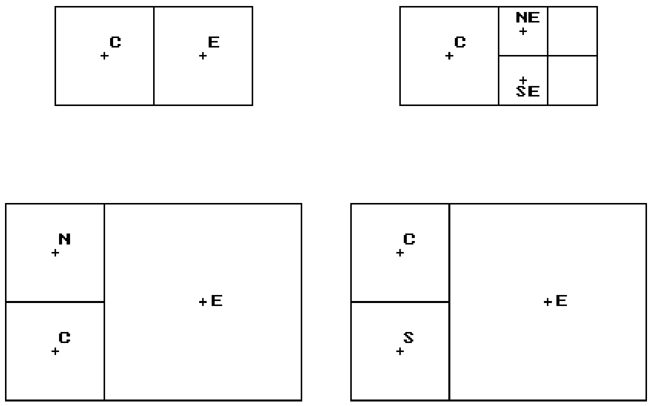

Figure 7. *Cell-centered integrated schemes. Cell configurations for Scheme 1*

Case (a): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := u_E - u_C$$

Case (b): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := \frac{\frac{u_{NE} + u_{SE}}{2} - u_C}{\frac{3}{4}}$$

Case (c): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := \frac{u_E - \frac{u_C + u_N}{2}}{\frac{3}{2}}$$

Case (d): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := \frac{u_E - \frac{u_C + u_S}{2}}{\frac{3}{2}}$$

Scheme 2:
Let us modify the shapes of the cells is such a way that the sides are perpendicular to the line between C and the centers of the neighbouring cells. Such a typical configuration is shown in Figure 8, in which the cell C is distorted to a pentagonal domain.

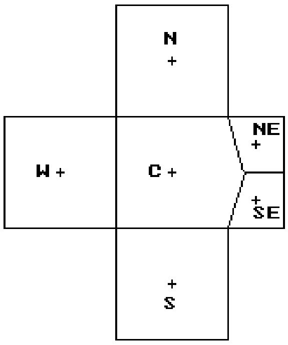

Figure 8. *Cell-centered integrated schemes. Cell configuration for Scheme 2*

Now the fluxes are approximated by the corresponding (not necessarily central) finite differences. From elementary geometry, one can easily obtain that in the case of Figure 8, the length of the eastern cell sides are as follows: $a = h \cdot \frac{\sqrt{10}}{6}$ and the distance between C and both NE and SE equals to $h \cdot \frac{\sqrt{10}}{4}$. Hence, the fluxes across the eastern sides of C are approximated by:

$$\int_{\partial NE} \frac{\partial u}{\partial n} d\Gamma := \frac{2}{3} \cdot (u_{NE} - u_C)$$

and

$$\int_{\partial SE} \frac{\partial u}{\partial n} d\Gamma := \frac{2}{3} \cdot (u_{SE} - u_C)$$

Consequently:

$$\int_{\partial E} \frac{\partial u}{\partial n} d\Gamma := \frac{2}{3} \cdot (u_{NE} + u_{SE} - 2u_C),$$

which coincides with the previous Scheme 1 in case (b). We see that in spite of the different constructions, this approach leads exactly to the same schemes as in the previous method.

However, the cell areas are different, which can yield differences in the right-hand sides of the discretized equations, when solving Poisson equation instead of the Laplace equation.

Scheme 3:
When approximating the fluxes, let us use always *central* differences: if a neighbouring cell has different cell size, let us take the *parent* of the smaller cell, which has, due to the regularity, always the same size as the bigger one. Here we exploit the obvious fact that though not every cell has children, every cell has a parent (except the root cell): see Figure 9 for notations.

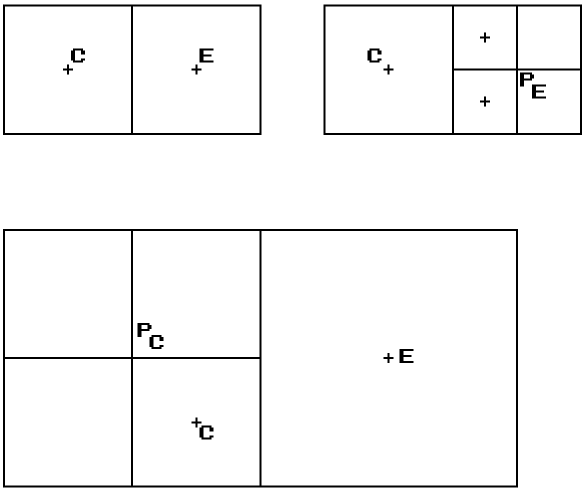

Figure 9. *Cell-centered integrated schemes. Cell configurations for Scheme 3*

Case (*a*): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := u_E - u_C$$

Case (*b*): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := u_{P_E} - u_C$$

Case (*c*): $$\int_{\partial CE} \frac{\partial u}{\partial n} d\Gamma := \frac{1}{2} \cdot (u_E - u_{P_C})$$

Here P_E and P_C denote the parents of the cell E and C, respectively. For the parent cells, the averages of the values corresponding to the child cells are attached (in a recursive sense). In addition to simplicity, this scheme fits also the multigrid techniques in a natural way, see later.

Based on the Bramble-Hilbert-lemma [12], it can be proved that all the above schemes are of first order with respect to the discrete H_0^1-norm (see also [13, 14]). Moreover, they are also stable, therefore they result in convergent, at least first order methods. Without going into details, however, we note that the order of the method has to be handled carefully in QT-context because of the possibly large scale of cell sizes.

3 Multigrid Technique in QT Context

The QT-algorithm results in not only a single computational grid but also a natural context to develop multigrid methods. As it is well known (see [15-17]), to construct a multigrid procedure, a nested *sequence* of grids have to be defined ("fine" and "coarse" grids) which are connected by proper inter-grid transfer operators i.e. *restricitions* and *prolongations*. Of course, the original problem is to be discretized at each grid level.

The QT algorithm *automatically* produces a nested cell system, so that the above grid sequence is always created together with the QT cell system. The individual grids are defined by cutting the QT-graph at the different levels and omitting the cells which are smaller than the cells belonging to this level of subdivision. Figure 10 shows such a nested QT-grid sequence. Here the original problem was the modelling of wind-induced flow patterns in Lake Balaton by solving the simplified shallow water equations (see later). The coarser grids are also QT-grids (therefore not equidistant grids in general), which consist of certain cells of the original QT-grid. Consequently, when constructing restrictions and prolongations, only the data transfer between the parent cell and the corresponding child cells should be defined. We deal with the case of cell-centered schemes only.

Restriction

If the parent cell P has the children C_1, C_2, C_3, C_4, then let the restriction be defined by the simple average:

$$u_P := \frac{u_{C_1} + u_{C_2} + u_{C_3} + u_{C_4}}{4}.$$

Prolongation

Now the values of the parent cells should be distributed to the child cells. The simplest way is the data transfer without modification:

$$u_{C_1} := u_{C_2} := u_{C_3} := u_{C_4} := u_P$$

A more smooth prolongation can be defined by a weighting technique (see Figure 11). Here the regularity is exploited again, since this assures the existence of the neighbouring cells N, W, E, S (provided that the cell C does not lie along the boundary of the domain of the original problem).

$$u_{NW} := \frac{2u_C + u_N + u_W}{4} \qquad u_{SW} := \frac{2u_C + u_S + u_W}{4}$$

$$u_{NE} := \frac{2u_C + u_N + u_E}{4} \qquad u_{SE} := \frac{2u_C + u_S + u_E}{4}$$

This prolongation, as it can be easily checked, keeps the constant and the linear functions unchanged (the previous one does the same for the constant functions only).

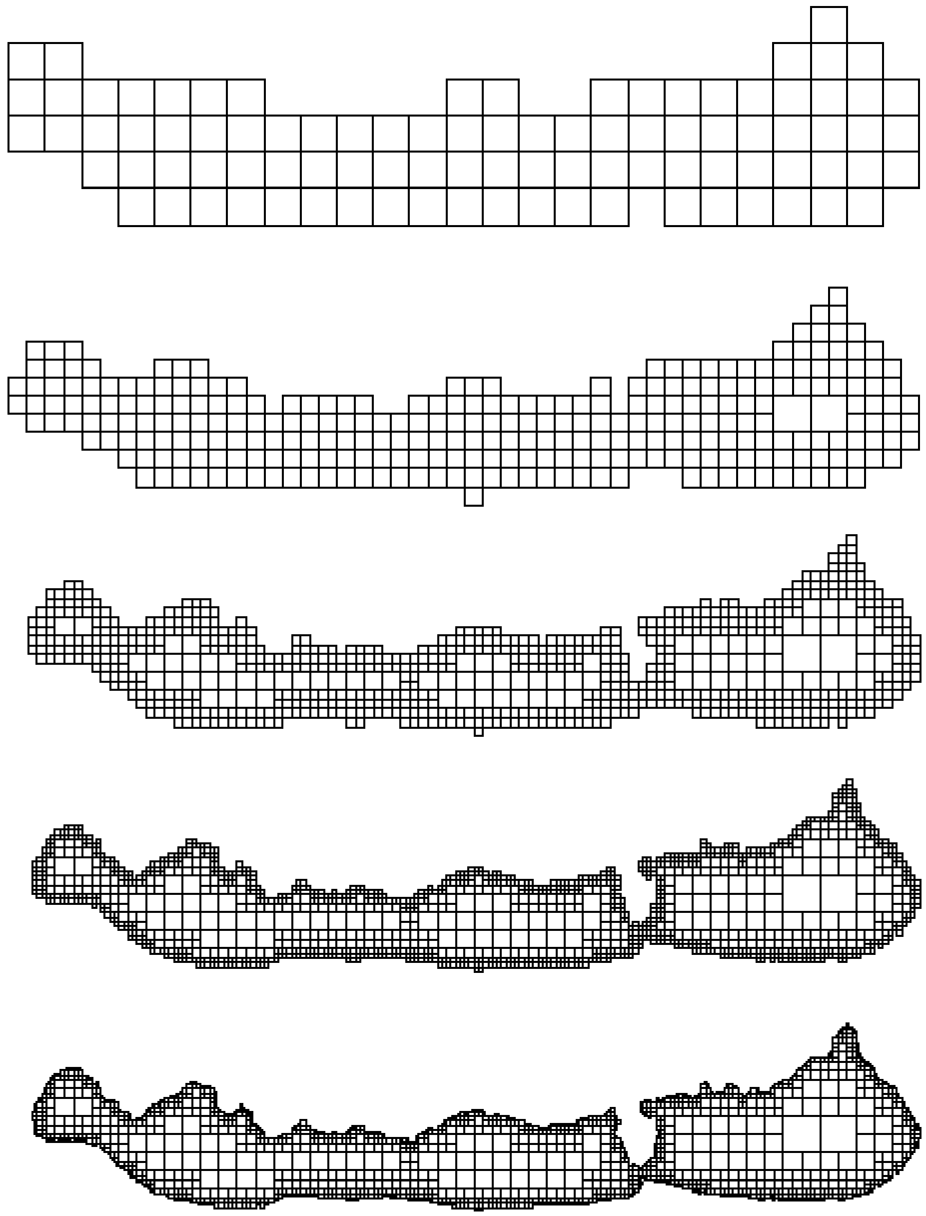

Figure 10. *QT-grid sequence for multigridding*

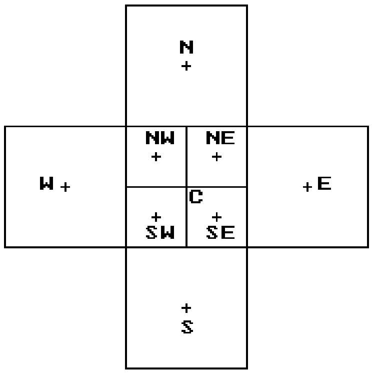

Figure 11. *Prolongation in QT-grids by weighting*

Having defined the grid sequence as well as the restrictions and the prolongations, one can build the multigrid technique in the usual way [3, 4, 14].

4 Scattered Data Interpolation

Before modelling a real flow problem, perhaps the most frequently arising task is the proper definition of the initial data (such as bottom topography, depth, transmissibility etc.) These values are often given in scattered points only, instead of the gridpoints of a computational structure. Therefore, in general, some interpolation technique is needed to produce these values in the gridpoints (or cell centers). Sometimes it is carried out heuristically or by unnecessarily complicated software, or even manually, which can be very exhausting if a huge number of gridpoints (cells) are used. The problem is a typical case of the *scattered data interpolation* problem, which can be, in principle, solved by several ways, see, e.g. [18].

One of the nicest interpolation surfaces can be obtained by the Method of Multiquadrics (MQ, see [19, 20]). Here the interpolation function has the following form:

$$f(\mathbf{x}) := \sum_{j=1}^{N} \alpha_j \cdot \sqrt{|\mathbf{x}-\mathbf{x}_j|^2 + r_j^2}\,, \tag{5}$$

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are the scattered points and $f_1, f_2, \ldots, f_N$ are the corresponding data values. The coefficients $\alpha_1, \ldots, \alpha_N$ have to be chosen in such a way that the interpolation equations

$$f(\mathbf{x}_k) = f_k \qquad (k = 1,2,\ldots,N) \tag{6}$$

are satisfied. The parameters r_j play some optimization role. This method results in an interpolation function with nice smoothness properties (see Figure 12 for an oversimplified example).

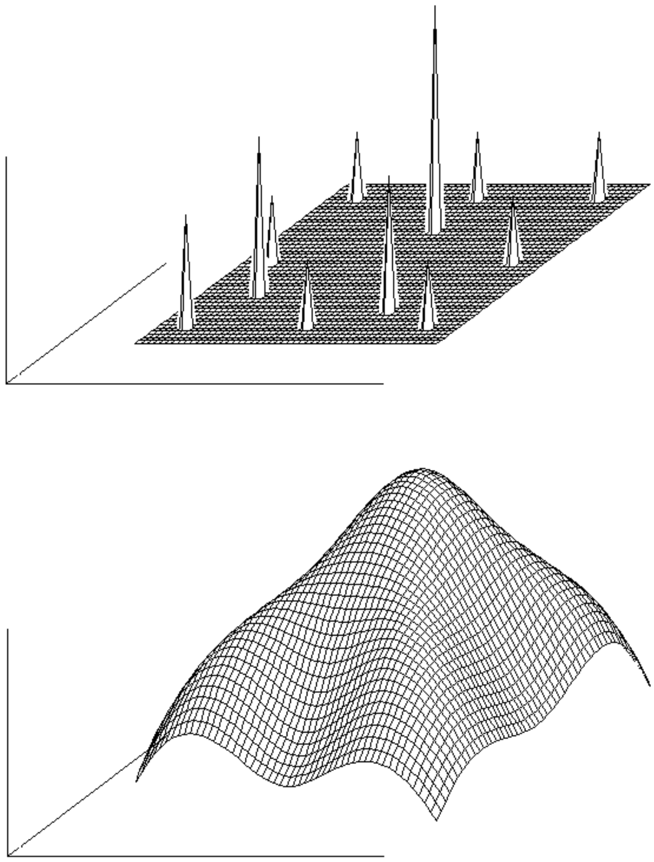

Figure 12. *Scattered data points and interpolation by the Method of Multiquadrics*

Unfortunately, the method requires extremely much computational work, since a large linear algebraic system of equations with a fully populated matrix is to be solved.

Another method is to seek the interpolation function as the solution of the *biharmonic equation*:

$$\Delta\Delta f = 0 \tag{7}$$

supplied with the interpolation equations (6), which can be considered as special "boundary conditions" at the discrete points $\mathbf{x}_k$. (Obviously this condition is incorrect for second order differential equations but not for higher order equations which have continuous fundamental solutions.) This interpolation technique results in also nice surfaces, see Figure 13. This approach makes it possible to use economic multigrid tools in solving the biharmonic equation, therefore, from computational point of view, it is much cheaper than the previous MQ method. The efficiency can be increased further if the biharmonic equation (7) is solved on a QT-grid generated by the scattered points (and using a multigrid method again), see [21]. For solving (7), it is not necessary to develop difference schemes for the fourth-order derivatives, since (7) can be split into a pair of Laplace-Poisson equations:

$$\Delta g = 0, \qquad \Delta f = g \tag{8}$$

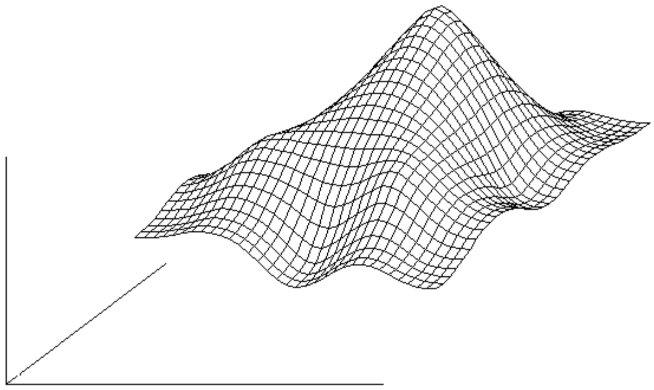

Figure 13. *Scattered data points and interpolation by solving the biharmonic equation. Equidistant grid*

Figure 14 shows the solution of the previous model problem on the QT-grid generated by the scattered points. The solution essentially coincides with the previous biharmonic solution computed on an equidistant grid, but requires even less computational work. It should be pointed out, moreover, that if the grid generation is controlled not only by these interpolation points but also by the other controlling points used in the QT-grid generation of the original flow problem, then the interpolation gives the interpolated values directly on the computational grid.

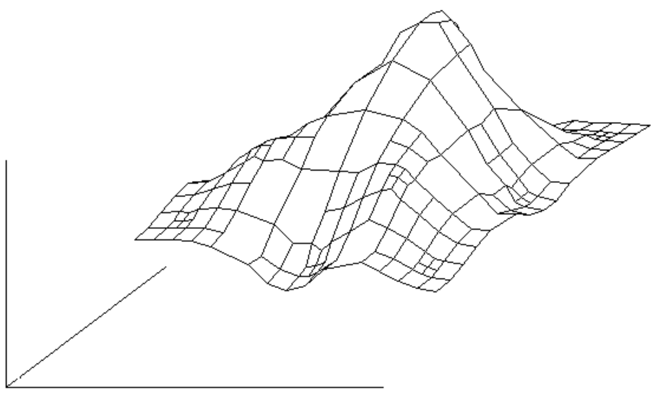

Figure 14. *Scattered data points and interpolation by solving the biharmonic equation. QT-grid*

5 Application to Flow Modelling

We illustrate the capabilities of the QT – Multigrid Method outlined above through some examples. Our goal is to show the applicability of the numerical model to different problems: however, we do not compare the numerical results with physical measurements, which is a quite different problem.

5.1 2D Navier-Stokes equations

In two dimensions, the Navier-Stokes equations can be significantly simplified by introducing the vorticity function ω and the stream function ψ by

$$\omega := \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \qquad u = -\frac{\partial \psi}{\partial y}, \qquad v := \frac{\partial \psi}{\partial x},$$

where $\mathbf{u} = (u,v)$ denotes the velocity field. The functions ω, ψ satisfy the following system of equations:

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \operatorname{grad}\omega - \nu \cdot \Delta\omega = 0 \tag{9}$$

$$\Delta\psi = \omega \tag{10}$$

which does not contain the pressure. Equation (9) is a (nonlinear) transport equation for the vorticity ω. Equation (10) is always a simple Poisson equation which contains no derivatives with respect to time.

To solve Equation (9), Lagrangian techniques are often used. In this approach, the vorticity is approximated by a finite sum of Dirac distributions (called "vortex particles"):

$$\omega(t,\mathbf{x}) = \sum_{k=1}^{N} \omega_k \cdot \delta(\mathbf{x} - \mathbf{x}_k(t)).$$

Using a coordinate system which moves together with the flow, the convective derivatives in (9) are eliminated, and Equation (9) is reduced to a diffusion type problem, which can be handled also by a Lagrangian technique, namely, the random walk method.

Equation (10) requires a completely different method. It can be solved either by some grid-free technique such as the fast multipole method [22, 23] or by applying some finite difference discretisation. In this way, perhaps the most efficient technique is to use a QT-grid *generated by the vortex particles* and speeded up by the above mentioned multigrid procedure. See [4] for details.

5.2 Wind-Driven Circulations in Shallow Lakes

Free surface flows in shallow waters (in lakes, rivers etc.) can be sufficiently described by the shallow water equations, which are the vertically integrated formulations of the Navier-Stokes equations taking into account the external forces (wind friction, bottom friction). This leads to the following equations (neglecting the Coriolis force), see [24]:

$$\frac{\partial \eta}{\partial t} + \frac{\partial p}{\partial x} + \frac{\partial q}{\partial y} = 0$$

$$\frac{\partial p}{\partial t} + \frac{\partial}{\partial x}\left(\frac{p^2}{h}\right) + \frac{\partial}{\partial y}\left(\frac{pq}{h}\right) + gh\frac{\partial \eta}{\partial x} + Ap - \nu \cdot \Delta p = B \cdot |\mathbf{W}| \cdot W_x \tag{11}$$

$$\frac{\partial q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{pq}{h}\right) + \frac{\partial}{\partial y}\left(\frac{q^2}{h}\right) + gh\frac{\partial \eta}{\partial y} + Aq - \nu \cdot \Delta q = B \cdot |\mathbf{W}| \cdot W_y$$

where p, q are the specific discharges: $p=uh$, $q=vh$, $\mathbf{W}$ denotes the wind vector, η is the water surface elevation, h is the water depth and ν is the viscosity coefficient. The term A describes the bottom friction. Assuming quadratic bottom fricition, A can be expressed as

$$A = \frac{g}{k^2 h^{4/3}} \cdot |\mathbf{u}|, \tag{12}$$

where k is the Manning roughness coefficient.

A more simplified approximation of the bottom friction is as follows:

$$A = \frac{r}{h} \tag{13}$$

where r is assumed to be constant.

If we are interested in wind-driven circulations in lakes, the terms due to the convective acceleration as well as the diffusion can often be neglected. Then from Equation (11), we obtain:

$$\frac{\partial \eta}{\partial t} + \frac{\partial p}{\partial x} + \frac{\partial q}{\partial y} = 0$$

$$\frac{\partial p}{\partial t} + gh\frac{\partial \eta}{\partial x} + Ap = B \cdot |\mathbf{W}| \cdot W_x \tag{14}$$

$$\frac{\partial q}{\partial t} + gh\frac{\partial \eta}{\partial y} + Aq = B \cdot |\mathbf{W}| \cdot W_y .$$

Steady-state solution

Equation (14) can be simplified further if the steady solutions are seeked i.e. all time derivatives identically vanish. In this case the stream function ψ can be introduced by

$$p = \frac{\partial \psi}{\partial y}, \qquad q = -\frac{\partial \psi}{\partial x},$$

and Equation (14) implies that

$$\operatorname{div}\frac{A}{h}\operatorname{grad}\psi = B\cdot\left[\frac{\partial}{\partial y}\left(\frac{|\mathbf{W}|\cdot W_x}{h}\right)-\frac{\partial}{\partial x}\left(\frac{|\mathbf{W}|\cdot W_y}{h}\right)\right] \tag{15}$$

Assuming linear bottom friction of the form (13), Equation (15) is a *linear*, elliptic partial differential equation, where the depth h is approximated by the difference of a *fixed* average surface level and the bottom level.

Equation (15) can be solved on a QT-grid generated by the shoreline and possibly some extra controlling points located at the deeper areas. In solving Equation (15), a multigrid method can be applied without difficulty, see [25]. Figure 15 shows the computational grid generated for a flow modelling in the Finnish lake named Pyhäjärvi.

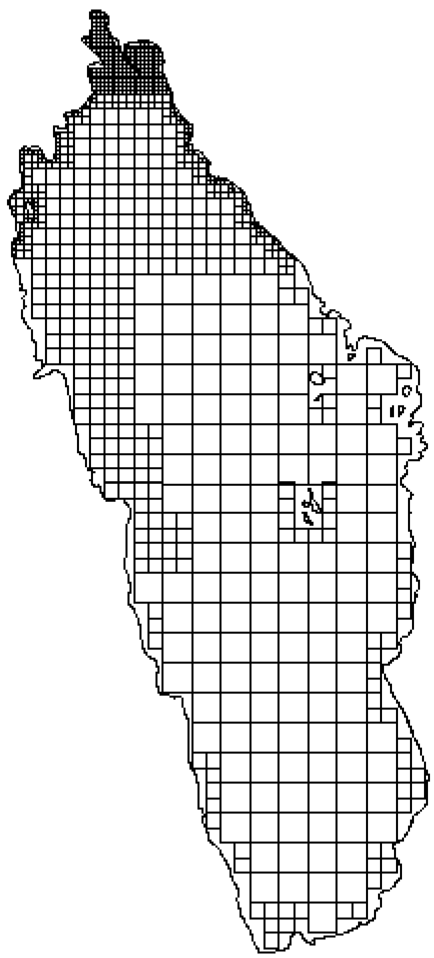

Figure 15. *QT-grid for Lake Pyhäjärvi, Finland*

The main geometrical data of the lake are as follows: the length and the width of the lake are approximately 25 and 10 km, respectively. The maximal level of subdivision was 8, which allowed the use of the cell sizes 100 m at the finest level. Figure 16 shows a typical steady flow pattern induced by an east wind. Only the northern part of the lake is displayed, where the discretisation was forced to be finer.

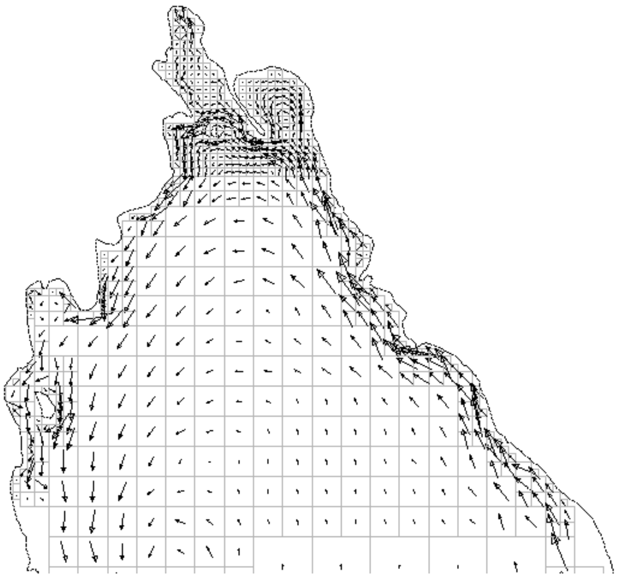

Figure 16. *Steady wind-driven circulations in Lake Pyhäjärvi, Finland*

Unsteady solution

If the flow problem is not steady, the full system of Equation (14) has to be solved simultaneously. Using uniform (rectangular or curvilinear) grids, a frequently applied discretisation technique is the use of *staggered grids*, where the values of the unknown functions are attached to different points of the computational cells. Usually the p-values are attached to the vertical, while the q-values are attached to the horizontal *side centers*. The values of the surface elevation belong to the *cell centers* (see Figure 17):

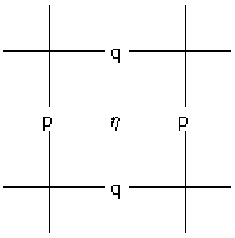

Figure 17. *Staggered grid: discretisation of the shallow water equations*

Following this discretisation strategy and applying a constant time step Δt, Equation (14) can be discretized in the following way (see the cell C in Figure 17 for notations):

$$\eta^{(n+1)} = \eta^{(n)} - \Delta t \cdot \left(\frac{p_E^{(n)} - p_W^{(n)}}{\Delta x} + \frac{q_N^{(n)} - q_S^{(n)}}{\Delta y} \right)$$

$$p^{(n+1)} = \frac{1}{1 + \Delta t \cdot A^{(n)}} \cdot \left[p^{(n)} + \Delta t \cdot \left(B \cdot |\mathbf{W}^{(n)}| \cdot W_x^{(n)} - g h^{(n)} \frac{\partial \eta^{(n)}}{\partial x} \right) \right] \tag{16}$$

$$q^{(n+1)} = \frac{1}{1 + \Delta t \cdot A^{(n)}} \cdot \left[q^{(n)} + \Delta t \cdot \left(B \cdot |\mathbf{W}^{(n)}| \cdot W_y^{(n)} - g h^{(n)} \frac{\partial \eta^{(n)}}{\partial y} \right) \right].$$

Here the upper indices refer to the time level.

Scheme (16) is essentially explicit, therefore the time step is bounded by a CFL-type stability criterion. With obvious modification, the above staggered grid discretisation can be extended to QT-grid context as well. A typical arrangement of the unknowns is seen in Figure 18.

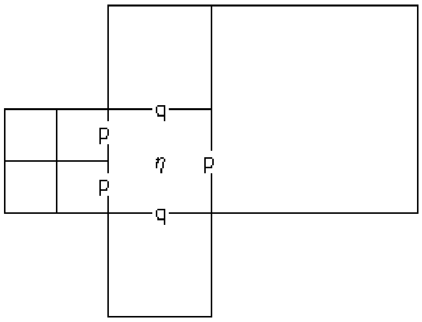

Figure 18. *Staggered QT-grid*

Due to the regularity of the QT-grid, no more complex configuration occurs. The spatial derivatives of η are approximated by the schemes derived for the fluxes, see Scheme 3 in Section 2.

Because of the reduction of the number of unknowns due to the applied QT-grid, the computational cost remains modest, which makes the method competitive compared with the traditional methods, see [25].

6 Conclusions

An efficient method of the flow modelling has been outlined. The method is based on the quadtree algorithm, which produces a non-equidistant, non-uniform but Cartesian cell system as a computational grid. The structure of the grid can be easily controlled, which makes possible the definition of multi-level local refinements. Thus, nice boundary-fitted grids can be created, in spite of the fact that the resulting grid in an essentially Cartesian one. The extreme values and/or the presence of steep gradients can also control the structure of the grid. The algorithm makes it possible to highly reduce the number of unknowns introduced by the numerical model. The data structure of the QT-grids is a tree-like graph, so that the data handling needs tree-traversal algorithms. Simple finite difference schemes as well as multigrid techniques are defined in the QT-grid context. It seems that the marriage of the above two techniques – i.e. quadtree and multigrid – makes this modelling technique uniquely efficient, which is applicable to a large class of different problems including scattered data interpolation, Lagrangian methods, shallow water equations etc.

Acknowledgement

The research was partly supported by the Hungarian National Research Fund (OTKA) under the contract T17323.

References

[1] J.H. Cheng, P.M. Finnigan, A.F. Hathaway, A. Kela, W.J. Schroeder, "*Quadtree-octree meshing with adaptive analysis*", Numerical Grid Generation in Computational Fluid Mechanics '88. (Edited by S. Sengupta, J. Häuser, P.R. Eiseman, J.F. Thompson.) Pineridge Press, 1988

[2] J. Makino, "*Vectorization of a Treecode*", Journal of Computational Physics, Vol. 87, 1990

[3] C. Gáspár, J. Józsa, P. Simbierowicz, "*Lagrangian Modelling of the Convective Diffusion problem Using Unstructured Grids and Multigrid Technique*", Proceedings of the First International Conference on Water Pollution, 3-5 September, Southampton, Great Britain, 1991

[4] C. Gáspár, J. Józsa, "*Two-dimensional Lagrangian Flow Simulation Using Fast, Quadtree-based Adaptive Multigrid Solver*", Proceedings of the 9th GAMM Conference, September 25-27, 1991, Lausanne, Switzerland, Vieweg Verlag, 1991

[5] C. Gáspár S. Szél, „*Application of Unstructured Grids in Monte-Carlo Simulations*", Proceedings of the XXIV IAHR Congress, September 9-13, 1991, Madrid, Spain, 1991

[6] S.∅. Wille, "*A Structured Tri-Tree Search Method for Generation of Optimal Unstructured Finite Element Grids in Two and Three Dimensions*", International Journal for Numerical Methods in Fluids, Vol. 14, 1992

[7] P. Nielsen, O. Skovgaard, "*A Depth-Adaptive Grid Using a Conrol-Function Approach*", Numerical Grid Generation in Computational Fluid Mechanics '88. (Edited by S. Sengupta, J. Häuser, P.R. Eiseman, J.F. Thompson.) Pineridge Press, Great Britain, 1988

[8] C. Gáspár, "*A Multigrid Based Boundary Element Method Without Integral Equations*", Proceedings of the 13th International Conference on Boundary Element Methods, 21-23 August, Tulsa, Oklahoma, U.S.A., 1991

[9] A. Perronnet, *"Some Present Fully Automatic Mesh Generators*", Proceedings of the Fifth International Symposium on Numerical Methods in Engineering. (Edited by R. Gruber, J. Periaux, R.P. Shaw.) Computational Mechanics Publications/Springer-Verlag, 1989

[10] D.P Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samat, J.E. Busoletti, "*A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics*", Journal of Computational Physics, Vol. 92, 1991

[11] A. Evans, M.J. Marchant, J. Szmelter, N.P. Weatherill, "*Mesh Adaptivity with the Quadtree Method*", Numerical Grid Generation in Computational Fluid Dynamics and Related Fields. (Edited by A.S. Arcilla, J. Häuser, P.R. Eiseman, J.F. Thompson.) Elsevier, 1991

[12] J.H. Bramble, S.R. Hilbert, "*Bounds for a class of linear functionals with application to Hermite interpolation*", Numer. Math., Vol. 6., pp. 362-369, 1971

[13] R.E. Ewing, R.D. Lazarov, P.S. Vassilevski, "*Finite Difference Schemes on Grids with Local Refinement in Time and Space for Parabolic Problems. I. Derivation, Stability and Error Analysis*", Computing, Vol. 45, 1990

[14] C. Gáspár, P. Simbierowicz, "*Difference Schemes in Tree-Structured Multigrid Context*", Proceedings of the IX. International Conference on Numerical Methods in Water Resources, 9-12, June, Denver, Colorado, U.S.A., 1992

[15] A. Brandt, "*Multigrid Technique. 1984 Guide with Applications to Fluid Dynamics*", GMD-Studien, Nr. 85, Bonn, Germany, 1984

[16] K. Stüben, U. Trottenberg, *"Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications*", GDM-Studien, Nr. 96, Birlinghoven, Germany, 1984

[17] W. Hackbusch, "*Multi-Grid Methods and Applications*", Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985

[18] R. Franke, "*Scattered Data Interpolation: Test of Some Methods*", Mathematics of Computation, Vol. 38, No. 157, 1982

[19] R.L. Hardy, "*Theory and Applications of the Multiquadric-Biharmonic Method, 20 Years of Discovery 1968-1988*", Computers Math. Applic., Vol 19, No. 8-9, pp. 163-208, 1990

[20] E.J. Kansa, "*Multiquadrics – A Scattered Data Approximation Scheme with Application to Computational Fluid Dynamics I. Surface Approximations and Partial Derivative Estimates*", Computers Math. Applic., Vol 19, No. 8-9, pp. 127-145, 1990

[21] C. Gáspár, P. Simbierowicz, "*Scattered Data Interpolation Using Unstructured Grids*", Proceedings of HYDROCOMP '92 (International Conference on Interaction of Computational Methods and Measurements in Hydraulics and Hydrology), 25-29 May, Budapest, Hungary, 1992

[22] J. Carrier, L. Greengard, V. Rokhlin, "*A Fast Adaptive Multipole Algorithm for Particle Simulations*", SIAM J. Sci. Stat. Comput., Vol. 9, pp. 669-686, 1988

[23] L. van Dommelen, E.A. Rundensteiner, *"Fast, Adaptive Summation of Point Forces in the Two-dimensional Poisson Equation*", Journal of Computational Physics, Vol. 83, pp. 126-147, 1989

[24] M.B. Abbott, D.R. Basco, *"Computational Fluid Dynamics*". An Introduction for Engineers. Longman/Wiley, 1989

[25] C. Gáspár, J. Józsa, J. Sarkkula, *"Shallow Lake Modelling Using Quadtree-based Grids*", Proceedings of the X. International Conference on Computational Methods in Water Resources, 19-22, July, 1994, Heidelberg, Germany, 1994