TASK QUARTERLY 4 No 4 (2000), 531–553

A NEW PROGRAM FOR RADICAL TESSELLATION CONSTRUCTION AND ANALYSIS

ROBERT LASKOWSKI

Department of Solid State Physics,
Faculty of Technical Physics and Applied Mathematics,
Technical University of Gdansk,
Narutowicza 11/12, 80-952 Gdansk, Poland
rolask@mif.pg.gda.pl

(Received 26 June 2000; revised manuscript received 4 September 2000)

Abstract: The purpose of this work is to describe usage of computer program SIMPL designed for construction and analysis of radical tessellations of any computer simulated sample. The radical polyhedra (RP) [Gellatly B J and Finney J L 1982 *J. Non-Cryst. Solids* **50** 313] can be considered as a generalization of the Voronoi polyhedra more widely used in literature. RP is a minimal polyhedron whose faces are sets of such points that distances from the points to tangent points of two neighboring atomic spheres are equal. The tessellation technique is a very efective tool for structural analysis of computer simulated samples, giving a great amount of easily accessible information. SIMPL allows to construct RP network for defined system and analyze shape, composition and mutual geometrical relation of radical polyhedra and radical simplices. Implemented capabilities give, for example, the posibility to recognize atomic environments' shape and investigate the non-local order in computer simulated materials. The pattern recognition technique [Laskowski R *et al.* 1997 *TASK Quart.* **1** 96] is based on analysis of the shape of the radical polyhedra, and contraction of short edges and small faces of the polyhedra. Non-local order analysis involves geometric relations between tessellation simplices.

Keywords: radical tessellation, stochastic geometry, structure recognition, computer modelling

1. Introduction

Stochastic geometry methods are a useful technique for the study of geometrical properties of computer simulated systems. They provide more information about the structure of the sample than the analysis based only on the straightforward interpretation of the angular distribution functions (ADF) and radial distribution functions (RDF), especially in the case of amorphous or multiphase materials. The most widely used techniques of the structural analysis are: common neighbors analysis (CNA) [1, 2], SO(3) invariants analysis [3–7] and various tessellation techniques [8–13].

The CNA method can be used to determine the abundance of inherent structures in a sample. Each pair of atoms from the first or the second RDF peak is described by sets of indexes. These sets determine to which peak a given pair belongs, the number of common nearest-neighbors, the number of pairs of the first neighbors in the set of

common neighbors, *etc*. The CNA is very efficient in the study of the nearest order in computer simulated amorphous materials. However, the method can give merely partial and ambiguous information about the types of crystalline phases in the analyzed samples. In particular, the pairs with the same sets of indexes can belong to different phases (*e.g.* both the face-centered (*fcc*), and the hexagonal-close-packed (*hcp*) structures contain identically indexed pairs).

The SO(3) invariants analysis is based on the construction of certain sets of the SO(3) invariants for local structure (local order or bond order parameters). The comparison of values of these invariants calculated for analyzed sample allows one to distinguish some predefined reference patterns. The sets of invariants suitable for the identification of the *fcc*, *hcp* and icosahedral structures are well known [5], but the extension of the method to other geometries is rather difficult: due to a limited number of invariants the conclusions about the local structure are not necessarily unambiguous. A similar method was used for the investigation of a non-local order [3]: the bond order parameters were constructed for all the atoms in the sample, making it possible to draw out some conclusions about the orientational order.

The tessellation method consists in the division of the total volume of the simulation box into an array of subvolumes belonging to the atoms. A number of various constructions is known. The most popular one is the partition into the Voronoi polyhedra (VP). VP is defined as the minimal polyhedron whose planar faces bisect at right angles the lines joining an atom to its neighbors; a pedantic definition is given by Brostow and co-workers [11]. However, one can use more general tessellation, based on radical planes [8]. In this case each atom from the array is characterized by an additional parameter, which can be interpreted as the atom radius. Vertex of a radical polyhedron (RP) is such a point, at which distances from the vertex to tangent points of four neighboring atomic spheres are equal. It is obvious that when all atomic radii are equal, the radical-plane tessellation is identical to the Voronoi partition. The RP diagram (alternatively RP network or RP graph), *i.e.* the set of RPs constructed for all the atoms in the sample, is rigorous; it splits in a unique manner the total sample volume into the zones owned by each atom. Radical simplices (RSs) are geometrically dual polyhedra to RPs; that is, a vertex of an RP is the central site of the corresponding RS, and each atom (center of a RP) is a vertex of the corresponding RS. One can assign each vertex of the former lattice to the elementary units of the latter one. The faces of RSs intersect the edges of RPs, and the faces of RPs intersect the edges of RSs. In the case of the Voronoi partition the simplices are called Delaunay simplices (DS). RP and RS (VP and DS, respectively) networks contain a huge amount of information about the structure of the analyzed sample. The difference between these two descriptions is in the access to this information. The shape of a RP reflects the arrangement of all the neighbors of the given atom. RSs represent the structure of clusters composed of four adjacent atoms. In the amorphous structures, RSs are disordered tetrahedrons, whereas RPs are more complex polyhedra [5]. The VP technique was applied to the analysis of the structure of the close packed [5, 12, 11, 10], and continuous network materials [13, 14] in a rather simple way; only some statistics on the geometric properties of the polyhedra were given. A more systematic and direct approach to the usage of the tessellation methods was proposed in [15, 16].

The purpose of this paper is to describe the usage of the computer program SIMPL, designed for the construction and analysis of radical tessellations. The core techniques

implemented in SIMPL were first described by Laskowski *et al.* in [15] and also applied in [16]. In order to make this work more comprehensive, the description of tessellation algorithm and the methodology of the networks treatment are presented in Section 2.1. Section 3 describes, how to set up and run SIMPL program. Section 4 gives examples of SIMPL applications. Section 5 contains final remarks.

2. Core techniques

2.1. Tessellation algorithm

In order to construct a network of radical polyhedra for a given atomic structure, two arrays of atoms should be defined: the centers array, and the neighbors array. The centers array contains the atoms, for which the RPs are to be determined. The neighbors array contains all the neighbors of the atoms of the centers array. The neighbors array can include atoms of the centers array. This initial decomposition makes the algorithm more flexible. For instance, in SiO_2 sample we can chose Si atoms as centers and all (O and Si) atoms as neighbors. Thus, we determine the RP for Si atoms only. The geometrical neighbors of Si atom could be either other silicon atoms, or oxygen atoms. If the two arrays are equal, we deal with the classical partition.

Let us introduce some auxiliary definitions, useful in further discussion. An *open center* is a center that is actually not contained in any of already constructed simplices. Two *simplices are contiguous* if they share a common face. Let *open simplex* be a simplex that has no contiguous simplices in the set of already constructed simplices. An *open face* is a face of a simplex that is not shared with any simplices already constructed (all the faces that do not contain any center are never open). The algorithm can be summarized as follows:

1. Create list of centers, list of neighbors, and empty list of simplices. Copy the centers list to an open centers list.
2. Choose an open center C from the open centers list. If the open centers list is empty, go to step 7, else go to the next step.
3. Construct the first simplex that contains C and three atoms from the neighbors array (an exact description of the construction is given below). Add this simplex to the simplices list. Remove C from the open centers list.
4. Choose from the simplices list a simplex S which has an open face. If such a simplex does not exist, go to step 6, else go to the next step.
5. Construct a new simplex D contiguous to S (for an exact description see below). The simplices must share the open face considered in step 4. A new fourth atom of D must be taken from the neighbors array. If it belongs simultaneously to the center array, remove it from the open centers list. Add D to the simplices list, and go to step 4.
6. Go to step 2.
7. The end of the RS construction procedure.

Steps 2 and 4 do not require any comments. The construction of the first simplex and the next simplices in steps 3 and 5 is more complicated. In our realization we adopted the routine proposed by Tanamura and co-workers [10]. Since their algorithm was written only for the Voronoi tessellation, here it is generalized on the case of the radical tessellation. The changes have been made in all the steps, were the distances between the atoms, and

the vertices or the positions of faces, are calculated. The suitable modifications base on the equation for the RP network vertices given by Gellatly and Finney [8]. At this stage it is convenient to introduce a routine of the degeneration detection, making the algorithm more stable. Degeneration occurs when one can construct more then one simplex (by adding an atom to three atoms defining simplex face) that have the same RP vertex coordinations. It is possible to construct an algorithm that can work with degenerated systems, but because the degeneration rarely occurs in practice, it would be computationally inefficient. In the case of degeneration we simply disturb the positions of atoms by a small fraction of the nearest-neighbor distance (of order 10^{-5}), and then we start the procedure from the beginning. The positions of atoms are restored after the procedure ends. In view of the further treatment of RP network, the perturbation of atomic positions does not have any influence on the final conclusions about the structure of the analyzed sample. Realization of step 5 is similar to the last step of the construction of the first simplexes. We know the face (in this case it is an open face) and we have to build a new simplex by adding an atom to this face. Care must be taken at selection of candidates for this fourth atom. We have to choose an atom which is placed on the opposite side of the face (in respect to the fourth atom of the simplex from step 4). After executing steps 1–7 described above we have the complete list of simplices of the analyzed sample. If centers array and neighbors array are identical, we have a full RS graph, else we have only its subgraph. In both cases, we can determine the set of radical polyhedra for all centers, utilizing only information contained in the simplices list.

The determination of the RS graph is only a preliminary step of our tessellation method. At this stage the RPs, which are geometrically dual do RSs, are rather complex polyhedra, containing many faces and vertices. Their topology does not allow to draw out any unambiguous conclusion on the structure. It is clear that the basic information about the local structure at any atom is carried mainly by big faces and long edges. The small faces and short edges have only inconspicuous meaning. To utilize the RPs efficiently we should first contract short edges and small faces of the RP graph. In RSs graph this contraction results in amalgamation of strongly distorted simplexes associated to vertices that are the ends of the contracted edge.

2.2. Graph contraction algorithm

To motivate the need of the contraction, let us consider the influence of small perturbations of atomic positions in an arbitrary crystalline lattice on their RP networks. In practice the source of this perturbation can be for example thermal motion of atoms about equilibrium lattice positions. A characteristic feature of certain crystalline networks (*e.g. fcc* or *hcp*) is the existence of degenerate vertices and edges in their RP graphs. Degenerate neighbors corresponding to such vertices or edges have been defined by Brostow and co-workers [11]. A degenerate vertex is common to more than four edges, while a degenerate edge is common to more than three faces of the RP graph. It is obvious that an arbitrary small displacement of atoms in the crystalline structure removes the degeneracy. In the place of a degenerate vertex, a small face or a short edge will appear, and degenerate edges will become small elongated faces. Figure 1 and Figure 2 demonstrate this concept. Figure 1 presents a distribution of face areas in a distorted *hcp* lattice; *fcc* leads to similar results. Positions of points in the ideal structure are shifted by a certain distance (perturbation displacement) at a random direction. The displacements are scaled to the nearest neighbor

distance. It is seen that the faces of polyhedra can be grouped together into two sets, which contain only small and only large faces. We can assume that small faces are of the perturbative origin. Figure 2 presents distribution of edge lengths for a distorted hcp lattice after the face elimination. It is seen that also the edges can be divided into two subsets, and the short ones are of perturbative origin. The local arrangement is practically the same in perturbed and in unperturbed structures, but the topology of radical polyhedra is much more complicated in the former case. Thus, eliminating short edges and small faces from the RP network (by contracting them to vertices or edges), we remove the effects due to the perturbations and simplify the polyhedra topology. The same result could be achieved by suitable displacements of the atoms. However, since we have no information about the individual fluctuations of the atoms positions in the sample, such a procedure cannot be realized in practice. In a structure in which the degenerate vertices are absent (*e.g. bcc* lattice), a small perturbation of the position does not change the topology of the network. Thus, the analysis of such structures can be performed without any contraction.

To proceed with the graph contraction one needs to find the set of edges that are to be contracted. This can be done directly by computing all edges lengths and selecting all the edges shorter than a certain predefined threshold value. The threshold is to be determined on the basis of the edge length distribution. The same can be done with the faces, but in this case special attention should be paid to contraction of elongated faces, which originate from degenerate edges. As it is seen in Figure 1 and Figure 2 the contraction threshold is equal to 0.25 in the case of faces, and 0.5 or more in the case of edges.

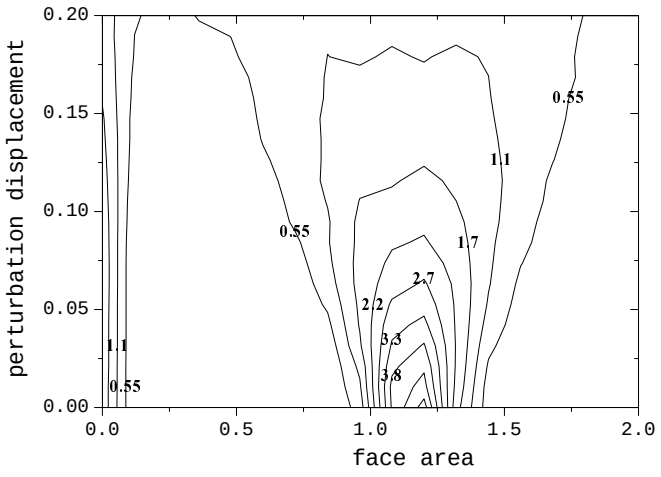


Figure 1. *The distribution function of the face areas, averaged for all RPs, plotted for a perturbed hcp lattice. The numbers on the lines represent the average number of faces. The points in the structure are shifted by the perturbation displacement in a random direction. The displacements are scaled to the nearest neighbor distance, and face areas are expressed in units of the average face area. Because in this case the lattice is monoatomic, the RPs are simply the Voronoi polyhedra*

In the case of large perturbation, as it is seen in Figure 2, it is difficult to establish the threshold value, because the large peak of longer edges can not be separated from the

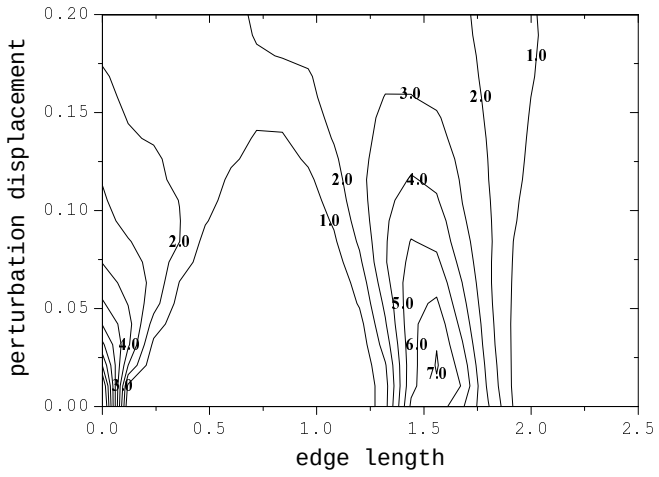


Figure 2. *The distribution of the edge lengths, averaged for all RP, plotted for a perturbed hcp lattice after the elimination of small faces. The numbers on the lines represent the average number of edges. The edge length is scaled to the average edge length. Because in this case the lattice is monoatomic, the RPs are simply the Voronoi polyhedra*

peak of shorter edges. To avoid the difficulty SIMPL uses an algorithm which is able to analyze subgraphs of the RS network. Each of the simplices belonging to any of these subgraphs should contain a specified central atom. Contraction of the whole RS network is performed subsequently in each subgraph. After each step the RS network is updated, and the subgraphs are recalculated. The algorithm can be summarized as follows:

1. Take a center C from the centers list. C should not be considered previously. If it is not possible go to step 7.
2. Select a subgraph S of the RS network. All simplices from the subgraph must contain C. The set of atoms constituting S is an atomic polyhedron related to C, in the case of no degeneration. Else, it contains in addition all degenerated neighbors.
3. Determine the radical polyhedron P of center C, using the subgraph S (in this step care must be taken for the selection of vertices, because not all vertices associated with simplices from S are included in P; see the text below). Evaluate edge lengths of this polyhedron.
4. Select a set E of the edges of P, that are not contained in the previously contracted polyhedra. It can be done simply by checking a composition of the simplices which define the vertex of the edges. If the simplices associated to both vertices of the edge contain the center considered previously, the edge is not being selected.
5. Contract the subgraph S, using the list E (for exact description see below), and update the RS network.
6. Go to step 2.
7. The network is contracted.

Some of the above steps should be elucidated. In step 3 we determine a polyhedron using a subgraph of the RS network. This can be done by examining the geometrical relations

between the simplices from S. The vertices associated to these simplices constitute the polyhedron. Two mutually contiguous simplices in S define the edges of a polyhedron. The faces are defined by a set of simplices that contain the central atom, and its geometrical neighbors. Because the RS network is now after several contractions of the subgraphs determined for other centers, it is possible that some vertices on the polyhedron constructed in such a way are topologically incorrect. There may occur vertices in which only two edges meet. Topologically correct situation is when three or more edges of a polyhedron meet in a vertex. Taking it into account one must reconstruct the polyhedron removing all the incorrect vertices and changing the geometrical relation between the simplices. The simplices that are contiguous to incorrect vertex become contiguous to each other, but only for the moment in which the polyhedron is being constructed. In step 5 we utilize our previous technique of Voronoi polyhedra contraction [17]. The algorithm applied here allows for a very careful contraction of subgraph S, by an appropriate selection of contracted edges and permanent control of polyhedron shape during the process. Let us itemize the main points of this technique:

1. For the RP under consideration contract all the edges shorter than a certain fraction x of the average edge length.
2. Find the shortest edge.
3. If the edge is shorter than a fraction y ($y > x$) of the average edge, check the shape of the polyhedron (for description see the next subsection). If the shape belongs to the set of the predefined patterns, take the next polyhedron, and go to step 1; otherwise contract the edge under consideration, and go to step 2.
4. If the edge is longer than a fraction y of the average edge end the procedure.

As it is seen the edges contraction algorithm involves procedures of polyhedron shape recognition, which may be avoided setting parameter x equal or greater than y. Detailed tests of the algorithm efficiency accomplished for monoatomic metallic MD-simulated samples allowed to establish the optimal values of the parameters x and y to be 0.3 and 0.6, respectively. $x < 0.3$ results in switching on the shape recognition procedure sooner; if the shape of a given polyhedron is undefined, there are no consequences except the slowing down of computations. On the other hand, $y > 0.6$ for an undefined shape may result in too many contractions, leaving eventually a polyhedron with only a few edges and faces.

2.3. Polyhedron shape recognition

Let us turn to the problem of the polyhedron shape recognition. The shape of an arbitrary polyhedron is described in SIMPL by three sets of integers, as:

$$F = (f_3, f_4, f_5, \ldots)$$
$$V = (v_3, v_4, v_5, \ldots)$$
$$E = (e_4, e_5, e_6, \ldots).$$

Here f_i is the number of the i-edged faces in the polyhedron; v_i is the number of the vertices, from which exactly i edges do originate. In the case of a non-degenerated RP, only v_3 do not vanish, hence $i-3$ equals to the degeneration degree. Finally, e_i is the number of edges for which i equals to $j+k+4$, where j and k are the degrees of degeneration of both vertices associated to the edge. It is not complicated to introduce more sets like these (*e.g.* in the form of two or three dimensional array), but our experience shows, that the sets F, V,

and E, defined above suffice to perform an efficient structural analysis. Two polyhedra are said to have the same topological structure if they have the same F, V, and E sets. Thus, we can compare all the constructed polyhedra with an arbitrary set of reference polyhedra. For instance, a *fcc* polyhedron has $F=(0,12)$, $V=(8,6)$, $E=(0,24)$, and a *hcp* polyhedron has $F=(0,12)$, $V=(8,6)$, $E=(3,18,3)$.

We have tested the efficiency of the contraction and recognition method on some perturbed crystalline lattices. As previously in Figure 1 and Figure 2, the lattice points are shifted by a certain distance in a random direction. All the polyhedra in the structures tested were correctly recognized for the displacement range lower than 0.13 of the nearest neighbor distance. Figure 3 presents the results of the pattern recognition applied to fluctuated hcp structures. No hcp polyhedra have been detected in the structures perturbed within displacement greater than 0.25.

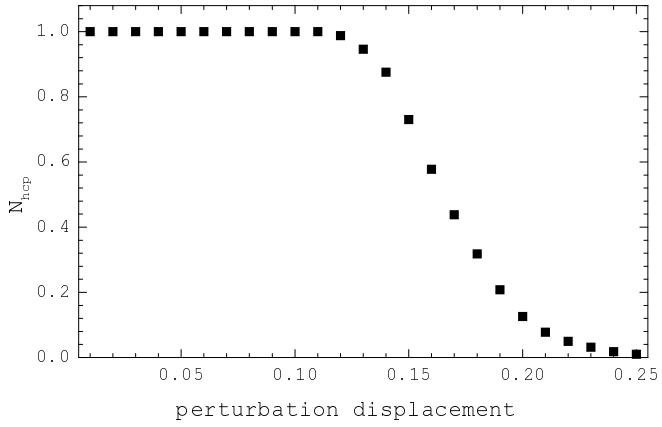


Figure 3. *The dependence of the fraction N_{hcp} of the recognized polyhedra on the perturbation displacement*

3. Setup and running SIMPL

SIMPL is written in FORTRAN77, to setup maximum system size one should edit file *simpl.par*, and recompile sources. The program needs two input files. The first one named *simpl.ini* contains commands describing the tasks to be performed. The second one is an atom configuration file. At the beginning of execution SIMPL scans *simpl.ini* (until it encounters *end* command) ignoring empty records and those that begin with '#', all other records are interpreted as command records. Reading a command record the program extracts task identifier and parameters, and checks correctness of the syntax. In the case of the syntax error, interpretation is terminated giving a short description of the problem. If all the commands are correctly interpreted, SIMPL starts to perform suitable tasks sequentially, writing results to file *simpl.out* (some commands create additional output file). The interpretation process is case insensitive. The full list of commands is presented in the next subsection.

In general, SIMPL has been designed to perform tessellation of a given atomic system, contraction of RS network, and analysis of contracted RS and RP networks. Commands and

tasks that SIMPL performs can be grouped in three sets: preparatory commands, tessellation commands, and analysis commands group. Preparatory commands allow one to open the configuration file, read in suitable part of this file, match records of configuration file with atoms names and atomic radius, and also defines periodic boundary box.

Bellow, a typical preparatory part of *simpl.ini* is presented:

```
Open_File_xyz XYZ.xyz

Skip_Rec 2
read_rec 2916

box
 54.2997225303 2.1238777148 0.0583893457
 2.0696896577 51.3836355703 0.099681211
 0.0902413917 0.1495607081 31.4948373164
 1.0

atom_spec
 Pb1,1,   1000, 1.0
 Pb2,1001,2916, 1.0

neig_atoms Pb1,Pb2
cent_atoms Pb1
setup_conf
```

Command *Open_File_xyz* informs the program that configuration file name is *XYZ.xyz*, and its format is of *xyz* type (see also *open_file_asc* command). *Skip_Rec* moves file pointer by two records down in configuration file, in order to skip headers, for example. Next 2916 records are read due to *read_rec* command. SIMPL reads only coordinates, atoms names are ignored. *box* command initializes definition of a configuration box. SIMPL imposes the periodic boundary conditions to analyzed configuration. The first three records after *box* command define cell vectors, the next record contains scaling factor (cell vectors and coordinates of an atom are multiplied by this scaling factor). *atom_spec* command begins the block of atom specifications. It initializes matching of atoms names and radius with records of configuration file. In our case records from 1 to 1000 are declared to be Pb1 atoms with 1.0 radius, and records from 1001 to 2916 are Pb2 atoms with 1.0 radius. All records declared in *read_rec* must have a specification, and no record must have two or more different specifications. Because the number of specifications is arbitrary, empty record must appear after the last one in order to inform SIMPL to end of reading the specification block. Two next commands *neigh_atoms* and *cent_atoms* define, which atoms should be treated as centers and, which are neighbors. Preparatory command group must be ended by *setup_conf* command, which activates all declared settings.

Having defined configuration SIMPL is ready to perform tessellation. At this stage one can also perform common neighbor analysis, using *cna* command. This part of *simpl.ini* might look like an example below:

```
del_net
contract_del_net  0.2,0.5
vp_from_dc 1
```

The first command *del_net* triangulates the specified configuration according to the algorithm described in Section 2.1. *contract_del_net* command activates contraction of the resulted RS network using contraction algorithm presented in Section 2.2. The parameters

control the contraction process, and are described in Section 2.2 (x and y). *vp_from_dc* command extracts radical polyhedra form the contracted RS network. The parameter indicates that data concerning edges length, faces area and volumes are also to be extracted. In order to execute a slightly modified contraction algorithm, which has better performance in recognition of the local neighborhood shape, one should use *vp_indep* command instead of *contract_del_net* and *vp_from_dc*. For example:

```
vp_indep 0.1, 0.5, 0.1, 1
```

vp_indep treats all radical polyhedra individually, which may cause some topological inconsistencies in the RP network. The first two parameters have a similar meaning to the parameters of *contract_del_net* command. The third one controls the process of small faces neglecting (the value is threshold in unit of average face area of a polyhedron). In our case all faces smaller than 0.1 of the average face area are neglected before the contraction process. The last parameter is similar to the parameter *vp_from_dc* command.

Because common neighbor analysis (CNA) [1, 2] is a very useful and simple method of description of local order it is implemented in SIMPL. One should note that the execution of CNA does not need any tessellation, but only definition of configuration data. CNA is invoked by *cna* command in the way presented below:

```
cna
Pb1,Pb1,Pb1
0.0, 4.0
0.0, 4.0
0.0, 4.0
0.0, 4.0
rdf_1.dat, 50
```

The first record after *cna* contains identifier of atoms from the CNA pair and identifier of their common neighbor. In our case these three are Pb1. The second record defines the range of distance for the pair atoms. The third and the fourth records define the distance range for the first and the second atom from the pair, and their common neighbor. The fifth record defines the distance range for common neighbor. The last record contains file name, where decomposition of pair distribution function (between *the pair atoms*) is also written, and a number of bins.

After tessellation and contraction, one can use the RP network to establish the geometrical properties of the analyzed structure in many various ways. A particular realization depends on the questions one asks. SIMPL is able to perform a couple of basic tasks. For example:

```
shapes  Pb1
vp_vol_diag  Pb1,vol.dat,50

select_cent  fcc

pdb_vp  Pb1, fcc.pdb
cluster_vp Pb1

select_cent  all
select_cent  bcc

pdb_vp  Pb1, bcc.pdb
cluster_vp Pb1
```

shapes command analyses local neighborhood of Pb1 atoms and tries to recognize its shape writing it to file *simpl.out*. *vp_vol_diag* command creates file vol.dat containing distribution of volumes of Pb1 polyhedra. *select_cent* command restricts further analysis to *fcc* polyhedra only. *pdb_vp* creates pdb file containing Pb1 atoms, which have *fcc* like local neighborhood. *cluster_vp* writes to *simpl.out* information about cluster properties of Pb1 atoms with *fcc* local neighborhood. Next *bcc* polyhedra are selected, pdb file and cluster data are output. In the case when implemented in SIMPL analysis capabilities do not fit ones needs, RP and RS network might be written to a file using *write_vp* and *write_dc* commands.

3.1. Commands list

3.1.1. Commands defining atomic configuration

▷ *box*
initializes definition of configuration cell. Next four records of *simpl.ini* must contain three vectors defining the cell and an unit of length. SIMPL imposes periodic boundary condition to the sample.
Syntax:

```
box
 cell(1) cell(2) cell(3)
 cell(4) cell(5) cell(6)
 cell(7) cell(8) cell(9)
 unit_length
```

where *cell(1) cell(2) cell(3)* define the first vector, *cell(4) cell(5) cell(6)* define the second vector, *cell(7) cell(8) cell(9)* define the third vector.

▷ *open_file_asc*
opens configuration file. SIMPL expects four–column ASCII file, one atom per record (first integer number, next coordinates:

```
1 -27.90219274 -26.06457700 -15.70217249
2 -25.69346326 -23.39167052 -15.56629017
3 24.14252009  -18.58279617 -15.74169706
```

integer is ignored).
Syntax:

```
open_file_asc  filename
```

▷ *open_file_xyz*
opens configuration file. SIMPL expects configuration file to be in *xyz* format (atom name followed by coordinates:

```
Pb -27.90219274 -26.06457700 -15.70217249
Pb -25.69346326 -23.39167052 -15.56629017
Pb 24.14252009  -18.58279617 -15.74169706
```

atom name is ignored).
Syntax:

```
open_file_xyz filename
```

▷ *skip_rec*
moves file pointer by a defined number of records of configuration file.

Syntax:

```
skip_rec  n
```

▷ *read_rec*
reads a number of records of configuration file. First, file must be opened with *open_file_asc* or *open_file_xyz* command. In both cases only coordinates are read.
Syntax:

```
read_rec  n
```

▷ *atom_spec*
initializes atomic specification. It relates atomic names and radius to read-in records of configuration file. Each of the following records contains species name, indexes of the first and the last records related to this name, and radius of this atom. The last record must be empty.
Syntax:

```
atom_spec
  name, first record, last record, radius
  ...
  ...
  ...
empty record
```

▷ *cent_atoms*
defines a list of centers (see Subsection 2.1).
Syntax:

```
cent_atoms  name1, name2,...
```

▷ *neig_atoms*
defines a list of neighbors (see Subsection 2.1).
Syntax:

```
neig_atoms  name1, name2,...
```

▷ *setup_conf*
actualizes configuration data (last command in preparatory commands group).
Syntax:

```
setup_conf
```

3.1.2. Tessellation and contraction commands

▷ *del_net*
performs radical tessellation (see Subsection 2.1).
Syntax:

```
del_net
```

▷ *contract_del_net*
contracts RS network. Command needs two parameters controlling contraction procedure (x, y described in Subsection 2.2).

Syntax:

```
contract_del_net  x, y
```

▷ *vp_from_dc*
extracts radical polyhedra from RS network. RP are constructed only for atoms from centers list.
Syntax:

```
vp_from_dc  n
```

n controls output level. For *n* equal to 0 only information concerning polyhedra shapes and geometrical neighbors is extracted, for *n* equal 1 also length of edges, area of faces and volumes are calculated.

▷ *vp_indep*
controls modified version of the contraction algorithm. In standard version (initialized by *contract_del_net*) whole RP network is contracted, keeping it dual to RS network. Modified version makes contraction of each polyhedron individually, which increases efficiency in recognition of shape of local neighborhood.
Syntax:

```
vp_indep  x, y, p, n
```

x and *y* control contraction like in *contract_del_net*, *n* controls output level like in *vp_from_dc*. *vp_indep* allows one to get rid of small faces at the beginning of contraction, *p* is a threshold (in unit of average face area of a polyhedron).

3.1.3. Analysis commands

▷ *end*
terminates execution of SIMPL.
Syntax:

```
end
```

▷ *write_vp*
writes out composition of all constructed radical polyhedra to file of a given name (one record per polyhedron, first index of a polyhedron, next number of geometrical neighbors, and indexes of the neighbors).
Syntax:

```
write_vp  file
```

file is output file name.

▷ *write_dc*
writes out composition of all contracted radical simplices to file of a given name (one per record, first index, next number of composing atoms, indexes of these atoms, position of vertex).
Syntax:

```
write_dc file
```

file is output file name.

▷ *shapes*
writes to output file (*simpl.out*) statistic of local neighborhood shapes. Program can recognize only shapes defined in procedure DEF_SHAPE (file *def_pol.f*, where polyhedra topology corresponding to each local neighborhood are coded). Each defined shape is identified by a shortcut. Actually there are 14 defined shapes: *fcc* (local neighborhood corresponding to face centered cubic structure), *bcc* (body centered cubic), *hcp* (hexagonal close packed), *kub* (simple cubic), *ico* (icosahedron), *tet* (tetrahedron), *bpt* (triangle bipiramid), *bpp* (pentagon bipiramid), *pis* (square piramid), *pip* (pentagon piramid), *prt* (triangle prism), *prs* (square prism), *prp* (pentagon prism), *fhp* (*fcc*+*hcp*). In this case undefined local neighborhood are identified by *und*.
Syntax:

```
shapes  p
```

where p is an atom name from center list.

▷ *colors*
writes to output file information about composition of local neighborhood.
Syntax:

```
colors  p,
```

where p is an atom name from center list.

▷ *select_cent*
selects centers of given types of local neighborhood. The following output is only for selected centers (it does not concern *tet_diag*, *edg_len_diag* and *cna*, where output is created and written for all centers).
Syntax:

```
select_cent  all
```

selects all centers.

```
select_cent  [not] p1, p2,..., pn
```

selects centers with given shapes of local neighborhood ($p1,\ldots,$ pn are shape identifiers – *fcc*, *bcc*, *hcp*, *tet*,…). Putting *not* at the beginning of the parameters list results in selecting all centers except the centers appearing in the parameters list.

```
select_cent  col p1, p2,..., pn
```

selects centers with a given composition of local neighborhood. It means centers, which have $p1$ neighbors of first type (according to neighbor list), $p2$ – second type, *etc*. If $p1$, $p2,\ldots,$ pn are negative, program select centers with number of neighbors of given type greater or equal to $-pi-1$.

▷ *face_area_diag*
writes out diagram of faces area (in box unit).
Syntax:

```
face_area_diag  c, n, file_name, nbin
```

where c is the center type identifier (from centers list), n is the neighbor type identifier (from neighbors list), *file_name* is name of file, where the diagram is written to, *nbin* is number of bin in the diagram.

▷ *vp_area_diag*
for a given centers type, writes out diagram of fraction of polyhedron area, that consists of faces originated from a given type of neighbors.
Syntax:

```
vp_area_diag  c, n, file_name, nbin
```

c is the center type identifier (from centers list), *n* is the neighbor type identifier (from neighbors list), *file_name* is name of file, where the diagram is written to, *nbin* is numbers of bin in the diagram.

▷ *edg_len_diag*
writes out diagram of edges length.
Syntax:

```
edg_len_diag  file_name, nbin
```

file_name is name of file, where the diagram is written to, *nbin* is numbers of bin in the diagram.

▷ *tet_diag*
writes out diagram of tetrahedricity ($T = \sum_{i,j}(l_i - l_j)^2/l^2$, i, j are the indexes of simplex edges, l is edge length) of radical simplices (before contraction).
Syntax:

```
tet_diag  file_name, nbin
```

file_name is name of file, where the diagram is written to, *nbin* is numbers of bin in the diagram.

▷ *vp_vol_diag*
writes out diagram of polyhedra volume.
Syntax:

```
vp_vol_diag  c, file_name, nbin
```

where *c* is the center type identifier (from centers list), *file_name* is name of file, where the diagram is written to, *nbin* is numbers of bin in the diagram.

▷ *dc_size*
writes out to *simpl.out* statistic of size contracted radical simplices.
Syntax:

```
dc_size
```

▷ *cluster_vp*
writes out to *simpl.out* information of cluster properties of given atoms.
Syntax:

```
cluster_vp  c, n1,..., ni
```

c is the center type identifier (from center list), *n1*,..., *ni* are the neighbor type identifier (from neighbor list). If there are no neighbors in the parameter list ($i = 0$) the program considers clusters of *c*-type atoms being geometrical neighbors (they share polyhedra faces). For $i > 0$ clusters are composed of *c*-type atoms sharing *n1*,..., *ni* geometrical neighbors.

▷ *cluster_dc*
writes out to *simpl.out* information of cluster properties of contracted radical simplices.
Syntax:

```
cluster_dc  size, ns, [t1, t2]
```

size is the size of contracted simplex, *ns* is a number of shared atoms of contracted simplex in a cluster. In the case of *size* = 4, *t1* and *t2*, defining range of tetrahedricity, must be specified.

▷ *dc_comp*
writes out to *simpl.out* information of composition of contracted radical simplices.
Syntax:

```
dc_comp
```

▷ *pdb_vp*
writes out centers of a given name to *pdb*-formatted file (to view in *rasmol* program).

```
pdb_vp  c, file
```

c is the center type identifier (from center list), *file* is output file name.

▷ *cna*
initializes CNA (common neighbors analysis) and writes the results to *simpl.out*.
Syntax:

```
cna
atom1, atom2, atom3
r12_1, r12_2
r13_1, r13_2
r23_1, r23_2
r33_1, r33_2
rdf_name, nbin
```

atom1, *atom2*, *atom3* are a type identifier of respectively the first and the second atom from the CNA pair, and a common neighbor of these atoms, *rij_k* defines the range of distance between atom i and j, *rdf_name* is the name of file, where decomposition of pair distribution function (between *atom1* and *atom2*) is written to, *nbin* is number of bins. *rdf_name* is ASCII file, first column is tabulated distance, next columns contain RDF data in the order in which corresponding information appears in *simpl.out*.

4. Example application

4.1. Multiphase Pb sample

Lets us first analyze structure of monoatomic Pb sample. The sample is an end result of a numerical experiment, where influence of external pressure on sample structure has been investigated [18]. On increasing of the external pressure on the initial *fcc* crystal, the structural phase transition to *bcc* has been detected. After this, during decreasing pressure, transition to close packed structure occurred. Because *fcc* and *hcp* structures are energetically undistinguished by the interatomic potential used, the sample should be a mixture of *fcc* and *hcp* structures. In this case, SIMPL is used to recognize shape of local neighborhood. Below is the contents of used *simpl.ini*:

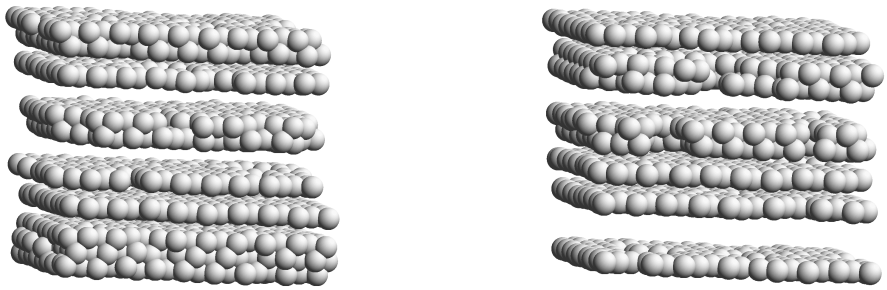
```
Open_File_xyz XYZ.xyz
Skip_Rec 2
read_rec 2916
box
54.2997225303 2.1238777148 0.0583893457
2.0696896577 51.3836355703 0.0996812118
0.0902413917 0.1495607081 31.4948373164
1.0

atom_spec
Pb,1,2916, 1.0

neig_atoms Pb
cent_atoms Pb

setup_conf
# end of preparation, start analysis

del_net

vp_indep 0.2, 0.5, 0.0,0
shapes Pb

select_cent fcc
pdb_vp Pb fcc.pdb

select_cent all
select_cent hcp
pdb_vp Pb hcp.pdb

# end of analysis

end
```

As it is seen tessellation is done for all atoms as centers and neighbors. RP network is contracted using *vp_indep* command. Recognition procedure is activated by *shapes* command, and *pdb* files are generated for *fcc* and *hcp* subsystems using *select_cent* and *pdb_vp* commands. Resulted *simpl.out* file is presented below:

```
file: XYZ.xyz XYZ

number of skipped records: 2

number of read records: 2916

box: 54.2997 2.1239 0.0584
     2.0697 51.3836 0.0997
     0.0902 0.1496 31.4948
box unit: 1.0000
bx, by, bz: 54.3413 51.4254 31.4953
alfa, beta, gama: 85.4534 89.7638 89.6105

--------------------------------
     system specification
--------------------------------
atom      points      radius
--------------------------------
Pb      1    2916     1.000
--------------------------------

neighbors list: Pb
```

```
centers list : Pb

Triangulation of specified system is done.
Number of simplexes in DS network is 17597

VP are now independent, each is contracted with parameters:
0.2000 0.5000 0.0000

-----------------------------------------------------------------
                  VP SHAPE ANALYSIS
-----------------------------------------------------------------
shape of local neighborhood   |  fcc    hcp    und    bcc
number of Pb in % of selected |  50.7   38.3   10.9   0.0
-----------------------------------------------------------------

The centers with local neighborhood shapes: fcc are selected,
so the following output is only for these centers.
Number of selected centers is 1479

VP network is in file fcc.pdb

All type centers are selected, so the following output
is for all centers.

The centers with local neighborhood shapes: hcp are selected,
so the following output is only for these centers.
Number of selected centers is 1118

VP network is in file hcp.pdb
```

It is seen that all used commands generate self-explanatory output into *simpl.out* file. For example: command *shapes* creates table of recognized local neighborhood shapes. More than 50% is *fcc* type, and about 40% is *hcp* type. *fcc* and *hcp* subsystems are in *fcc.pdb* and *hcp.pdb* files, which can be visualized by (for example) *rasmol* program (see Figure 4).



(a) atoms with *fcc* local neighborhood

(b) atoms with *hcp* local neighborhood

Figure 4. *Results of local neighborhood shape analysis*

4.2. Na dopped disilicate glass

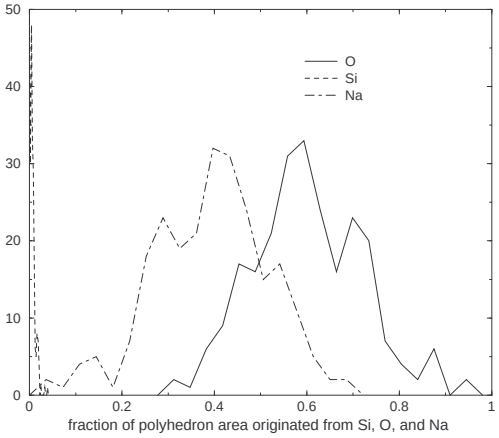
As an example of application of SIMPL to multiatomic systems let us analyze structure of Na dopped disilicate glass. In this case we are interested in local neighborhood of Na and Si atoms. Below *simpl.ini* file is proposed:

```
box
24.1790000000 .0000000000 .0000000000
.0000000000 24.1790000000 .0000000000
.0000000000 .0000000000 24.1790000000
1.0

atom_spec
Na,1, 240, 1.86
Si,241, 480, 1.18
O, 481, 1080, 1.40

neig_atoms Na,Si,O
cent_atoms Na,Si

Open_File_xyz XYZ.xyz
Skip_Rec 2
read_rec 1080

setup_conf

del_net
vp_from_dc 1

vp_area_diag Si,O, sio.dat,20
vp_area_diag Si,Si, sisi.dat,20
vp_area_diag Si,Na, sina.dat,20
vp_area_diag Na,O, nao.dat,20
vp_area_diag Na,Si, nasi.dat,20
vp_area_diag Na,Na, nana.dat,20

vp_vol_diag Si, volsi.dat, 20
vp_vol_diag Na, volna.dat, 20

vp_indep 0.1, 0.5, 0.2, 1

shapes Si
colors Si
shapes Na
colors Na

end
```

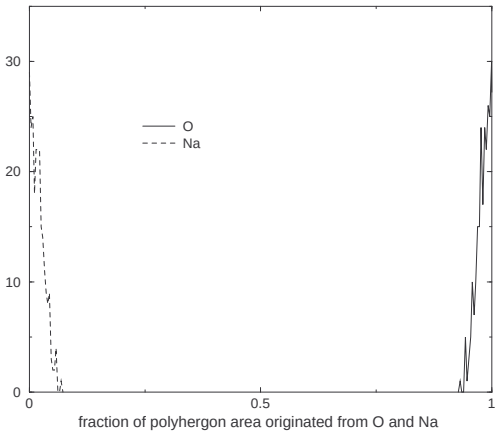
In the previous subsection the sample was monoatomic, so the value of atomic radius had no significance. Now the radiuses are equal to known chemical values. At the beginning we decided to calculate diagrams of fraction faces area originated from different neighbors for Si and Na polyhedra in uncontracted RP network. In both cases *vp_area_diag* command is used. Faces originate from O, Si and Na are analyzed separately. Results are presented in Figure 5. Diagrams of volume of Si and Na polyhedra are generated using *vp_vol_diag* command and plotted in Figure 6.

Contraction process realized by *vp_indep* command removes Na originated face from Si polyhedra. As it is seen in *simpl.out* almost all Si polyhedra are terahedrons with O originated faces. Shape (*shape* command) and composition (*colors* command) of Na polyhedra are more complicated. Below, the content of the resulting *simpl.out* file is presented:

```
box: 24.1790 0.0000 0.0000
     0.0000 24.1790 0.0000
     0.0000 0.0000 24.1790
```



(a) Na polyhedrons



(b) Si polyhedrons

Figure 5. *Atomic contribution to polyhedron area for Si and Na*

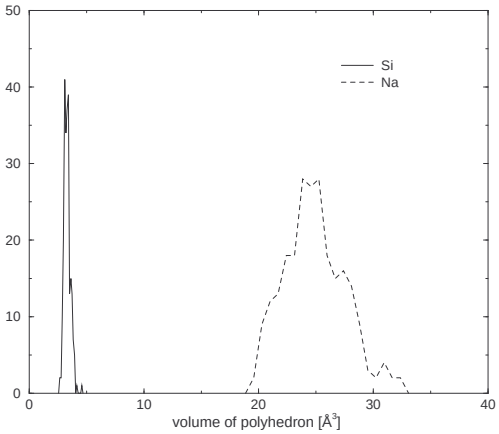


Figure 6. *Volume distribution of Si and Na polyhedra*

```
box unit: 1.0000
bx, by, bz: 24.1790 24.1790 24.1790
alfa, beta, gama: 90.0000 90.0000 90.0000

--------------------------------
      system specification
--------------------------------
atom      points      radius
--------------------------------
Na      1     240     1.860
Si    241     480     1.180
O     481    1080     1.400
--------------------------------

neighbors list: Na Si O

centers list : Na Si

file: XYZ.xyz XYZ

number of skipped records: 2

number of read records: 1080

Triangulation of specified system is done.
Number of simplexes in DS network is 6233

VP are evaluated directly from DC network
VP area diagram is in file sio.dat
VP area diagram is in file sisi.dat
VP area diagram is in file sina.dat
VP area diagram is in file nao.dat
VP area diagram is in file nasi.dat
VP area diagram is in file nana.dat
VP volume diagram is in file volsi.dat
VP volume diagram is in file volna.dat

VP are now independent, each is contracted with
parameters: 0.1000 0.5000 0.2000

-----------------------------------------------
          VP SHAPE ANALYSIS
-----------------------------------------------
shape of local neighborhood   | tet bpt
number of Si in % of selected | 98.7 1.2
-----------------------------------------------

-------------------------------------------------
                           | number of neighbors
number of Si atoms in %    |------------------------
of selected                | Na     Si     O
---------------------------|------------------------
         98.7              |  0      0     4
          0.8              |  1      0     4
-------------------------------------------------

-----------------------------------------------
           VP SHAPE ANALYSIS
-----------------------------------------------
shape of local neighborhood   |  und      ico
number of Na in % of selected |  99.6     0.4
-----------------------------------------------
```

number of Na atoms in % of selected	number of neighbors Na	Si	O
1.2	5	0	7
2.9	7	0	8
1.7	2	0	8
2.1	6	0	8
1.2	7	0	9
1.2	6	0	10
2.9	3	0	8
6.2	6	0	7
8.3	5	0	8
5.8	4	0	8
1.7	4	0	6
1.7	4	0	12
5.0	5	0	9
4.2	6	0	6
4.6	4	0	7
1.7	7	0	7
2.9	3	0	9
4.6	5	0	6
3.3	4	0	10
1.2	7	0	10
1.2	2	0	9
2.9	4	0	9
1.2	8	0	5
2.5	3	0	7
1.7	5	0	10
1.7	4	0	11
2.5	6	0	9
2.9	7	0	6
2.1	3	0	10
2.1	2	0	10

5. Summary and conclusions

Tessellation based methods are an effective approach to the structural analysis of computer simulated sample in the case of close packed and continuous network materials. Radical polyhedra network contains great amount of easily accessible information about structure of the sample. For example: analyzing shape and composition of radical polyhedra one has direct access to information about the structure of the nearest neighborhood; analysis of geometrical relation between polyhedra or radical simplices gives information about non local order in the sample.

In this paper we have described base theory and usage of computer program SIMPL designed for structural analysis of simulated materials. The program utilizes radical tessellation method, however CNA (common neighbor analysis) is also implemented. In general the structural analysis is performed in three stages. In the first step, mutually dual radical polyhedra and radical simplices networks are constructed. The algorithm of tessellation implemented in SIMPL is described in Subsection 2.1. Each radical polyhedra diagram constructed for any sample contains some excess information concerning thermal motion or other meaningless fluctuations of atomic position. Thus, in the second stage, the networks are contracted in order to clean the structure from the effects of these fluctuations.

The contraction of faces and edges of the polyhedra has a clear geometric significance: it simply changes the category of neighbor. For instance in *hcp* structure, the contraction algorithm allows one to separate the direct and the degenerate neighbors, which makes local order recognition possible. The contraction algorithm is described in Subsection 2.2. After construction and contraction the resulted radical polyhedra and radical simplices networks can be put into further analysis using implemented capabilities of SIMPL or simply exported to an external file. The analysis can be performed in many various ways which depend on particular questions one asks. In order to make SIMPL as flexible as possible it has been designed as commands interpreter. Section 3 describes usage and setup of SIMPL and also contains the list of all directives and commands actually implemented.

6. Availability

SIMPL program is available free from *www.task.gda.pl/software*.

References

[1] Falken D and Jonsson H 1994 *Comput. Matter. Sci.* **2** 279
[2] Posada-Amaralis A and Garzon I 1996 *Phys. Rev. B* **53** 8363
[3] Steinhard P, Nelson D and Ronchetti M 1983 *Phys. Rev. B* **28** 784
[4] Mitu A and Patashinskii A 1988 *Physica A* **150** 371
[5] Ogata S and Ichimaru S 1989 *Phys. Rev. A* **39** 1333
[6] Gades H and Mitu A 1991 *Physica A* **176** 297
[7] Mitu A, Marx D, Sengupta S, Nielaba P, Patashinskii A and Hahn H 1993 *J. Phys. CM* **5** 8509
[8] Gellatly B J and Finney J L 1982 *J. Non-Cryst. Solids* **50** 313
[9] Finney J L 1979 *J. Comp. Phys.* **32** 137
[10] Tanemura M, Ogawa T and Ogita N 1983 *J. Comp. Phys.* **51** 191
[11] Brostow W, Dussault J-P and Fox B L 1978 *J. Comp. Phys.* **29** 81
[12] Finney J L 1970 *Proc. Royal Soc. London A.* **319** 479
[13] Medvedev N N 1986 *J. Comp. Phys.* **67** 223
[14] Hiwatari Y and Saito T 1984 *J. Chem. Phys.* **81** 6044
[15] Laskowski R, Rybicki J and Chybicki M 1997 *TASK Quart.***1** 96
[16] Brostow W, Chybicki M, Laskowski R and Rybicki J 1998 *Phys. Rev. B* **57** 13448
[17] Laskowski R and Alda W 1996 *Workshop on Aperiodic Structures* Cracow, Poland, 1–5 July 1996
[18] Laskowski R 2000 *Phase transitions in metals: MD studies* PhD thesis, Technical University of Gdansk, Gdansk (in Polish)