9

Using Self Organizing Maps for 3D surface and volume adaptive mesh generation

Olga Nechaeva
Novosibirsk State University, NSU-Intel Laboratory
Russia

1. Introduction

Adaptive mesh methods are commonly used to improve the accuracy of numerical solution of problems without essential increase in the number of mesh nodes (Lebedev et al., 2002). Within the scope of all adaptive mesh methods, there is an important class of methods in which the mesh is an image under an appropriate mapping of a fixed mesh over a computational domain (Bern & Plassmann, 1999).
Most of widely used conventional methods from the above class, such as equidistribution method (Shokina, 2001), Thompson's method (Thompson et al., 1985), elliptic method (Liseikin, 1999), etc. determine the mapping by solving a complicated system of nonlinear partial differential equations (PDEs). This often leads to significant difficulties. First, the convergence of numerical solution of these PDEs highly depends on an initial mesh, requires fixing boundary mesh nodes beforehand and imposes quite strong limitations on the properties of mesh density function (Khakimzyanov et al., 2001). Second, efficient parallelization of solvers for the PDEs meets overwhelming difficulties. Finally, the PDEs for mesh construction are not universal and need to be proposed for 1D, 2D or 3D spaces specifically. The complexity of numerical solution of these PDEs essentially grows with increasing the dimensionalities (Khakimzyanov et al., 2001). Moreover, there is no methods and techniques in the above mentioned class that can provide a fully automatic adaptive mesh construction in 3D case.
This chapter demonstrates the great ability of the Kohonen's Self Organizing Maps (SOM) (Kohonen, 2001) to perform high quality adaptive mesh construction. Since the SOM model provides a topology preserving mapping of high-dimensional data onto a low-dimensional space with approximation of input data distribution, the proposed mesh construction method uses the same algorithms for different dimensionalities of a physical domain that proves its universality.
In our investigation, the classical SOM model has been studied and modified in order to overcome border effect and provide topology preservation. Based on the ideas in (Nechaeva, 2006), the composition of SOM models of different dimensionalities has been proposed which alternates mesh construction on the border and inside a physical domain. It has been shown that the SOM learning algorithm can be used as a mesh smoothing tool. All the algorithms has been implemented using the GeomBox (Bessmeltsev, 2009) and

GeomRandom (Nechaeva, 2009) packages and tested on a number of physical domains. The quality of resulting meshes is acceptable according to the commonly used quality criteria.
In order to support this alternative approach to mesh generation, a Theorem of Correspondence is proved, that states that goals of traditional PDE approach to construction of adaptive meshes from the considered class are equivalent to the goals of learning for Self Organizing Maps.
The obtained results showed that the neural network approach provides us a highly parallelizable technique (Nechaeva, 2005) for automatic construction of qualitative adaptive meshes and possesses the following properties: (1) due to the self organizing principles the algorithm transforms the mesh automatically, starting with arbitrary initial nodes positions, and does not require to fix the boundary nodes beforehand; (2) stochastic nature of the algorithm enables us to illuminate any limitations on the mesh density function; (3) internal parallelism of the method allows us to parallelize the mesh construction process, taking into account the requirements on the parallel implementation of a problem to be solved on the mesh; (4) the method uses the same algorithms for different dimensionalities of a physical domain that proves the universality of the proposed method.

2. Adaptive mesh construction: problem statement

In order to emphasize that the neural network approach is universal from the point of view of space dimensionality, the problem statement, methods and algorithms are formulated here for arbitrary dimensionalities.
Let G be a physical domain in a Euclidean space R_G^n with physical coordinates $x = (x^1,...,x^n)$. An adaptive mesh $G_N = \{x_1,...,x_N\}$ is to be constructed over G, where $x_i = (x_i^1,...,x_i^n) \in G$, $i = 1,...,N$ are the mesh nodes. Let Q be a computational domain in a Euclidean space R_Q^k, $k \le n$ with coordinates $q = (q^1,...,q^k)$. A mesh $Q_N = \{q_1,...,q_N\}$ is fixed over Q, where $q_i \in Q$, $q_i = (q_i^1,...,q_i^k)$, $i = 1,...,N$. Let a minimal distance among all pairs of nodes in Q_N be equal to d_Q. Usually, the fixed mesh Q_N is rectangular and uniform, then d_Q is just the distance between neighboring nodes. Also, let us denote by $B_\gamma(q)$ a bounded neighborhood of the point q in Q, where γ is a radius of the neighborhood, i.e. $B_\gamma(q) = \{p \in R_Q^k \mid d(p,q) \le \gamma\}$, where $d(\cdot,\cdot)$ is the Euclid distance.
The desired density of an adaptive mesh is given by a mesh density function $w : G \to \mathrm{R}^+$. Density of the mesh G_N is to approximate the function w in a sense of the equidistribution principle (Shokina, 2001). According to this principle, the product of a mesh cell area and the value of w, associated with this cell, should be the same for all mesh cells. As the consequence, the greater the value of w, the smaller the corresponding cell area and, then, the higher the density of the adaptive mesh.
The goal is to find a mapping of Q onto G which transforms the mesh Q_N into the adaptive one G_N with the given mesh density. The method of mapping determination is required to ensure that the boundary nodes of Q_N are automatically transformed into the nodes distributed along the border of G. At this point, the proposed problem statement differs from the traditional one where one needs to have boundary nodes already distributed along

the border (Shokina, 2001). Let Nb be the number of boundary nodes, and Nint be the number of the interior ones, $N_b \cup N_{\text{int}} = \{1,...,N\}$ and $N_b \cap N_{\text{int}} = \varnothing$.

3. Correspondence between adaptive meshes and Self Organizing Maps

The SOM is a neural network model that is able to perform a mapping from input to output space with topology preservation and approximation of input data distribution. When applying the SOM model for adaptive mesh construction, the input space, which contains the physical domain, is R_G^n and the output space is R_Q^k.

The SOM model can be considered as a triplet < *M*, *H*, *Alg* >: map of neurons (*M*), training set (*H*) and learning algorithm (*Alg*). The map of neurons is the set of neurons $M = \{e_1,...,e_N\}$ where each neuron has a location assigned by coordinates of the fixed node q_i in the output space R_Q^k. The number of neurons in the map *M* is equal to the number of mesh nodes *N*. It means that each *i*-th mesh node corresponds to the neuron e_i (Nechaeva, 2004). There is a weight vector associated with each neuron. Weight vector of the neuron e_i is an element of input space and assigned by the coordinates of x_i within the physical domain *G*. These coordinates can be found by the learning algorithm *Alg*. Therefore, the neuron is a pair $e_i = (q_i, x_i)$.

In order to obtain the desired distribution of x_i over *G* in a sense of equidistribution principle, a training set is to be a sample of the probability distribution $p(x)$ being equal to the normalized mesh density function $w(x)$:

$$p(x) = \frac{w(x)}{\int\limits_G w(x)dx}, \tag{1}$$

Let $H = \{y_1,...,y_T\}$ be a training set corresponding to (1), where *T* is the set size, $y_t \in G$, $t = 1,...,T$.

A basic SOM learning algorithm proposed by Kohonen (Kohonen, 2001) (formulated in terms of "mesh nodes"), where $t_{st} = 1$ and $t_{fin} = T$, is the following.

The procedure Alg.

1. Set arbitrary initial locations of mesh nodes $x_i(0)$ for all $i = 1,...,N$.
2. Repeat the following operations at each iteration $t = t_{st},...,t_{fin}$:
   a. Take the next vector y_t from the training set *H*.
   b. Calculate the Euclidean distances $d(\cdot,\cdot)$ between y_t and all nodes $x_i(t)$ and choose the node $x_m(t)$ which is closest to y_t, i.e.

$$m = m(y_t) = \arg\min_{i=1,...,N} d(y_t, x_i(t)). \tag{2}$$

   The node $x_m(t)$ is called a *winner*.

   c. Adjust locations of mesh nodes using the following rule:

$$x_i(t+1) = x_i(t) + \theta_{q_m}(t,q_i)\ (y_t - x_i(t)), \tag{3}$$

for all $i = 1,...,N$, where $\theta_{q_m}(t,q_i) \in [0, 1)$ is a *learning rate*.

At each iteration *t*, mesh nodes move towards the random point y_t. The magnitude of nodes displacements is controlled by the learning rate $\theta_{q_m}(t,q_i)$.

4. Strategy of learning rate selection

The learning rate $\theta_{q_m}(t,q_i)$ plays a crucial role in the SOM learning algorithm as it directly influences the quality of resulting adaptive meshes and speed of mesh construction.

Unlike usual approach, according to which the algorithm terminates once mesh nodes displacements are small enough, we propose a new strategy where the number of iterations *T* is to be fixed beforehand proportional to *N*, and the learning rate is to be scaled in such a way that all nodes are frozen after *T* iterations. The fact that the number of iterations *T* should be a function of number *N* of mesh nodes follows from the point that we need to have enough vectors in the training set for providing each mesh node with a possibility to move several times. For example, we obtained an acceptable quality of adaptive meshes if the number of iterations is 10 times greater than *N*, i.e. $T = 10N$. It also means that in average each mesh node becomes a winner about 10 times during the iteration process. Therefore, we think that it is incorrect to talk about the number of iterations without mentioning the number of mesh nodes. For example, for our learning rate: $T = 4000$, if $N = 20 * 20$; $T = 90000$, if $N = 30 * 30 * 30$; and so on.

The above technique showed good results in 1D, 2D and 3D cases and is very convenient in use since we can directly control the speed of learning process. The learning rate proposed in this Section is independent of a physical domain and mesh density function.

The learning rate $\theta_{q_m}(t,q_i)$ is a function, which takes its values from the interval [0, 1) and has a view of a product of two functions: learning step and learning neighborhood:

$$\theta_{q_m}(t,q_i) = \delta(t)\ \eta_{q_m}(t,q_i). \tag{4}$$

The learning step $\delta(t)$ is a decreasing function of time and it controls the overall size of nodes displacements at each iteration (Kohonen, 2001), (Fritzke, 1997), (Nechaeva, 2005). Based on experiments, we selected the following function for the learning step: $\delta(t) = t^{-0.2}\chi(t)$, $t = 1, ..., T$, where $\chi(t) = 1 - e^{5(t-T)/T}$. The function $\chi(t)$ is used to make the power member of $\delta(t)$ go down to zero.

Every two neurons q_m and q_i in the map *M* are connected by a lateral connection with a strength being assigned by the neighborhood function $\eta_{q_m}(t,q_i)$. This function has a shape of Gaussian but is transformed for convenience into the following view:

$$\eta_{q_m}(t,q_i) = s^{\left(\frac{d(q_m,q_i)}{r(t)}\right)^2}, \tag{5}$$

where a constant $s \in (0,1)$ is close to zero and fixed beforehand, *r*(*t*) is a learning radius at the iteration *t*. The shape of the function (5) is shown in Fig. 1.
The strength of lateral connections between the neuron e_m and all neurons $e_i \in M$, located inside the neighborhood $B_{r(t)}(q_m)$ of radius *r*(*t*), is less than *s*. The function for lateral connections (5) satisfies the following properties.

Properties

a. $\forall q_i \in B_{r(t)}(q_m) \Rightarrow \eta_{q_m}(t,q_i) \geq s$.

b. $\eta_{q_m}(t,q_m) = 1$;

c. if $d(q_m,q_i) = r(t)$, then $\eta_{q_m}(t,q_i) = s$;

d. lateral connection is symmetric: $\eta_{q_m}(t,q_i) = \eta_{q_i}(t,q_m)$.

e. $\eta_{q_m}(t,q_i) = \eta(d) = s^{\left(\frac{d}{r(t)}\right)^2}$ is a decreasing function of $d = d(q_m,q_i)$.

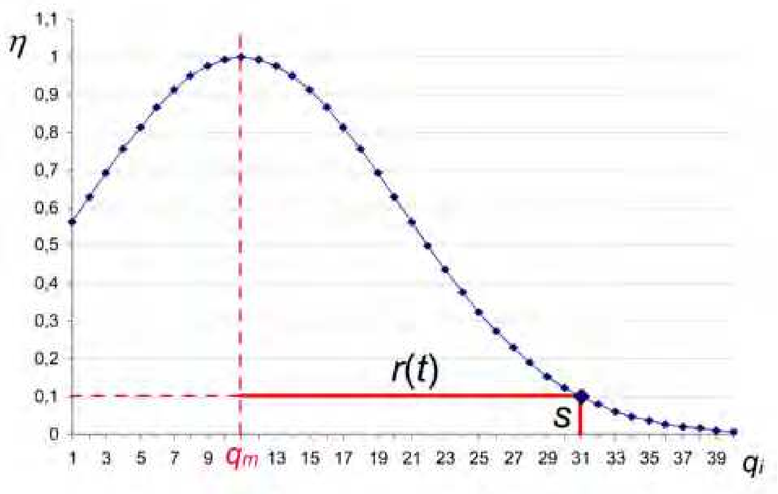


Fig. 1. The shape of the function $\eta_{q_m}(t,q_i)$ for lateral connections between neurons.

As a result, the winner takes the maximum displacement at each iteration. The greater the distance between *i*-th neuron and the winner in the computational domain, the less the displacement of this neuron within the physical domain. When implementing the SOM algorithm, if *s* is small enough, neurons for which $d(q_m,q_i) > r(t)$ can be disregarded during adjustment step of the algorithm with preserving the accuracy, since sizes of the displacements are less than *sδ*(*t*).
The function for lateral connections is responsible for the quality of mesh, e.g. mesh smoothness, shapes of mesh cells, etc. The learning radius *r*(*t*) is a decreasing function of *t* with fixed values at first and last iterations: *r*(1) and *r*(*T*), where $r(1) > r(T)$. Based on experiments, we selected the learning radius as $r(t) = r(T) + \chi(t)\left(r(1)0.05^{t/T} - r(T)\right) t^{-0,25}$.
The values *r*(1) and *r*(*T*) has to be selected in such a way that at first iteration all neurons fit into $B_{r(1)}(q_i)$ for any $i = 1,\ldots,N$ and at last iteration the neighborhood $B_{r(T)}(q_m)$ contains only the node q_m and its closest neighbors.

Thus, there are only few free parameters in the proposed learning rate, among them are $r(1)$, $r(T)$ and T. In Fig. 2, diagrams of all functions are shown which take part in the learning rate.

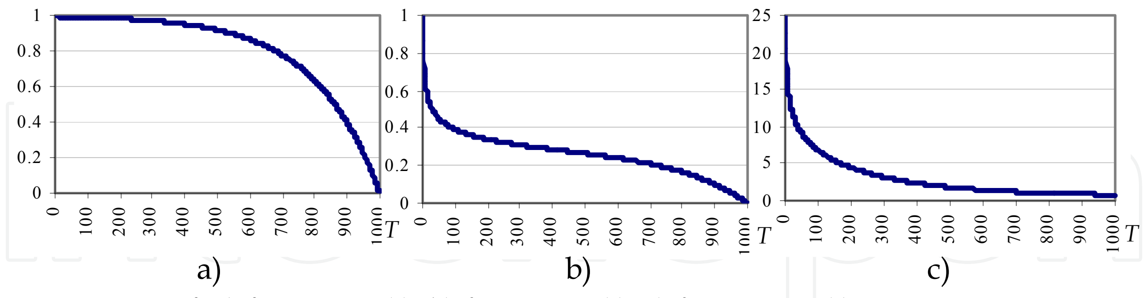


Fig.2. Diagrams of: a) function $\chi(t)$, b) function $\delta(t)$, c) function $\sigma(t)$.

5. Ability of SOM algorithm to order mesh nodes

The whole learning process can be divided into two stages: *ordering stage* and *refining stage* (Kohonen, 2001). During the ordering stage, the learning step and radius are large enough and all mesh nodes takes significant displacements. Therefore, even starting from random initial locations, the mesh nodes become ordered resembling the fixed mesh in the computational domain Q. During the refining stage, the learning step and radius slowly tend to zero and $r(T)$ correspondingly. It leads to a mesh approximating the physical domain in more and more details of a border and density distribution.

In Fig. 3 and Fig. 4, ability of the SOM algorithm to order neuron weights is demonstrated in 2D and 3D cases. During the ordering stage the algorithm tends to reproduce a regular structure of the fixed mesh starting from random initial data.

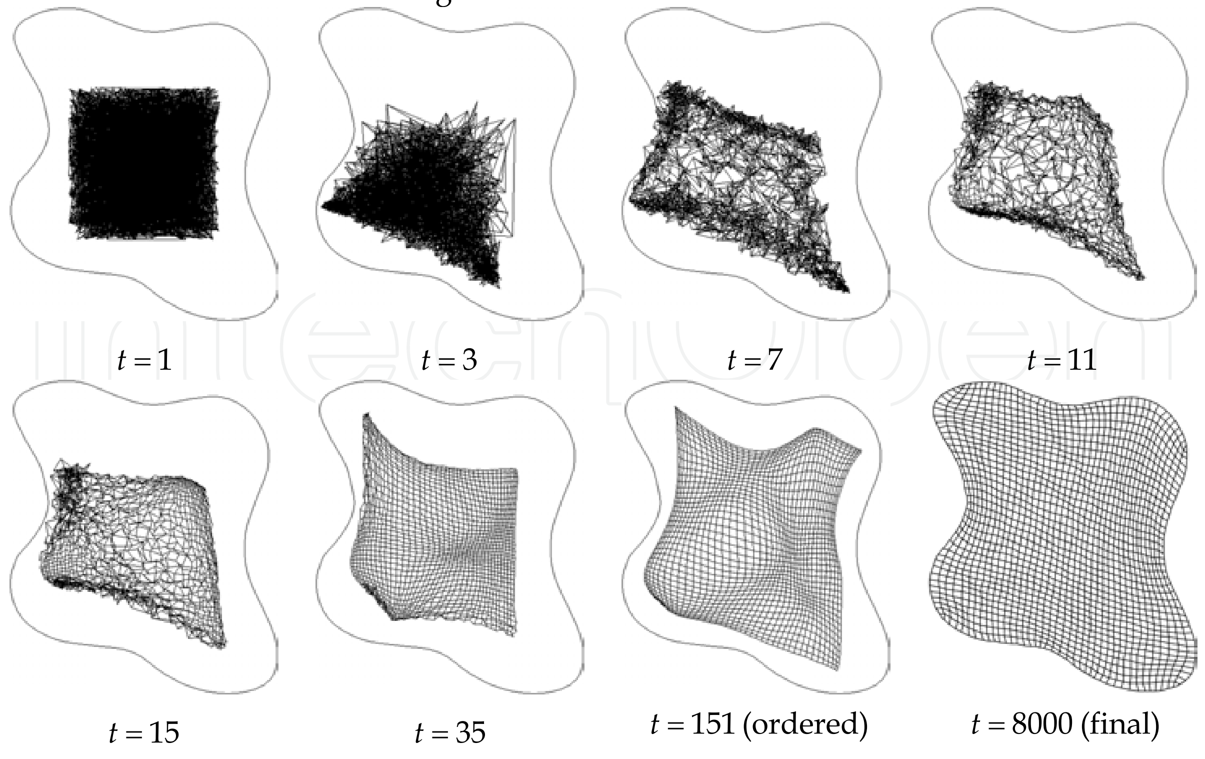
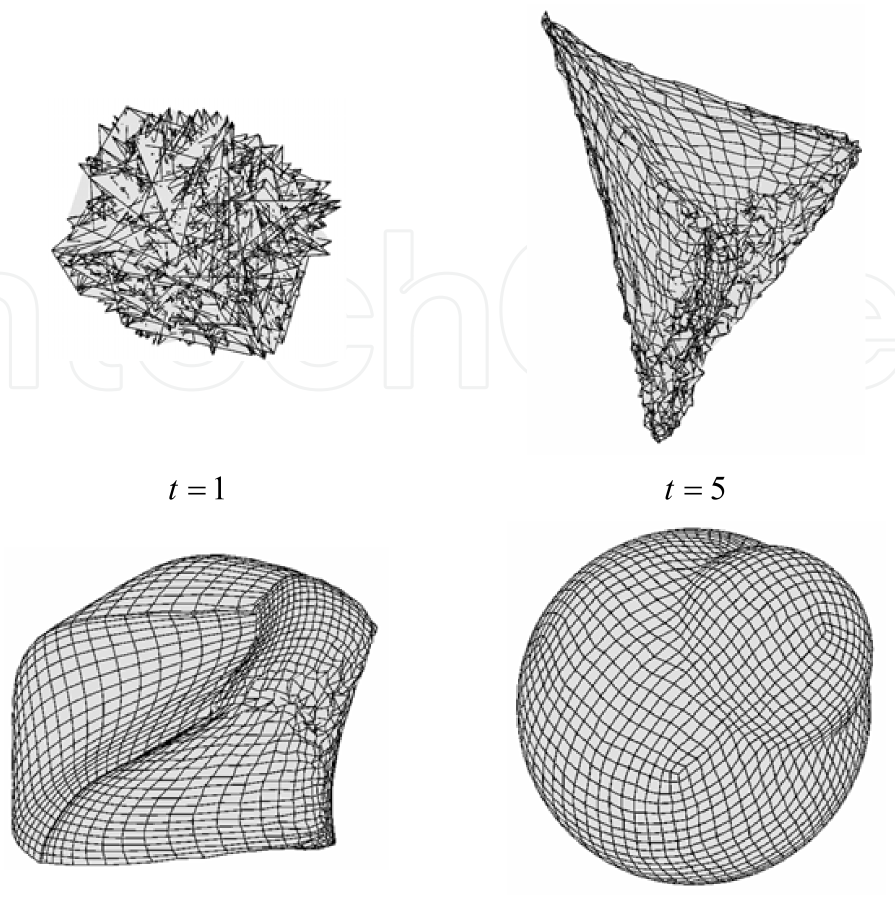


Fig. 3. The ordering stage of SOM learning and final mesh in 2D case.



$t = 20$ (ordered) $t = 60000$ (final)
Fig. 4. The ordering stage of SOM learning and final mesh in 3D case.

6. Theorem of correspondence

The SOM learning algorithm aims to find a mapping $m : R_G^n \to R_Q^k$ (more specifically $m : G \to Q_N$) by determining the set of weight vectors $x_1,...,x_N$. The mapping m has the following form:

$$m(y) = \arg\min_i d(y, x_i) . \tag{6}$$

At best, this mapping has to satisfy the following conditions (goals of SOM learning (Fritzke, 1997)).

(1) *Topology preservation*. If y_i and y_j are near each other in the input space, then neurons $e_{m(y_i)}$ and $e_{m(y_j)}$ are also nearby or $m(y_i) = m(y_j)$.

(2) *Equiwinning Percentage* (EWP). For each *i*-th neuron in the map *M*, there are the same number of vectors y_j in the training set *H* that are closer to the weight vector x_i than to any other weight vector. This also means that each neuron at the end of learning process has the same probability to become a winner for a randomly chosen input vector from *H*.

If these goals are satisfied, it is possible to proof that at the end of the learning process the resulting adaptive mesh reached desired approximation of mesh density function in a sense

of equidistribution principle. This theorem is called the Theorem of Correspondence, since it assigns the correspondence between goals of adaptive mesh construction in a traditional equidistribution sense (formulated in Section 2) and the goals of SOM learning algorithm.
The theorem of correspondence has been formulated and proven in order to show principal possibility to obtain adaptive meshes with given mesh density using SOM. This theorem states that if the EWP goal is reached, then an analogue of equidistibution principle is satisfied for Voronoi cells of the adaptive mesh.
The Voronoi cell V_i is the unbounded set of all point from G closer to x_i than to any other mesh node, i.e. $V_i = \{x \in G \mid d(x,x_i) < d(x,x_k), k = 1,...,N, k \neq i\}$ (Okabe et al., 2000). The whole G then can be represented by closure of the union of disjoint Voronoi cells.

Theorem of correspondence

Let the EWP condition be satisfied for the map of neurons M. Then the product of the square of Voronoi cell V_i and the value of probability density $p(x_i)$ can be estimated by $\frac{1}{N}$ for all i, i.e.:

$$|V_i| \cdot p(x_i) \approx \frac{1}{N}, \quad i = 1,...,N . \tag{7}$$

Proof

Let P_i be the number of elements in the sample $H = \{y_1,...,y_T\}$ which are closer to the node x_i than to any other node, i.e. $P_i = |\{y_j \in H \mid m(y_j) = i\}|$. If there are several closest nodes, the one from such nodes is to be chosen randomly.
According to the definition of integral, the square of V_i can be represented as:

$$|V_i| = \int_G \chi_{V_i}(x)dx , \tag{8}$$

where $\chi_A(x)$ is an indicator of a set A, i.e. $\chi_A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases}$. Let us calculate the square of V_i using the Monte Carlo methods (Mikhailov & Voitishek, 2006). After multiplication by the density $p(x)$ of sample H distribution, the integral (8) has the following form:

$$|V_i| = \int_G \chi_{V_i}(x)dx = \int_G \frac{\chi_{V_i}(x)}{p(x)} p(x)dx , \tag{9}$$

and can be considered as an expectation of the stochastic variable having the values $\frac{\chi_{V_i}(x)}{p(x)}$ and being defined over the domain G. Using the sample H, the expectation (9) can be estimated by the finite sum:

$$\int_G \frac{\chi_{V_i}(x)}{p(x)} p(x)dx \approx \frac{1}{T} \sum_{j=1}^{T} \frac{\chi_{V_i}(y_j)}{p(y_j)} . \tag{10}$$

Among all items of the sum (10), there are some which correspond to the elements y_j with the indicator value $\chi_{V_i}(y_j)=0$. It just means that x_i is not the closest to y_j. The number of nonzero items in (10) is equal to P_i. After simplification, the sum (10) has the following view:

$$\frac{1}{T}\sum_{j=1}^{T}\frac{\chi_{V_i}(y_j)}{p(y_j)}=\frac{1}{T}\sum_{\substack{\alpha=1\\ y_\alpha\in H\cap V_i}}^{P_i}\frac{1}{p(y_\alpha)}. \tag{11}$$

Further, the values $p(y_\alpha)$ can be approximated by the values $p(x_i)$, since x_i is close to a center of gravity of V_i for the majority of Voronoi cells. It is clear that with $N\to\infty$ the error of such an approximation tends to zero. Finally, we have the following estimation:

$$\frac{1}{T}\sum_{\substack{\alpha=1\\ y_\alpha\in H\cap V_i}}^{P_i}\frac{1}{p(y_\alpha)}\approx\frac{1}{T}\sum_{\substack{\alpha=1\\ y_\alpha\in H\cap V_i}}^{P_i}\frac{1}{p(x_i)}=\frac{P_i}{Tp(x_i)}. \tag{12}$$

Since the EWP condition is satisfied, for each i-th neuron in the map M, there are the same number of vectors y_j in the training set H which are closer to the weight vector x_i than to any other weight vector. From this condition it follows that the fraction $\frac{P_i}{T}\to\frac{1}{N}$ with $T\to\infty$. Proceeding to limit in (12), we can get the following:

$$|V_i|\approx\frac{P_i}{Tp(x_i)}\to\frac{1}{Np(x_i)}. \tag{13}$$

After multiplication by $p(x_i)$, we obtain the estimation:

$$|V_i|\,p(x_i)\approx\frac{1}{N}. \tag{14}$$

The estimation (12) is correct for all $i=1,\ldots,N$.♦

Now, after this theorem is proved, the traditional goals of adaptive mesh construction and ones of the SOM learning can be considered as equivalent. Unfortunately, there is no proof that the goals of SOM learning can always be reached. Moreover, if we apply the basic SOM algorithm, a number of notorious problems often occur leading to the failures of these goals. First, it is impossible to obtain an accurate approximation of border of a physical domain, as it can be seen from the example in Fig. 4 (c) and (d), because boundary nodes never reach the border and they are influenced by the border effect. Second, sometimes, boundary nodes can propagate into the interior of the domain, especially if the probability distribution $p(x)$ is non uniform. That is the result of bad topology preservation as Fig. 6 (a) shows. Finally, the mesh may contain self-crossings (Fig. 6 (c)) that makes it entirely unusable for numerical simulations. In the next Sections, all these problems are considered in details and our solution to them in the form of the composite algorithm is introduced.

7. Border effect evaluation after applying the basic SOM model

The border effect is closely connected with failure of the EWP condition. Thereby, in this section, the EWP condition is evaluated. According to the definition, if the EWP condition is satisfied, then each neuron has the same probability to become a winner. It is convenient to measure it statistically. In other words, the values of function $m(y)$ can be recorded for all vectors of the training set H.

Let a mesh be constructed by a basic SOM algorithm. For evaluation of the EWP condition, a winning statistics has been recorded for the constructed mesh, i.e. how many times each neuron became a winner. It has been found that equal winning percentage directly depends on the final learning radius $r(T)$. If a mesh has been constructed with $r(T)$ being such that the learning neighborhood $B_{r(T)}(q_m)$ contains only the nearest neighbors of q_m, then the winning percentage is almost the same for all neurons. But if $r(T)$ is large, then the adaptive mesh collapses inside the physical domain and, then, boundary nodes become a winner more frequently. In Fig.5, the winning statistics for two different $r(T)$ is shown at the mesh cut. On the other hand, the radius $r(T)$ essentially influences the mesh smoothness in such a way that small radius leads to unsmooth adaptive meshes, and this usually causes the decreasing of the accuracy of numerical simulations on these meshes. The larger the radius, the smoother the adaptive mesh as it is shown in Fig. 5.

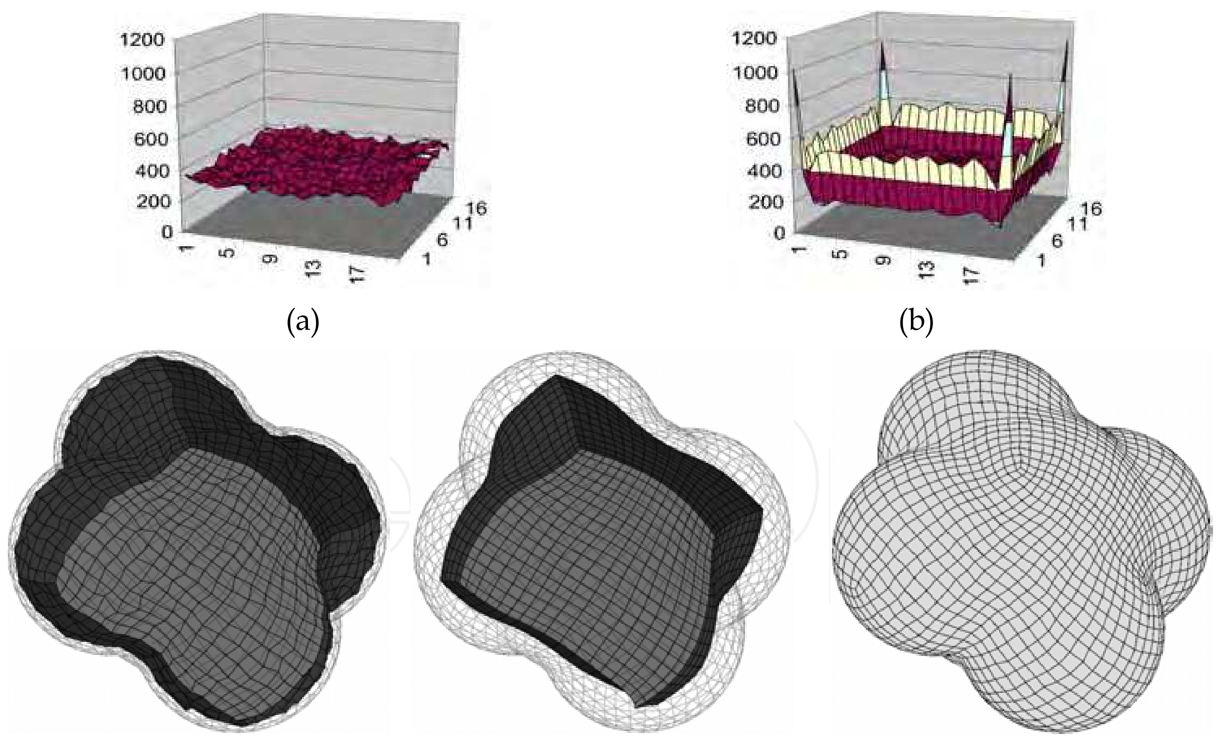


Fig. 5. The winning statistics for the mesh constructed by the basic SOM, mesh size is $20 \times 20 \times 20$. (a) $r(T) = 1$, (b) $r(T) = 10$. The corresponding meshes: (c) $r(T) = 1$, (d) $r(T) = 10$. The desired mesh (e) is constructed by the composite algorithm proposed in Section 9.

In Section 9, the smoothing algorithm is proposed, which is based on SOM learning with large learning radius and with a technique handling the border effect.
Let us study in details the origins of border effect in the basic SOM model. Being able to measure the border effect, we can handle it. To this end, the iteration number *t* is assumed to be fixed. Let us consider an interior neuron q_i for which the distance to border of the computational domain is greater than the learning radius *r*. If the mesh Q_N is rectangular uniform, all neurons q_j from $B_r(q_i)$ as well as all strengths of lateral connections $\eta_{q_j}(q_i)$ are symmetrically located around q_i. Therefore, as it follows from the EWP condition, the neuron q_i has the same probability to be influenced by any other neuron q_j. In the physical domain, it means that the node x_i has the same probability to move symmetrically in all directions being guided by neurons from $B_r(q_i)$. Since *s* is close to zero, then it is assumed that the mutual influence between neurons q_i and $q_j \notin B_r(q_i)$ is negligibly small.
If the distance from q_i to the border of the computational domain is less than *r*, then there are not enough neurons in $B_r(q_i)$ for symmetry. In this case, most of the neurons in $B_r(q_i)$ make the neuron q_i move mainly to the center of the physical domain. To balance the asymmetry, the neuron q_i needs to move aside the border of *G*.
To evaluate the asymmetry, let us consider the following characteristic of the neuron q_i:

$$\alpha_i = \sum_{j=1}^{N} \eta_{q_j}(q_i)\,. \tag{15}$$

For each node, this characteristic is a sum of lateral connection strengths with all other nodes. If q_i is near the border of *Q*, then there is not enough terms in the sum (15) corresponding to $B_r(q_i)$. Therefore, α_i is decreasing near the border of *Q*. It can be clearly seen from the diagram in Fig. 6. All the nodes located at a distance greater than *r* from the border have the same value of this characteristic.
Obviously, to handle the border effect, it is necessary to balance the asymmetry of lateral connections. It is still an opened question, how the diagrams in Fig. 5 and Fig. 6 are correlated with each other. In the future, this question is going to be answered in order to improve the EWP condition fulfillment.

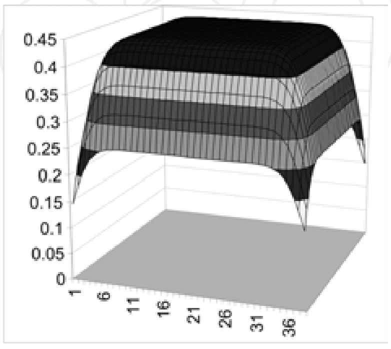


Fig. 6. Characteristic of lateral connections symmetry - values of α_i for the mesh cut, mesh size is $40 \times 40 \times 40$.

It has to be noted that there are some cases when the border effect does not appear. If a map of neurons is closed in such a way that it is not possible to pick out boundary neurons, then this map does not suffer from border effect. Examples of such a maps are: map of neurons forming a ring, map of boundary neurons belonging to a rectangular uniform 2D and 3D grid, a map in the form of torus, and so on. In Fig 5(e), a 3D surface mesh is constructed without border effect.
As a conclusion of this Section, let us point out that even if the EWP condition is fulfilled, the nodes of mesh, obtained by the basic SOM model, still do not reach the border. The explanation of this can be given as follows. Trying to meet the EWP condition, mesh nodes are getting close to the center of gravity of the corresponding Voronoi cell. Since a Voronoi cell is convex, then it is not possible for mesh nodes to appear on the border, at least for convex physical domains. It follows from this that the basic SOM model can be applied *only for interior nodes* when constructing an adaptive mesh. Therefore, in Section 9, the composite algorithm is proposed which is based on the alternative application of the basic SOM models separately to a border and to interior of the domain.

8. Topology preservation after applying the basic SOM model

According to the definition in Section 6, the topology preservation condition is when input vectors that are near to each other in the input space are mapped into nearby or the same neuron locations. As a measure of topology preservation at level γ the quantity $\tau_\gamma \in \mathbb{N}$ can be used, which is defined as follows. Let us consider (together with the winning function $m(y)$) the function of second winner: $m'(y) = \arg \min\limits_{\substack{i=1,\dots,N \\ i \neq m(y)}} d(y, x_i)$. Given $N > 2$ and $\gamma \in \mathbb{R}$, the function $\mu_\gamma(y)$ is defined, which answers the question whether the second winner is in the neighborhood of the first one in *Q* or not:

$$\mu_\gamma(y) = \begin{cases} 0, & q_{m'(y)} \in B_\gamma(q_{m(y)}) \\ 1, & \textit{иначе} \end{cases}$$

Thus, the measure of topology preservation at level γ is a quantity which is equal to the number of training vectors in H, for which the second winner is outside the neighborhood of the first one: $\tau_\gamma = |\{y \in H \mid \mu_\gamma(y) = 1\}|$. It is appropriate to take the value γ in such a way that for each node the neighborhood $B_\gamma(q_i)$ contains the nearest neighbors of q_i. A nonzero value of τ_γ indicates the failure of topology preservation, and non zero values of $\mu_\gamma(y)$ can help to find locations of this failures.
In 3D space, there are a number of typical cases of topology preservation failures when applying the basic SOM model. These cases have equivalents in 2D space too.
1. If the mesh density function is non uniform, then boundary nodes can propagate inside the physical domain being attracted by the high density of input vectors. Usually, such a boundary nodes never leave the attractor and it leads to undesirable bends of the mesh as it is shown in Fig. 7 (a). At this bands, the values of $\mu_\gamma(y)$ are nonzero.

2. If initial locations of mesh nodes are random, there is a probability to obtain a mesh with self-crossings, as it is shown in Fig. 7 (c). But the larger the learning radius $r(1)$, the less the probability of self-crossings. Just because of this at the beginning of the learning process the radius $r(1)$ should cover all the nodes. To further decrease this probability, one can use an initial mesh without self-crossings, for example, rectangular uniform, located somewhere inside the physical domain.

3. When the configuration of physical domain is highly complex, the topology preservation failure can be caused by the inappropriate mesh layout, since its formation is based on self organization. To handle this, the coloring technique can be used, which is described in (Nechaeva, 2007).

All the above cases can be overcome by the composite algorithm.

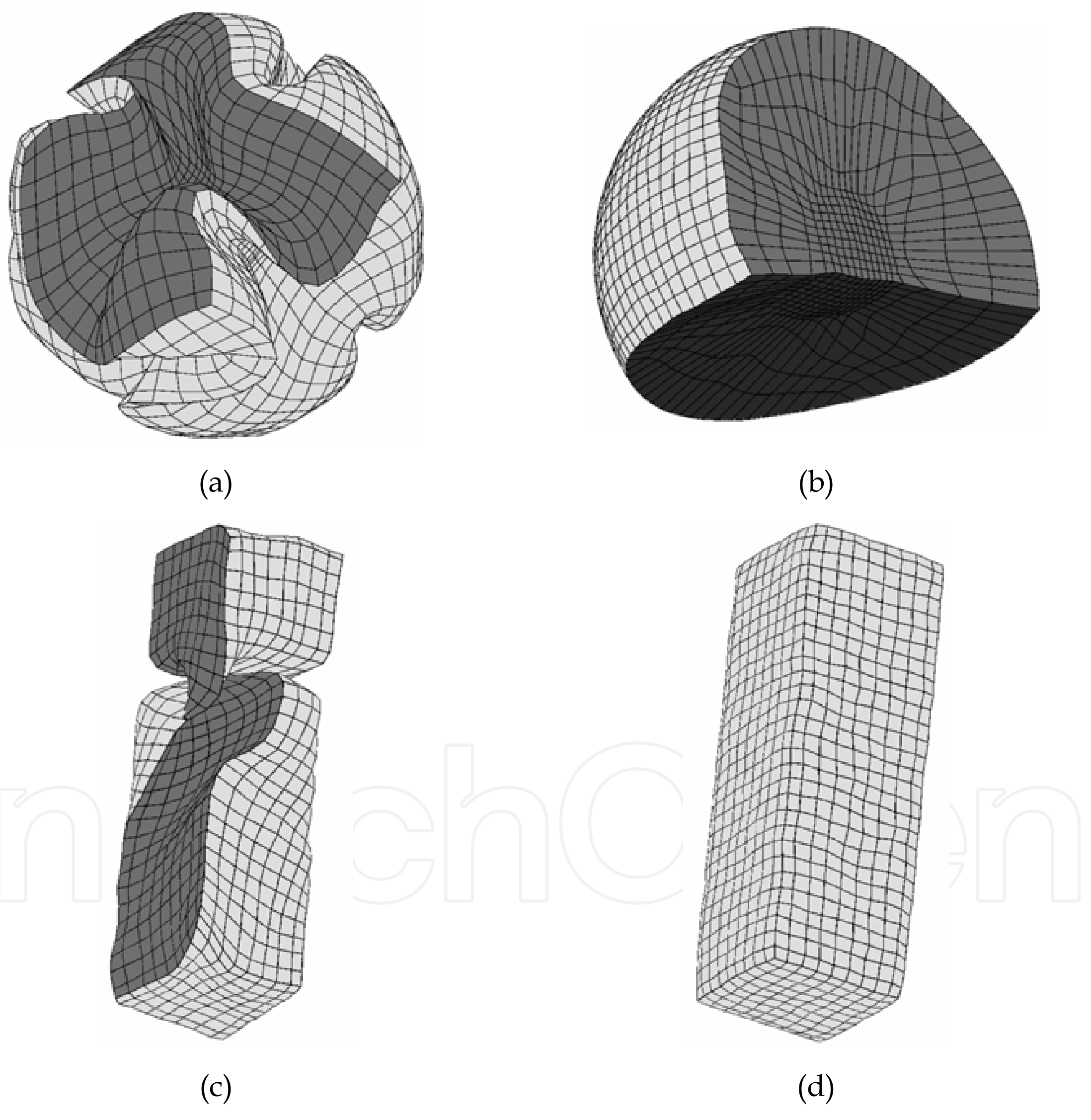


Fig. 7. Topology preservation failures when using the basic SOM model and the desired adaptive meshes constructed by the composite algorithm proposed in Section 9; (a) mesh nodes propagating inside the domain when the mesh density function is non uniform; (c) mesh self crossings.

9. Composite algorithm for adaptive mesh construction

The idea of the composite algorithm (Nechaeva, 2006) is to combine a number of SOM models interacting between each other in a special way and self-organizing over their own set of input data. For example, all neurons in the 3D map M can be divided into two subsets: M_{int} is the set of neurons which correspond to interior nodes and form a 3D volume mesh, and M_b is a set of neurons which correspond to boundary ones and form a 3D surface mesh. In addition, the physical domain G can be divided into a border and interior. Let H_{int} be a training set consisting of vectors only from the interior of G, and H_b is a training set consisting of vectors from the border of G. Taking into account the SOM learning algorithm Alg, we have two SOM models: SOM$_{int}$ = < M_{int}, H_{int}, Alg > and SOM$_b$ = < M_b, H_b, Alg >. This kind of division seems to be the most convenient for the majority of physical domains which have been studied.
The composite algorithm is based on special alternation of training for each SOM model. The main requirement for the composite algorithm is to provide the consistency between boundary and interior mesh nodes.
Each alternation stage of the composite algorithm consists in training of all SOM models during a given number of iterations, is referred to as a macroiteration, and is denoted by s. For each map M_k, $k = int$, b, there is a private counter of iterations t^k, and the maximum number of iterations T_k is given in such a way that T_k is proportional to $|M_k|$, i.e. to the number of neurons in the map M_k. Let $\varphi_k(s)$ be the number of iterations at the macroiteration s during which the learning procedure is to be applied to the k-th SOM model.

Composite algorithm

(0) Set arbitrary initial weights of all neurons $x_i(0)$, $i = 1,...,N$.

(1) At the first macroiteration ($s = 1$), apply the procedure Alg to the general map M with input vectors taken from H and $t_{st}(1) = 1$, $t_{fin}(1) = T_0$, where T_0 is a given number of iterations.

(2) Repeat the following operations at each macroiteration $s > 1$ until the maximum number of iteration is reached:

  (a) Training of SOM$_b$. Apply the procedure Alg to the map M_b with input vectors taken from H_b and $t^b_{st}(s) = t^b_{fin}(s-1)+1$, $t^b_{fin}(s) = t^b_{st}(s) + \varphi_b(s) - 1$.

  (b) Training of SOM$_{int}$. Apply the procedure Alg to the map M_{int}, but with winner selection from the whole map M. Input vectors are taken from H_{int}. If the winner e_m is from M_b, then replace the input vector $y_{t^{\text{int}}}$ by the weight vector x_m; $t^{\text{int}}_{st}(s) = t^{\text{int}}_{fin}(s-1)+1$, $t^{\text{int}}_{fin}(s) = t^{\text{int}}_{st}(s) + \varphi_b(s) - 1$.

The step (1) of the composite algorithm is an ordering stage. Application of Alg to all mesh nodes makes the mesh become ordered and take roughly the form of G. The number of iterations T_0 depends on the physical domain configuration. Typically, T_0 is varying from $0.005T$ to $0.01T$. After this step, boundary nodes are located near their appropriate border positions.
The step (2) is a refining stage. Both M_{int} and M_b consistently fit more and more fine details of the interior and border of G. At this stage, at substep (a), boundary nodes have a leading

role. It has been noted that boundary nodes more easily approximate the border than interior nodes approximate the interior of *G*, because in most cases the map M_b is closed (does not have borders). This is true for 3D space as well as 2D space. Therefore, boundary nodes can move within the domain even independently of the interior nodes. At substep (b), interior nodes always follow the boundary ones by means of special winner selection. Since the winner is selected among all the neurons, time to time the winning neuron is a boundary one. In this case, an input vector is replacing by a winner weight vector and all interior nodes move towards the boundary winner. This technique, first, does not let boundary nodes and their interior neighbors propagate inside the physical domain even if there is a subdomain of high density of input vectors; second, keeps a topological connection between interior nodes and their nearest boundary neighbors; and third, excludes mesh self crossings, if there is no self crossings among boundary nodes (that is easy to handle).
We found that the form of functions $\varphi_k(s)$ for defining the number of iterations at each macroiteration is not crucial for the composite algorithm. The resulting mesh is quite the same for different functions $\varphi_k(s)$. For example, these functions can be assigned as $\varphi_k(s) = T_k / S$ for $s > 1$ and $\varphi_k(1) = T_0$, where *S* is the maximum number of macroiterations. But sometimes, it is possible to accelerate the mesh construction by appropriate selection of $\varphi_k(s)$. For example, good results could be obtained if $\varphi_b(s)$ increases and $\varphi_{\text{int}}(s)$ decreases (Fig. 8). The acceleration is achieved because the boundary nodes quickly take correct distribution along the border of *G* and then get frozen giving the interior nodes an advantage until the termination of the composite algorithm. The functions $\varphi_k(s)$ also can be chosen depending on the physical domain configuration.

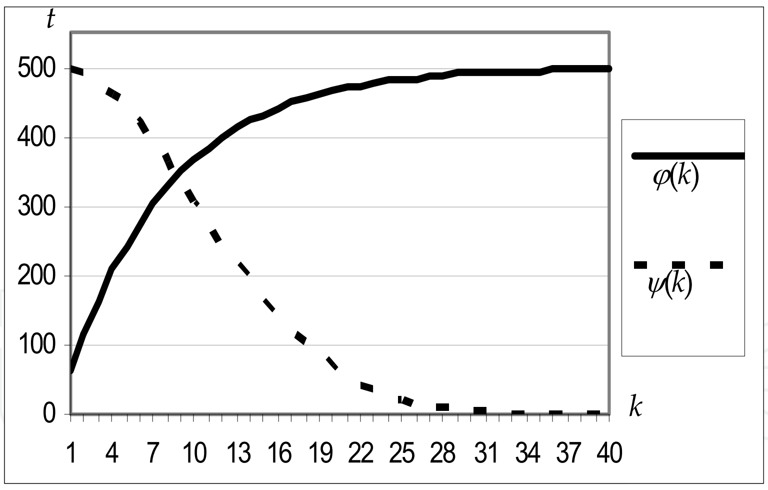


Fig. 8. Diagrams of the functions $\varphi(k)$ and $\psi(k)$.

In Fig. 9, some examples of adaptive meshes constructed by the composite algorithm in 2D and 3D cases are shown.

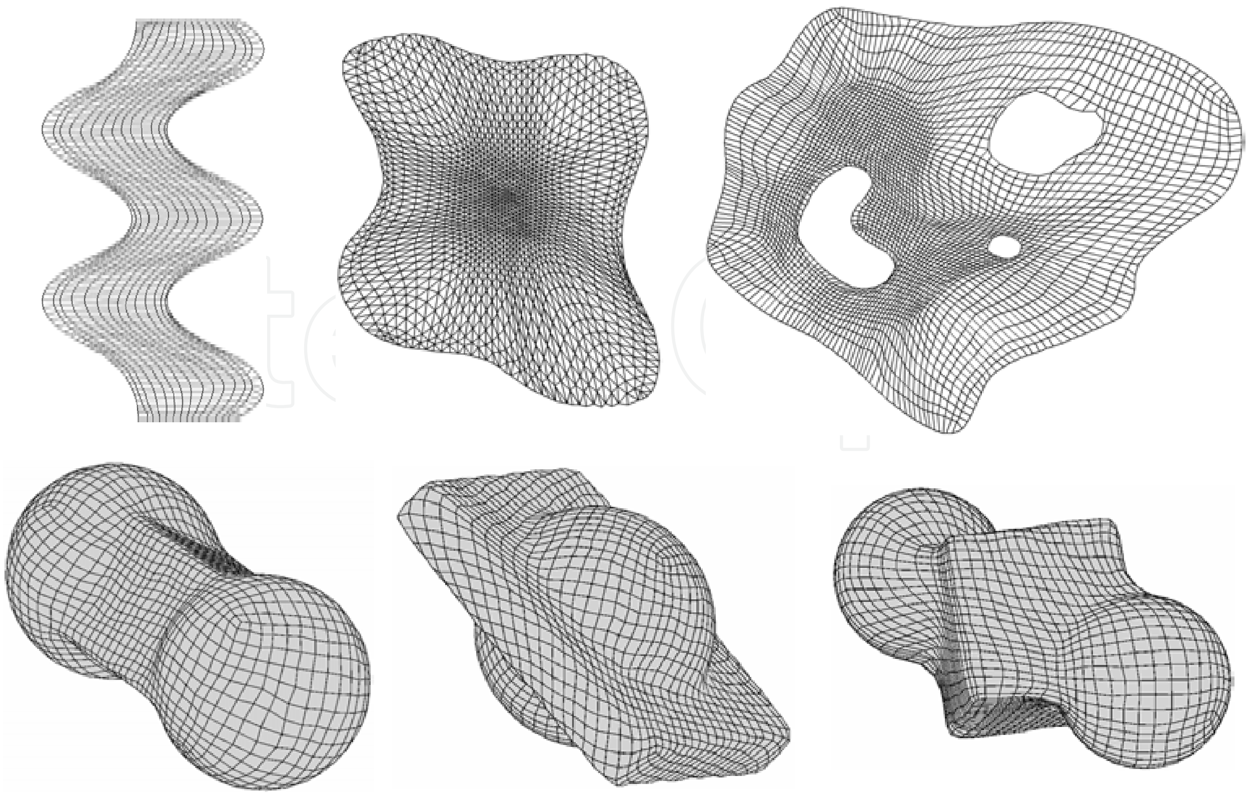


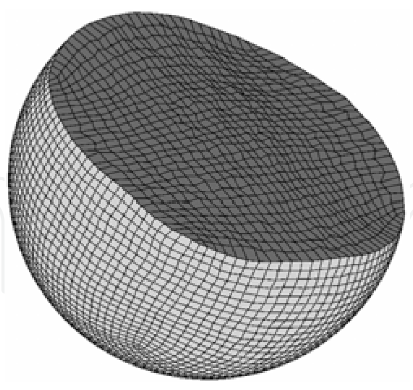
Fig. 9. Examples of meshes constructed by the composite algorithm.

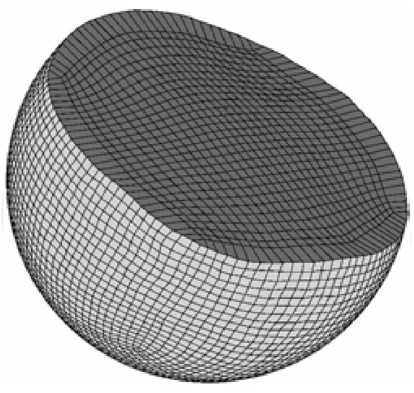
10. Topology preservation after applying the basic SOM model

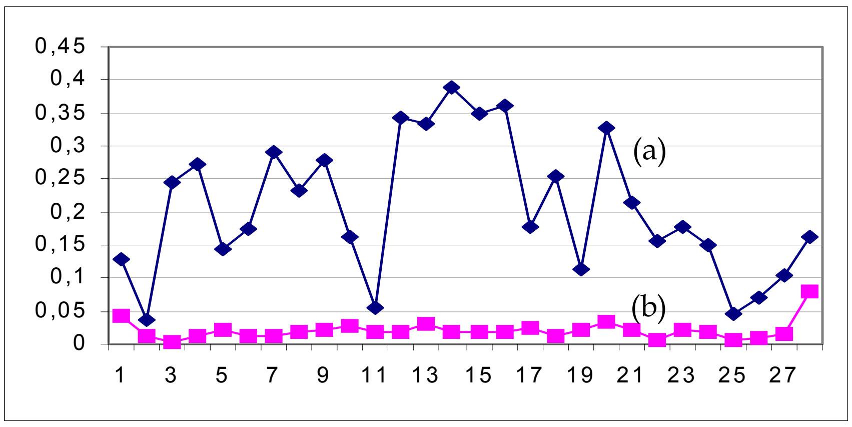
The composite algorithm overcomes the border effect for small values of learning radius. However, a small radius leads to unsmooth adaptive meshes, and this usually causes the decreasing of the accuracy of numerical simulations on these meshes. The aim of this Section is to propose the technique that allows us to use large learning radius for obtaining the smooth enough adaptive meshes but without the border effect.

Let us consider the case when Q_N is a rectangular uniform mesh. It means that each node of Q_N has four neighbors, and distances between neighboring nodes are equal to d_Q. To measure the smoothness of a quadrilateral adaptive mesh, let us consider a notion of a mesh line which is a set of nodes being the image of a line of the fixed uniform mesh Q_N. Smoothness of a mesh line can be measured by the sine values of angles between two segments, connecting the neighboring nodes in the mesh line, in a sense that the less the quantity of sign inversions and the amplitude of these values, the smoother the line.

Our experiments showed that the mesh smoothness depends on the relation between learning step and radius. Given a fixed learning step, the larger the radius, the smoother the mesh. It can be clearly seen from the example below. In Fig. 10(a), the mesh constructed by the composite algorithm with artificially small final radius $r(T)$ is shown. This mesh is unsmooth even visually. For comparison, in Fig 10(b), the mesh is shown which has been constructed with the final radius 2 times greater. Boundary nodes did not move in this experiment, but they were allowed to become a winner. As it is shown in Fig. 10(c), the smoothness of the last mesh is much better because the sine values are comparatively small and have less sign inversions. But the resulting mesh is inappropriate for numerical simulations because of bad approximation of the border of the physical domain.


(a)


(b)



(c)

Fig. 10. Measure of mesh smoothness; (a) the mesh obtained by the composite algorithm with the artificially small final radius *r*(*T*) = 0.5; (b) the mesh with final radius *r*(*T*) = 6; (c) diagrams of sine values for the meshes (a) and (b) respectively.

This experiment is a bright demonstration of the border effect in the SOM, which appears when the learning radius is large. The necessary condition for obtaining a smooth enough adaptive mesh is a large learning radius. Therefore, the main problem while smoothing is to handle the border effect.

The general scheme of adaptive mesh construction with employment of the smoothing technique proposed below is as follows. The composite algorithm constructs adaptive mesh with the learning radius being suitable for proper mesh nodes distribution and for fulfillment of the EWP condition. Boundary nodes of this mesh are distributed along the border of *G*. Starting from this mesh, a SOM-like procedure is applied during the a fixed number of iterations with the constant learning rate, i.e. $r(t) = r$, $\delta(t) = \delta$, $\eta_{q_m}(t,q_i) = \eta_{q_m}(q_i)$, where the learning radius r is comparatively large and the learning step δ is small. This procedure adjusts locations only of the interior mesh nodes and can be regarded as the last stage of the composite algorithm.

After the termination of the composite algorithm, all mesh nodes are distributed over the physical domain according to the given mesh density function. We assume here that the EWP condition is satisfied for this mesh. Therefore, the probability to be a winner is equal to $1/N$.

As it has been shown in Section 7, to eliminate the border effect, it is necessary to balance the asymmetry of lateral connections and to achieve the same value of α_i defined in (15) for all neurons. We propose the technique that allows us to use the boundary nodes as representatives of missing neurons near the border of Q.

Let us imagine that for each boundary neuron, there are K virtual neurons located outside the computational domain. These virtual neurons do not exist in the algorithm but they will help to understand the underlying idea of the proposed technique. The exact locations of virtual neurons are unknown. The only available information is that the distance between k-th virtual neuron and the corresponding boundary neuron q_m is equal to kd_Q, $k = 1,\dots,K$, where $K = \lceil r / d_Q \rceil$ and $\lceil a \rceil = \arg\min_{n\in\mathbb{Z}}(a < n)$ is the smallest integer no less than a.

To involve virtual neurons into learning process, the following questions are to be resolved: (1) in what conditions a virtual neuron becomes a winner? (2) what are the strengths of lateral connections between neurons q_i, $i = 1,\dots,N$, and virtual ones? (3) what are the directions and magnitudes of mesh nodes displacements in the physical domain when the winner is a virtual neuron?

Answer to the question (1)

In the case of presence of virtual neurons, winner selection can not be based only on the closeness to the random point, because there are no points outside the physical domain. Therefore, at each iteration, first of all, it is necessary to decide from which kind of neurons the winner is to be selected. Since the EWP condition is satisfied, virtual neurons have the same probability to become a winner as all the other neurons. The probability of virtual neurons to become a winner is equal to $N_b K / (N_b K + N_{\text{int}})$, and hence, an interior neuron can be a winner with the probability $1 - N_b K / (N_b K + N_{\text{int}})$. To select the winner among virtual neurons, an input vector y is selected from the boundary training set H_b, a boundary node which is closest to y is determined, and then the k-th virtual neuron randomly selected (with uniform probability) from the set of virtual neurons, which correspond to the determined boundary node, is assigned to be a winner.

Answer to the question (2)

To define lateral connections strengths between virtual and ordinary neurons, it is necessary to know distances between them in the computational domain. The distance between k-th virtual neuron and the neuron q_i is assumed to be equal to $d(q_m, q_i) + kd_Q$, where q_m is the boundary neuron corresponding to the virtual neuron. This distance is approximate, because the exact location of this virtual neuron in the computational domain is unknown. The lateral connection between k-th virtual neuron related to the boundary neuron q_m and the neuron q_i is taken as follows.

$$\eta_{q_m,k}(q_i) = s^{\left(\frac{d(q_m,q_i)+kd_Q}{r}\right)^2} .$$

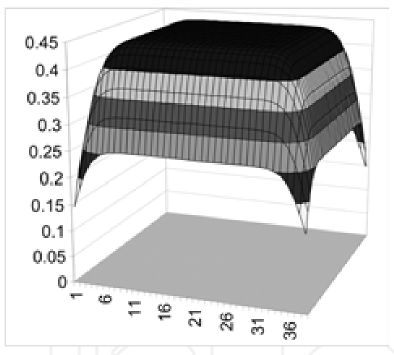
Answer to the question (3)

To specify the directions and magnitudes of the mesh nodes displacements in the physical domain when the winner is a virtual neuron, it is proposed to use the random point y on the border of G, which has been generated for winner selection from the virtual neurons. Let us remind that only interior mesh nodes can move during the smoothing stage. For each interior node x_i, the direction of its displacement is given by the vector $y - x_i(t)$, i.e. the node x_i moves toward the point y located on the border of G. The magnitude of the displacement is equal to $\delta\eta_{q_m,k}(q_i)\cdot v_i(t)\cdot d(y,x_i(t))$, where $v_i(t) = 1 + kd_Q / d(q_m,q_i)$ and q_m is the boundary neuron which corresponds to the virtual winner. This value has been found on the ground of assumption that the ratio between $d(q_m,q_i)$ and $d(y,x_i(t))$ is equal to the ratio between $d(q_m,q_i) + kd_Q$ and $v_i(t)\cdot d(y,x_i(t))$. Since the rule is applied only to interior nodes, then $d(q_m,q_i) \neq 0$.

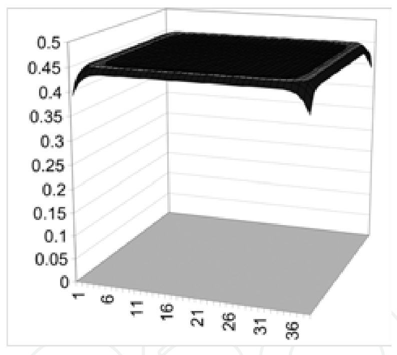
Taking into account virtual neurons, the characteristic (15) changes and is equal to:

$$\bar{\alpha}_i = \sum_{j=1}^{N}\eta_{q_j}(q_i) + \sum_{k=1}^{K}\sum_{m=1}^{N_b}\eta_{q_m,k}(q_i), \tag{16}$$

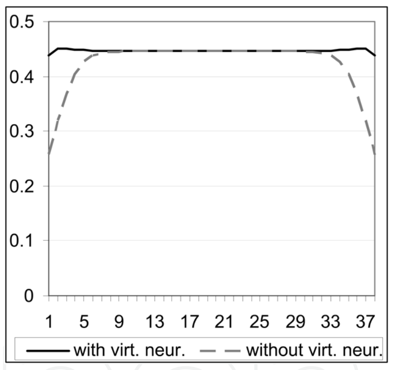
where $m = 1,\ldots,N_b$ is an index of a boundary node. In Fig. 11(c), the diagrams of α_i and $\bar{\alpha}_i$ for a mesh cut are shown. It can be seen that $\bar{\alpha}_i$ is almost constant for all neurons in this cut. Therefore, the proposed technique balances the asymmetry of lateral connection near the border.



(a)



(b)



(c)

Fig. 11. Characteristic of lateral connections symmetry; (a) values of α_i without virtual neurons; (b) values of $\bar{\alpha}_i$ with virtual neurons; (c) the cut of diagrams (a) and (b) where dashed grey line is correspond to (a) and solid black line is correspond to (b).

The algorithm of the smoothing stage is a SOM-like procedure with constant learning parameters. The learning radius r is chosen to be comparatively large, but it is bounded by the curvature of the border of G. The learning step δ is to be small, because fine tuning is needed for smoothing and it does not impair essentially the mesh density approximation.

Besides, when a virtual neuron is a winner, an imaginary random point is outside the physical domain, which can lead to the mesh nodes crossing the border of G. To exclude this situation, the learning step should satisfy the following condition: $\delta(1 + kd_Q / d(q_m, q_i)) < 1$ for any boundary neuron q_m and interior neuron q_i. From this the following condition can be obtained.

$$\delta < \min_{m,i,k}\left(\frac{d(q_m, q_i)}{d(q_m, q_i) + kd_Q}\right) = \frac{d_Q}{d_Q + Kd_Q} = \frac{1}{K+1} \,. \tag{17}$$

Smoothing Stage Algorithm

Repeat the following operations during the fixed number of iterations.

1. Generate a random number α from [0,1] with uniform probability distribution.
2. If $\alpha \in [0,\ N_{\text{int}} / (N_b K + N_{\text{int}})]$, then perform a SOM-like procedure which consists in the following:
   a) Take randomly an input vector y from G using the probability distribution $p(x)$.
   b) Select a winning node $x_m(t)$ among all the neurons. If $x_m(t)$ is a boundary neuron, then replace an input vector with the weights of the winning neuron: $y := x_m(t)$.
   c) Adjust the weights only of the interior neurons according to the rule:

$$x_i(t+1) = x_i(t) + \delta\eta_{q_m}(q_i)(y - x_i(t))$$

3. If $\alpha \in (N_{\text{int}} / (N_b K + N_{\text{int}}),\ 1]$, then perform the following operations:
   a) Take randomly an input vector y from the border of G with the probability distribution $p(x)|_{\partial G}$.
   b) Select the boundary node $x_m(t)$ which is closest to the point y.
   c) Choose randomly the number k from $\{1,...,K\}$.
   d) Adjust the weights only of the interior neurons according to the rule:

$$x_i(t+1) = x_i(t) + \delta\eta_{q_m,k}(q_i)\big(1 + kd_Q / d(q_m, q_i)\big)(x_m(t) - x_i(t)) \,.$$

It has to be again pointed out that virtual neurons do not exist, and thus, there is no need to change the structure of the fixed mesh when counteracting the border effect. Additionally, our efforts have been directed towards the making of the learning rule as simple as possible because of the following reasons: (1) to safe an inherent parallelism of the SOM algorithm which consists in that all neurons are processed according to the same rule independently of each other; (2) to avoid problems when constructing the mesh on a complex multiply-connected domain, i.e. the ones with a single or multiple holes, since the border effect is to be controlled at each of the borders.

10. Quality of resulting adaptive meshes

There are generally accepted quality criteria for quadrilateral 2D and 3D meshes such as the criteria of cell convexity and oblongness, the criterion of mesh lines orthogonality (Prokopov, 1989). In Table 1, possible and admissible values of these criteria are shown. Also, Table 1 contains the values of criteria for the constructed adaptive mesh, shown in Fig. 9 (b). The values are in the admissible range. Negative values of convexity and orthogonality criteria indicate that there are some non convex cells in the mesh.

Quality criterion	Values Possible/Admissible/Best	Values for Fig. 8(b) Average/Min
Cell convexity	$(-\infty;1]$ / $[0;1]$ / 1	before smoothing 0.825896 / -0.072630 after smoothing 0.927001 / 0.112040
Mesh planes orthogonality (min value of the sin of cell angles)	$[-1;1]$ / $[0;1]$ / 1	before smoothing 0.810640 / -0.034921 after smoothing 0.871268 / 0.158070
Cell oblongness (ratio between max and min edges of a cell)	$(0;1]$ / depending on a problem / 1	before smoothing 0.532211 / 0.000375 after smoothing 0.584676 / 0.098801

Table 1. Mesh quality evaluation.

11. Conclusion

The main result of this investigation is that we proposed an efficient method of adaptive regular mesh construction in 2D and 3D space based on Self Organizing Maps, which does not require solving complicated partial differential equations in order to achieve an acceptable quality of adaptive meshes. The principal possibility of construction adaptive meshes using the SOM models has been proved in the theorem of correspondence.
We believe that the proposed approach to efficient and automatic adaptive mesh construction will contribute to the theory and algorithms of mesh methods. In the future, the neural network approach will be extended to construction of moving structured adaptive meshes based on SOM-like models and unstructured adaptive meshes (Bohn, 1997) based on other self organizing models like Growing Neural Gas and Growing Cell Structures (Fritzke, 1997). The approach seems to be useful especially for building real life geometrical models from point clouds measured by lazer scanners, tomography devices, echo sounding, etc.

12. References

Bern, M. & Plassmann, P. (1999) Mesh Generation, *Handbook of Computational Geometry,* J.-R. Sack and J. Urrutia eds., Chapter 6, Elsevier Science, 1999.

Bessmeltsev, M. (2008) Geometry processing and visualization in scientific applications using GeomBox package, *Electronic publication*, http://aitricks.com/geombox/about.htm, SB RAS, Novosibirsk, Russia, 2009

Bohn, Ch.-A. (1997) Finite Element Mesh Generation using Growing Cell Structures Networks, *Neural Networks in Engineering Systems*, Turku, Finland, 1997.

Fritzke, B. (1997) Some competitive learning methods, *Technical report*, Systems Biophysics, Inst. for Neural Comp., Ruhr-Universität Bochum, April 1997

Khakimzyanov, G.S.; Shokin, Yu.I.; Barakhnin, V.B. & Shokina, N.Yu. *Numerical Modelling of Fluid Flows with Surface Waves,* SB RAS, Novosibirsk, 2001, 394 p.

Kohonen, T. (2001) *Self-organizing Maps,* Springer Series in Information Sciences, V.30, Springer, Berlin, Heidelberg, New York, 2001, 501 p.

Lebedev, A.S.; Liseikin, V.D. & Khakimzyanov, G.S. (2002) Development of methods for generating adaptive grids, *Vychislitelnye tehnologii*, Vol. 7, No. 3, 2002, pp. 29-43

Liseikin, V.D. (1999) *Grid Generation Methods,* Springer-Verlag, Berlin, Heidelberg, New York, 1999, 362 p.

Mikhailov, N.A. & Voitishek, A.V. (2006) *Numerical statistical modeling. Monte Carlo methods*, Publisher: Academia, 2006, 368 p.

Nechaeva, O. (2005) Neural Network Approach for Parallel Construction of Adaptive Meshes, *Lecture Notes in Computer Science, PaCT 2005*, Vol. 3606, Springer, Berlin Heidelberg, 2005, pp. 446-451

Nechaeva, O. (2006) Composite Algorithm for Adaptive Mesh Construction Based on Self-Organizing Maps, *Lecture Notes in Computer Science, ICANN 2006,* Springer Berlin/Heidelberg, Vol. 4131, 2006, p. 445-454

Nechaeva, O. (2007) Composition of Self Organizing Maps for Adaptive Mesh Construction on Complex-shaped Domains, *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)*, Bielefeld University, ISBN: 978-3-00-022473-7, 2007, 6 p.

Nechaeva, O. (2008) GeomRandom package for efficient random nonuniform distribution modeling on a surface and inside complex 3D shapes, *Electronic publication*, http://aitricks.com/geomrandom/about.htm, SB RAS, Novosibirsk, Russia, 2008

Nechaeva, O. I. (2004) Adaptive curvilinear mesh construction on arbitrary two-dimensional convex area with applying of Kohonen's Self Organizing Map, *Neuroinformatics and its applications: The XII National Workshop*, ICM SB RAS, Krasnoyarsk, 2004, pp. 101-102

Okabe, A.; Boots, B.; Sugihara, K. & Sung, N.Ch. (2000) *Spatial Tessellations – Concepts and Applications of Voronoi Diagrams*, 2nd edition, John Wiley, 2000, 671 p.

Prokopov, G. P. (1989) About organization of comparison of algorithms and programs for 2D regular difference mesh construction, *Preprint*, Keldysh Institute for Applied Mathematics, No. 18, Moscow, 1989

Shokina, N. Yu. (2001) Equidistribution Method For Adaptive Grid Generation, *Proceedings of 10th International Meshing Roundtable*, Sandia National Laboratories, 2001, pp.121-133

Thompson, J.F.; Warsi Z.U.A. & Mastin C.W. (1985) *Numerical grid generation, foundations and applications,* North-Holland, Amsterdam, 1985

Self-Organizing Maps
Edited by George K Matsopoulos

ISBN 978-953-307-074-2
Hard cover, 430 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

The Self-Organizing Map (SOM) is a neural network algorithm, which uses a competitive learning technique to train itself in an unsupervised manner. SOMs are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space and they have been used to create an ordered representation of multi-dimensional data which simplifies complexity and reveals meaningful relationships. Prof. T. Kohonen in the early 1980s first established the relevant theory and explored possible applications of SOMs. Since then, a number of theoretical and practical applications of SOMs have been reported including clustering, prediction, data representation, classification, visualization, etc. This book was prompted by the desire to bring together some of the more recent theoretical and practical developments on SOMs and to provide the background for future developments in promising directions. The book comprises of 25 Chapters which can be categorized into three broad areas: methodology, visualization and practical applications.