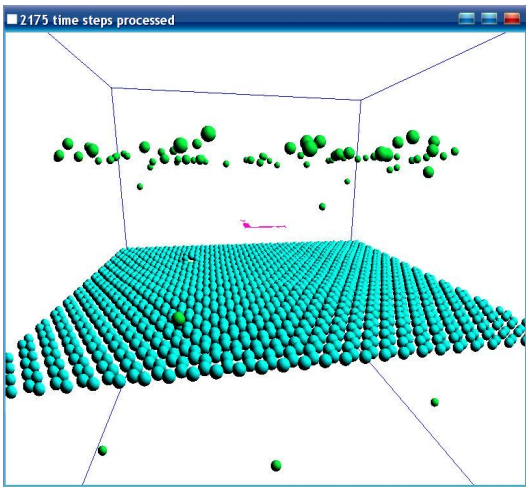
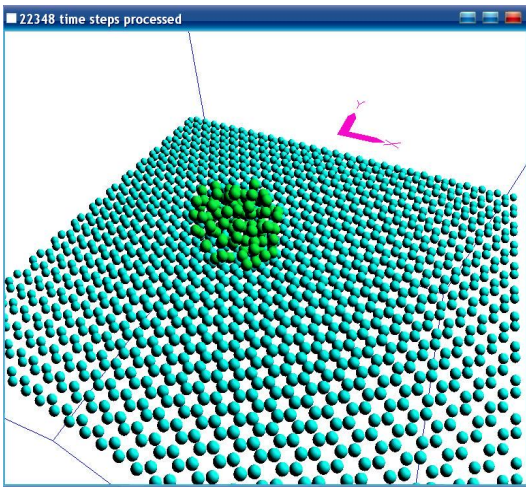
SurfaceGrowth

User guide

Nikolay Prodanov, January 2011





Contents

1 What is this document?

This document is a user guide for the SurfaceGrowth application – the program which is designed for classical molecular dynamics simulations of metal atoms interacting with a graphene sheet. This guide contains the requirements which should be satisfied in order to provide the possibility of compilation and the operation of the application, and also the description of graphical user interface (GUI) of the program. The brief contents of sections of this document is the following:

2 **Requirements** lists the hardware and software requirements.

3 **Compilation** describes some notes about compilation of the code and the settings for the integrated development environment.

4 **Graphical user interface** gives rather detailed description of GUI of the application in different regimes.

5 **Test runs** presents some typical values of the parameters for test runs.

It is recommended to read this document before starting the use of the program, as the latter implies the knowledge of the purpose of the parameters in GUI.

2 Requirements

SurfaceGrowth is a window application designed to operate in the 32 or 64-bit Microsoft (MS) Windows XP (or higher) operating system (OS). It is implemented using MS Visual Studio 2008 integrated development environment (IDE). The following conditions must be obeyed in order to compile and run the program:

- working personal computer (PC) with a motherboard supporting PCI express 2.0 slot. The PC must correspond to the requirements necessary for the operation of the graphics processing unit (GPU), which can be found in the GPU documentation;
- installed NVIDIA CUDA-enabled GPU with Compute Capability 1.2 or higher. See Ref. [1] for the list of such GPUs;
- 32 or 64-bit OS MS Windows XP or higher;
- installed CUDA Driver, CUDA Toolkit (version 2.3 or higher) and NVIDIA GPU Computing SDK (version 3.2 is now available) for 32-bit MS Windows OS [2];
- installed MS Visual Studio 2008 (or 2010) IDE.

Some notes about these requirements should be made. The general configuration of the PC is not critical for the functioning of SurfaceGrowth, so it may be enough to have a commodity x86 processor like Intel Core 2 Duo or AMD Turion, and the RAM of 1 Gb. The presence of the GPU with compute capability of 1.2 or higher is crucial for the work of the application, as this number means the support of atomic functions. For GPUs with smaller version of compute capability the program may not work properly. The version of the OS is not critical, but it is important to install 32-bit version of CUDA Toolkit even if 64-bit OS is used. Visual Studio is required for compilation, but executables compiled in Windows 7 x64 OS are distributed with the code, so it is possible to run the application without the IDE if this OS is used.

To increase the probability of the successful compilation, it may be better to install CUDA Driver and CUDA Toolkit to the default directories.

3 Compilation

MS Visual Studio (VS) 2008 is required in order to compile the executable. MS VS 2010 can be used, but it may irreversibly convert the solution from 2008 to 2010 version. So it may be worth to make a reserve copy of the 2008 version of the solution.

CUDA build rule should be specified in project settings in order to be able to compile GPU code. Most of the important project settings are saved in the project file *<SurfaceGrowth.vcproj>*. Note, that 2 project files *<SurfaceGrowth.vcproj>* and *<SurfaceGrowth x32 3_2.vcproj>* are distributed with SurfaceGrowth solution in order to allow for x64 and x32 OS compilation. The default *<SurfaceGrowth.vcproj>* file is intended for x64 Windows OS and was tested on Windows 7 Ultimate Edition. In order to compile the solution in x32 OS, the default project file should be renamed into e. g. *<SurfaceGrowth x64.vcproj>*, and the second project file should be given the name *<SurfaceGrowth.vcproj>*. After that, settings for x32 OS should be loaded automatically when VS IDE is launched. These files assume that CUDA Toolkit v 3.2 and CUDA SDK v 3.2 are used and have been installed to default directories. To open the solution the user should double click on the file *<SurfaceGrowth.sln>*. If the project file distributed with the code is used, then CUDA build rule should be loaded automatically. If the default project file is not used or the rules cannot be loaded for some reasons, the user should make the following: go to the menu Project → Custom Build Rules (all the solution files must be closed in VS in order to see this menu item). In the dialog box specify "CUDA Runtime Api Build Rule (v3.2)", see Fig. 1.

The user should also specify the include and library directories in the project setting. If the project file distributed with the code is used, then all the necessary paths have been specified automatically. If the project file is not used or the file paths are not specified in it, the user should make the following:

1. Go to Project → SurfaceGrowth Properties.
2. Go to C/C++ → General item.
3. Specify $(CUDA_PATH)/include;$(NVSDKCOMPUTE_ROOT)/C /common/inc;$(NVSDKCOMPUTE_ROOT)/C /common/inc/GL in "Additional Include Directories" field (see Fig. 2). It is very important not to use quotation marks when specifying the paths, as the compiler throws errors if the quotation marks are present.

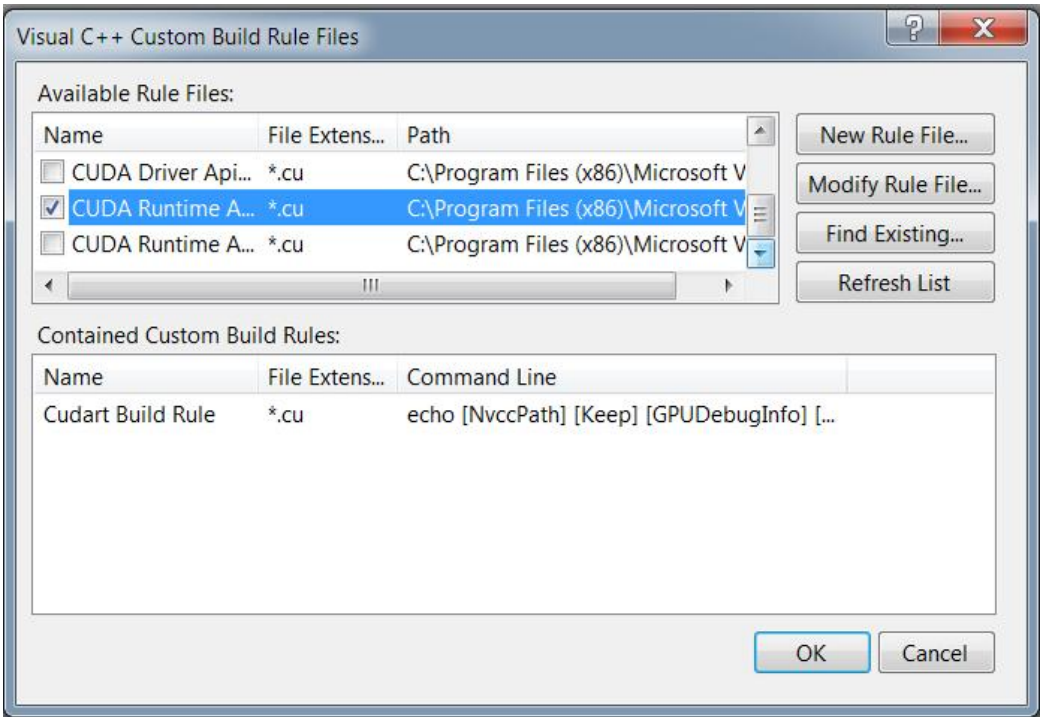


Figure 1: Setting CUDA build rule in MS Visual Studio 2008.

4. Go to Linker → General item.

5. Specify $(CUDA_PATH)/lib/$(PlatformName);$(NVSDKCOMPUTE_ROOT)/C/common/lib in "Additional Library Directories". Ensure, that "" is absent in the settings of the paths.

6. Go to Linker → Input item.

7. Specify cudart.lib cutil32.lib glew32.lib cudpp32.lib in "Additional Dependencies" field.

8. Go to Runtime API → General item.

9. Specify $(NVSDKCOMPUTE_ROOT)/C/common/inc/GL; $(NVSDKCOMPUTE_ROOT)/C/common/inc;$(CUDA_PATH)/include in "Additional Include Directories" field (see Fig. 2). It is very important not to use quotation marks when specifying the paths, as the compiler throws errors if the quotation marks are present.

10. Go to Runtime API → GPU item.

11. Specify sm_12 in GPU Architecture (1) field.

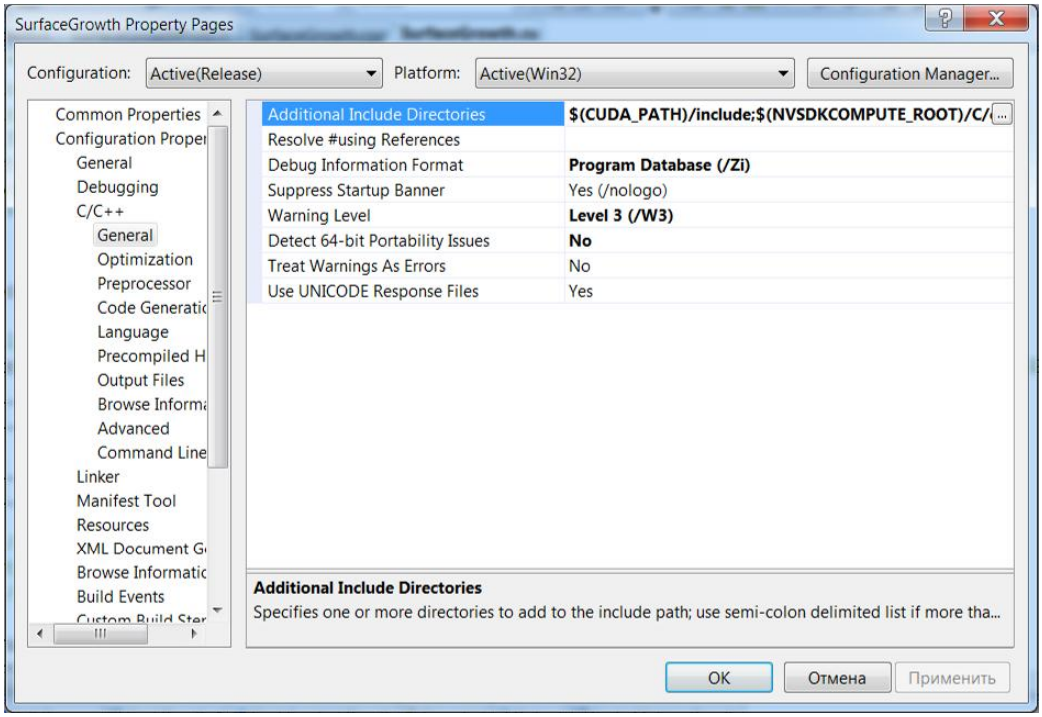


Figure 2: Setting additional include directories in MS Visual Studio 2008.

It should be noted, that all the above settings should be specified in all the configurations of the project (Debug, Release, EmuDebug), if the user wants to compile the application in all the three configurations. To compile the code and to link all the necessary files into an executable (which is called "building" the solution), use menu Build → Build Solution. After the executable has been built, use key F5 for debugging, or Ctrl+F5 to run the application without debugging. Also the user can use executable files *<SurfaceGrowth.exe>* in *<Release>*, *<Debug>* or *<EmuDebug>* directories in order to run the application. Note that executables *<SurfaceGrowth x64.exe>* compiled for Windows 7 x64 are located in the mentioned catalogs and redistributed with the code. They can be executed without compilation of the solution. The presence of dynamic link library files *<cutil32.dll>*, *<glew32.dll>* and *<glut32.dll>* is necessary for running SurfaceGrowth.

4 Graphical user interface (GUI)

Some general peculiarities of the interface of SurfaceGrowth are considered at first. The detailed description of GUI is given after that.

4.1 General description

SurfaceGrowth has a simple GUI implemented as a dialog box using Windows API [3]. GUI allows choosing the regime of the simulation (`Bulk`, `Surface Growth`, `Shear`), metal, and to specify the corresponding parameters. Besides that, it is possible to save the results of measurements in an output file and to choose the possibility of the displaying the system in the OpenGL (OGL) window. However, in this case the simulations are slower in about $3 - 8$ times compared to the regime without OGL window.

In the mode without OGL window there is a possibility to make back up copies of the system, which enable to resume the calculations in the case of their abrupt interruption. To this aim, in the root directory on C disk every 1000 time steps two files *<bckup0.sugr>* and *<bckup1.sugr>* are created intermittently. Two files make the probability to restart the interrupted calculations higher, as they take into account that failure can occur when a back up file is created. Note, however, that even with two files the back up may not work for some, especially large, systems.

Scientific orientation of the program caused the absence of the aspiration for convenient interface, as the most attention was focused on the functionality of calculations. In OGL mode there is the possibility to pause the calculations and to terminate the application in a normal way. Also with OGL window there is the possibility to follow the course of the simulation by the number of the processed time steps, which is output in the OGL window caption. However, OGL is intended only for demonstration purposes as it is very slow, so the mode without OGL window should be used for serious computations. However, in the latter mode there is no possibility for pause and for normal termination of the program before it finishes (the latter depends on the OS), which may be somewhat inconvenient for the user. Pausing can be achieved through the use of backup files. However, as was mentioned above, the back up may not work for some system sizes, so it is not reliable to use such an approach.

4.2 GUI guide

This section describes details of the GUI of SurfaceGrowth program.

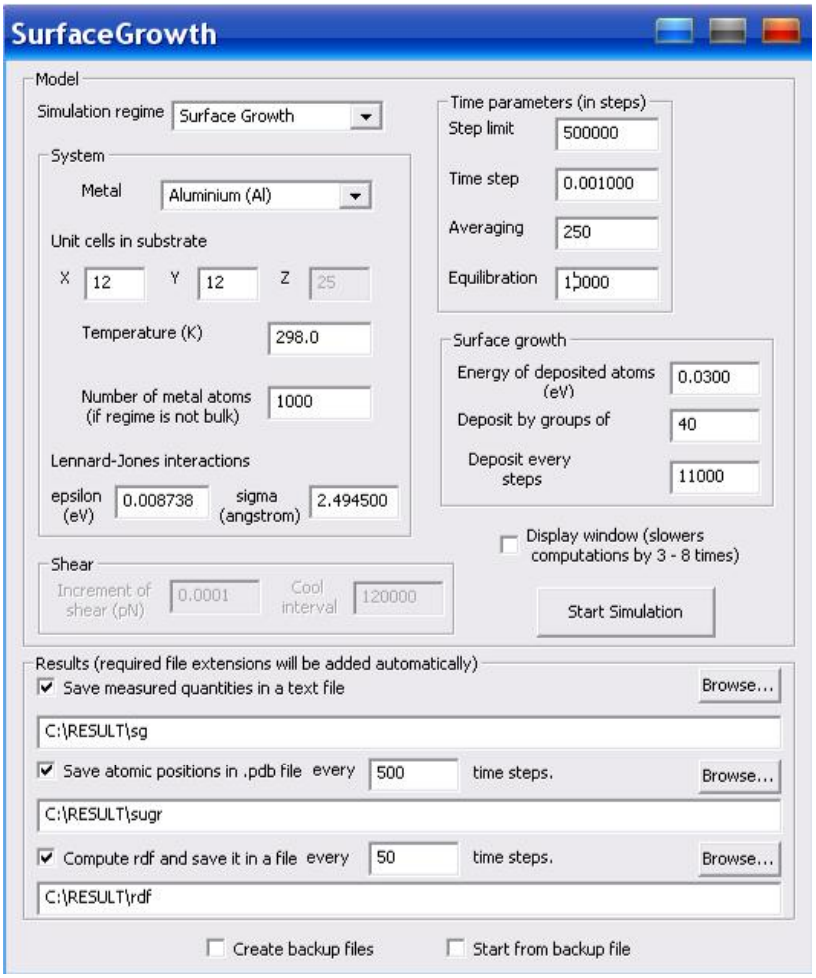


Figure 3: SurfaceGrowth dialog window.

The dialog window shown in Fig. 3 should appear after the launch to the application's executable. This window allows the user to specify the parameters for the simulation. Some parameters are common for all the three simulation regimes, and some parameters are required only in the specific regime. The parameters can be specified only before the computations, as in SurfaceGrowth there is no possibility to interactively change the simulation setup.

Let us consider the parameters which are common for all the three regimes (numbers of the elements in the following list correspond to the numbers in Fig. 4):

- 1 – choice of one of the 3 regimes. Depending on the regime some

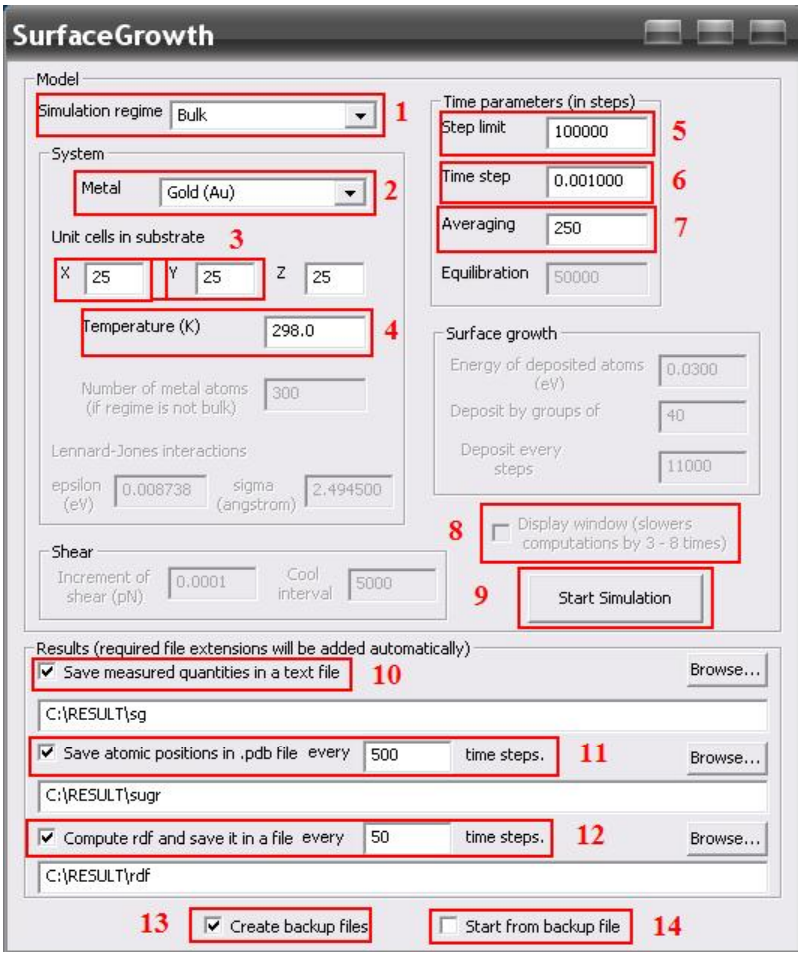


Figure 4: Parameters which are common for all the three simulation regimes.

controls of the window which are necessary for the regime are enabled, and some controls are disabled;

- 2 – choice of one of 6 metals;
- 3 – number of unit cells along the x and y directions. In `Surface Growth` and `Shear` regimes a unit cell used for the translation during the construction of the graphene layer contains 32 atoms (see the code);
- 4 – temperature in Kelvin;
- 5 – the duration of the simulation in time steps;
- 6 – time step dimensionless. Note that although the caption of the group "Time parameters" contains the text "(in steps)", the value of

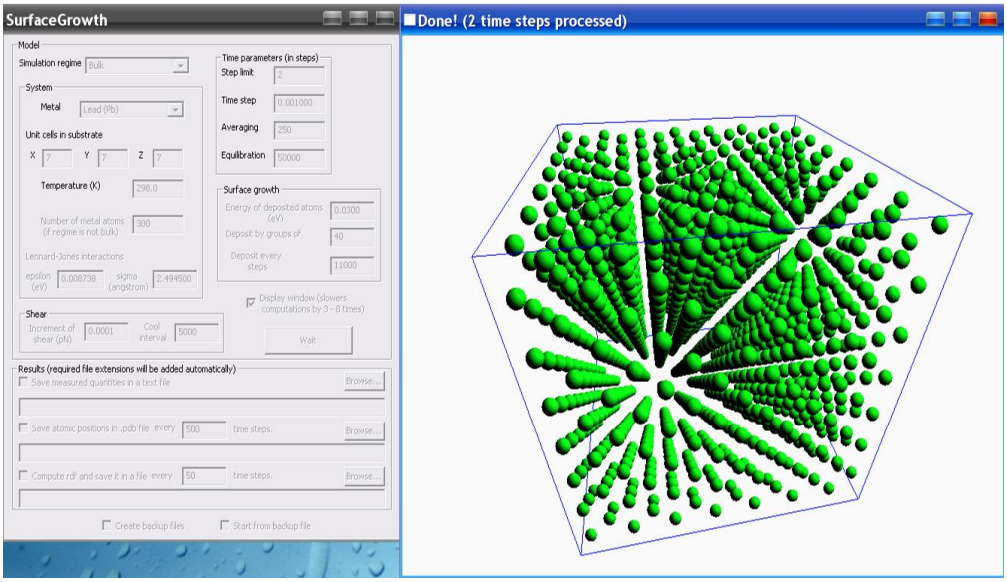


Figure 5: OGL window in `Bulk` regime.

the time step is not in steps. Value of 0.001 dimensionless corresponds to 0.2 fs;

- 7 – averaging interval in time steps. This value defines the time interval for averaging of quantities. Also after every such an interval the results are printed to the output file;

- 8 – flag which defines whether to show the system in the OGL window in real time (see Fig. 5). As has been already noted, this flag leads to the considerable degrade of the performance (up to 8 times), so OGL window should be used only for small systems (this depends on the GPU, may be up to about 10000 atoms) and for demonstration purposes. When the OGL window is used, the system is displayed inside a cube, and the length of its edge is defined by the maximal length of the simulation box in the xy plane. It is possible to perform the following actions in this mode:

  - rotate the system around x and y axes using the mouse with the pressed left button (LB);
  - rotate the system around the z axis using the mouse LB+Alt;
  - shift the system using LB+Shift, or by pressing middle button (MB) of mouse and moving it;
  - scale using Ctrl+LB or LB+MB;

– make pause using the whitespace.

Additionally, it is possible to keep track of the simulation using the window caption of the OGL window, and also to close the application in the usual way by pressing cross in the upper right corner of the window. In this mode back up is not possible;

- 9 – pressing this button causes the beginning of calculations. All the controls of the dialog box are disabled, and the button caption is changed to "Wait". If the OGL window is used, then this caption is not changed at the end of the calculation, and if OGL is not used, then at the end of the simulation the caption of this button becomes "Done! You can exit";

- 10 – flag defining whether to save the results into the output file. If it is set, the user must choose the file where the results will be printed (by pressing Browse, and then select the path and specify the file name). Note, that only the beginning should be specified in the file name . The extension *should not* be specified, as after the start of the calculations values of some quantities will be printed into the file name, and the extension will be added automatically;

- 11 – setting this flag provides the possibility to save atomic coordinates into the PDB (Protein Data Bank [4]) file in order to visualize the system. As in the previous case, if the flag is set it is necessary to choose the file path and specify its name. However, it is recommended to leave the default file name ("sugr_"), and to create directory "RESULT" on the C hard drive for the path, and save files to this directory. It is also necessary to specify the number of time steps after which the file will be saved. This number should not be too small, as printing the coordinates into the file may be rather slow (this number also depends on the chosen duration of the simulation). This flag is recommended to use without OGL window. The reason for the mentioned file name and the path is the following. A file <*vmd.rc*> is distributed with the code. This file defines the behavior of the program Visual Molecular Dynamics (VMD) [5] during its invocation. VMD is designed for the visualization of organic and inorganic molecules, the coordinates of which can be specified in files of different formats, in particular, PDB format. VMD provides wide possibilities for visualization. If the VMD is installed, then there is <*vmd.rc*> file in VMD directory. It is recommended to rename it and to copy to this directory <*vmd.rc*> file,

which is distributed with SurfaceGrowth code. Then, if in the directory "C:\RESULT\" there is the sequence of PDB files obtained using SurfaceGrowth, named <*sugr_0.pdb*>, <*sugr_1.pdb*>, and so on, they will be opened in VMD automatically and the evolution of the system will be visualized;

- 12 – setting this flag provides the possibility to measure the radial distribution function (RDF) and save it in the file. It is required to specify the number of steps after which the measurements will be performed, besides the file name and path. 100 measurements are performed, after that the file is created. So, if a user specified that measurements are made each 50 time steps, then the file will be created every 5000 time steps, reflecting average RDF during 5000 time steps;

- 13 – setting this flag leads to the creation of two back up files (<*C:\bckup0.sugr*>, <*C:\bckup1.sugr*>) during the simulation, and every 1000 time steps all the information necessary for the work of the application is saved to these files. In particular, `SimParams` structure, arrays of positions, velocities and accelerations are saved. If the simulation is interrupted, setting this flag gives the opportunity to resume the calculations. Note that if a simulation comes to the end and the application is closed using the cross in the upper right corner of the dialog box, then mentioned two files will be deleted. If the application did not finish normally, these two files retain on the disk C. Back up is not possible in OGL mode;

- 14 – this flag indicates the desire to resume the interrupted calculation using backup files. Setting this flag leads to disabling all the controls except the flag for the selection of the output file path (10 in Fig. 4) and "Start Simulation" button. If the file path is not selected, but before the interruption the file was created, then output will be saved in the directory from which the application was called. Note that after pressing the button "Start Simulation" dialog box controls *do not reflect* values of the parameters, being used in the simulation. Also note, that back up does not work always, especially for large systems after several interruptions.

Let us consider the parameters, which are specific for a particular regime. These controls are active only when the appropriate regime is chosen.

In `Bulk` regime in fact almost only general input parameters are used. The exception is the number of unit cells along z direction, outlined in Fig. 6.

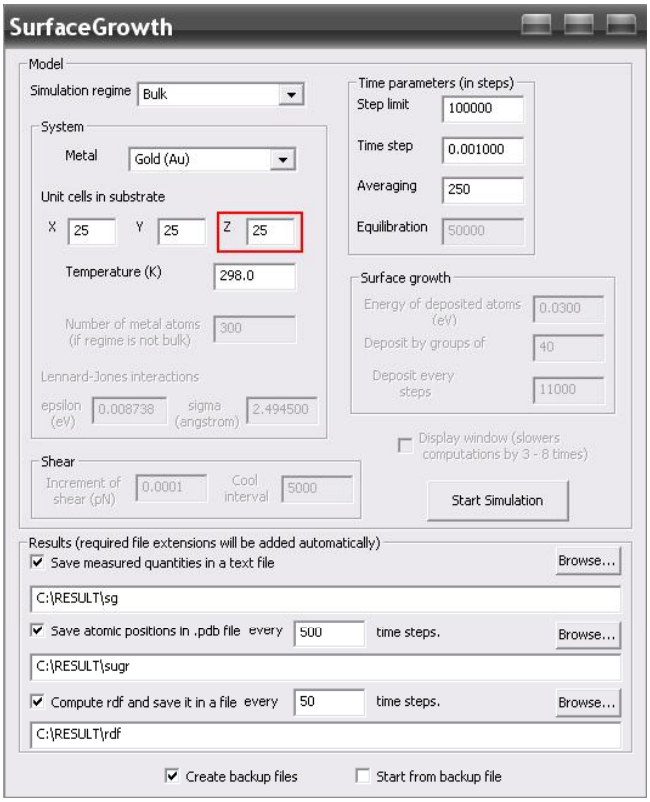


Figure 6: Parameters of the `Bulk` regime.

In `Surface Growth` regime the following 6 specific parameters are added (numbers in the list correspond to the numbers in Fig. 7):

- 1 – sets the equilibration period in time steps. Only graphene layer coupled to the thermostat is in the simulation cell during this time period. This time period is intended for the reaching the desired temperature. Its duration depends on the sizes of graphene layer and should be defined empirically. Note that metal atoms are not deposited immediately after this time interval. The first group is injected after the time, specified by the 6th parameter described below, has elapsed after the equilibration;

- 2 – number of metal atoms. Number of carbon atoms is equal to the product of the number of unit cells along x and y directions multiplied by 32;

- 3 – Lennard-Jones potential parameters responsible for the interactions metal-carbon. The value of `epsilon` should be chosen with taking into

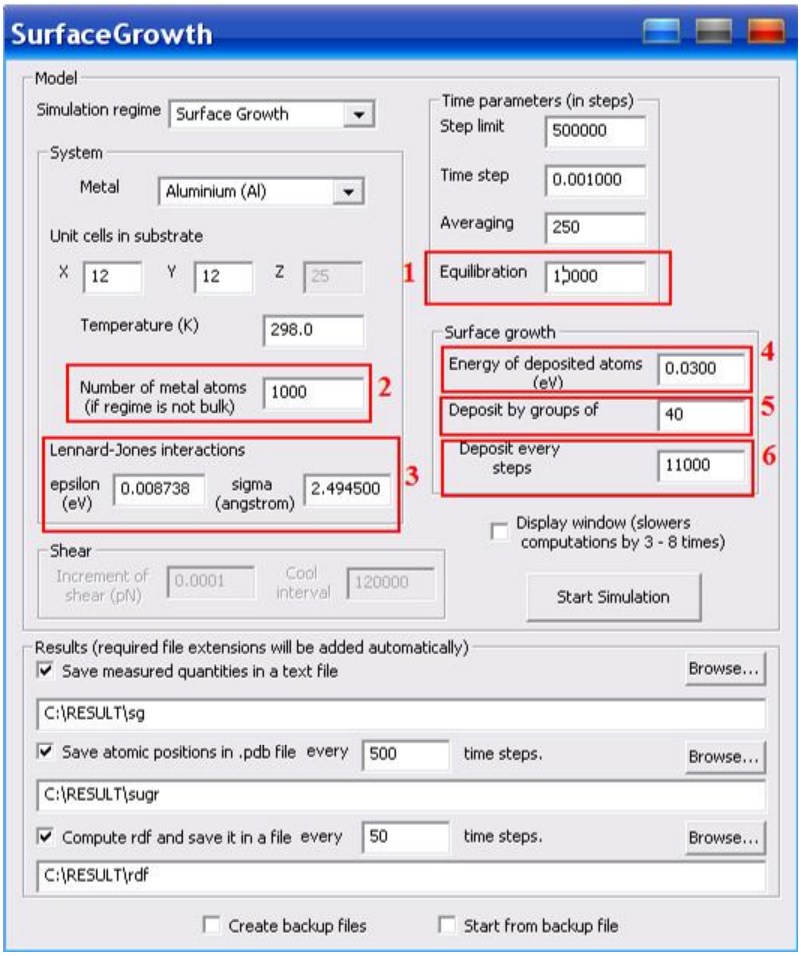


Figure 7: Parameters of the `Surface Growth` regime.

account the energy of the deposited atoms. Too small values may lead to the absence of the deposition, because atoms will be reflected or evaporated from the surface. Too big values will lead to the small mobility of atoms. And if the deposition energy is too large, this may lead to the additional acceleration of metal atoms near the surface and to the ultimate destruction of the graphene layer as a result of collisions with fast metal atoms. In general, the value of `epsilon` should be slightly higher than the deposition energy. The default value of `sigma` can be used in many cases. The ability to change these parameters provides a wide possibility for the investigation of the influence of the value of the interaction energy metal-carbon on the deposition;

- 4 – energy of the deposited atoms, defines the initial velocities of the

atoms. Too high values lead to the destruction of the graphene layer;

- 5 – defines the size of the group of atoms which simultaneously appear in the simulation cell, imitating the deposition. This parameter along with the next one defines the flux of the deposited atoms. This value should be defined empirically, it depends on the total number of atoms, the size of the simulation cell and the next parameter 6. Too high values lead to the destruction of the graphene sheet due to overheating or the collisions with fast metal atoms;

- 6 – time after which the groups of atoms (specified by the previous parameter) appear in the simulation cell. This parameter is defined empirically and depends on the total number of atoms and the parameter 5. If this time period is too small for the current size of the group, then due to the interaction of deposited metal atoms with each other, they will collide with each other, accelerate and destroy the graphene sheet. But too big values will lead to long simulation times. The following estimation can be made. If the number of atoms is 20000, the group size is 50, then the group will be injected into the simulation cell $20000/50 = 400$ times. If the current (6th) parameter is equal to 10000, then the duration of simulation will be equal to $400 \cdot 10000 = 4000000$ time steps. The duration of such a simulation will take more than about 10 hours on GeForce GTX 260.

In `Shear` regime the following 5 specific parameters are added (numbers in the list correspond to the numbers in Fig. 8):

- 1 – equilibration period (in time steps). During this time the system is free, the thermostat is not applied. Metal atoms are rearranged in the configuration which corresponds to the minimal potential energy, the system is heated. Note that in this regime the word "equilibration" does not mean reaching equilibrium state, and it is used for the consistency with the other regimes;

- 2 – number of metal atoms;

- 3 – as in the previous regime, these are the parameters of LJ interaction between metal and carbon atoms.

- 4 – this quantity defines how fast the shearing force acting on the atoms of the nanoparticle is increased. As is mentioned in the article, shear is imitated by applying the shearing force along the x axis to all the metal atoms which are located to the left from the center of mass of

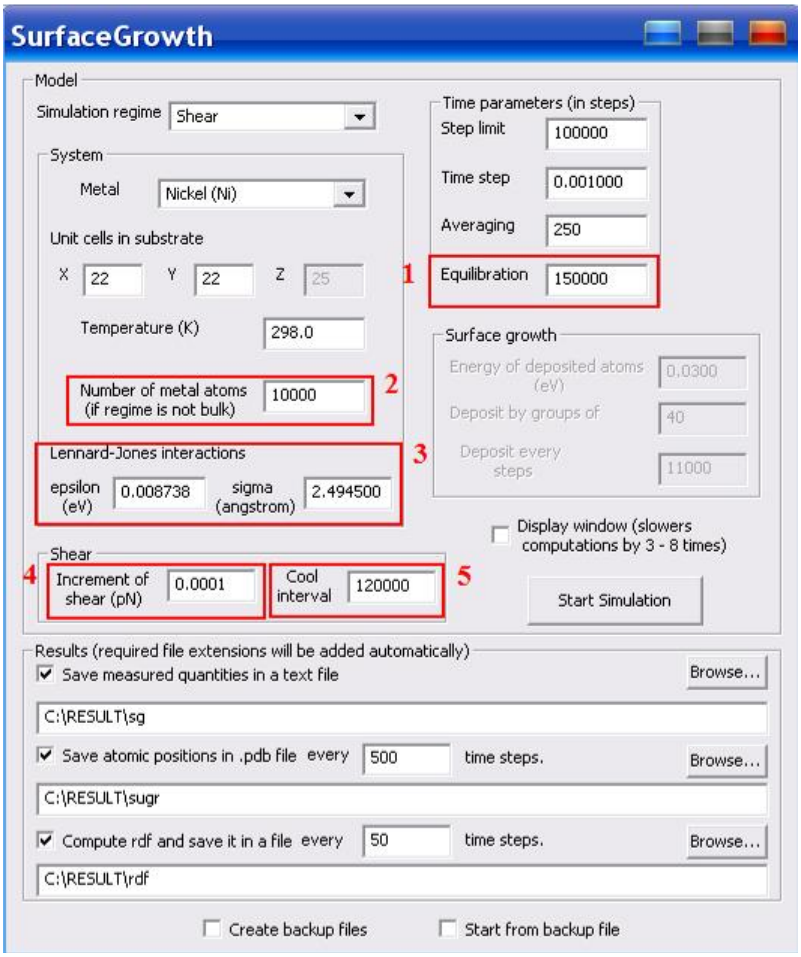


Figure 8: Parameters of the `Shear` regime.

the nanoparticle. As it is not known in advance what value of the force is required for shearing the nanoparticle (static friction force), in SurfaceGrowth the following approach is employed. The shearing force is increased step by step until the velocity of the center of mass of the nanoparticle reaches some predefined value (which is chosen to be 0.005 dimensionless, or 3.55 m/s). The current parameter 4 defines the value of the step, which is used for the increasing the force;

- 5 – the duration of the cooling of the nanoparticle. It defines the time interval during which the Berendsen thermostat is applied both to the metal and carbon atoms in order to cool the system down to the specified temperature.

5 Test runs

The values of parameters which can be used for testing purposes are presented in this section. It is recommended to perform testing on a desktop, but not on a mobile GPU, as there may be some problems with running the program in OGL mode on some mobile graphical cards (for example, the problems were observed on GeForce GT 320M). The program has been tested on MS Windows XP 32-bit Professional Edition and Windows 7 x64 Ultimate Edition. The release executable file distributed with the code is designed for the latter. When OGL window is not used, the application should work without problems on any GPU, including mobile models. The following configurations of the parameters may be used for testing in the mode without OGL window:

- `Bulk` regime. Metal: Copper (Cu); Unit cells in X: 30, Y: 30, Z: 30; Temperature: 298; Step limit: 100000; Time step: 0.001; Averaging: 250. Also check all the 3 check boxes, the default parameters for PDB files and RDF may be used. On GeForce GTX 260 it takes about 41 ms to calculate one time step for this system. Note that memory bandwidth is the most critical parameter of a GPU for the SurfaceGrowth. Thus, it runs faster on GeForce GTX 260 than on some new GPU such as some GeForce GTX 460 models, which have smaller memory bandwidth.

- `Surface Growth` regime. Metal: Aluminium (Al); Unit cells in X: 11, Y: 11; Temperature: 298; Number of metal atoms: 2000; epsilon: 0.05; sigma = 2.494500; Step limit: 800000; Time step: 0.001; Averaging: 250; Equilibration: 10000; Energy of deposited atoms: 0.03; Deposit by groups of: 20; Deposit every: 5000. Also check the first 2 check boxes, use 1000 steps for generating PDB files.

- `Shear` regime. Metal: Nickel (Ni); Unit cells in X: 23, Y: 23; Temperature: 298; Number of metal atoms: 13000; epsilon: 0.008738; sigma = 2.494500; Step limit: 600000; Time step: 0.001; Averaging: 250; Equilibration: 115000; Increment of shear: 0.0001; Cool interval: 150000. Also check all the 3 check boxes, the default parameters for PDB files and RDF may be used.

For the second set of the parameters the output is distributed with the user guide. However, not all PDB and RDF files produced in the course of the run are attached, but only 25 % of them, due to space limitations. When OGL is used, the smaller systems should be considered. Some of them are

depicted on the title page of this guide. It is recommended for the user to try different combinations of parameters for testing the application.

References

[1] NVIDIA CUDA webpage, http://www.nvidia.com/object/cuda_home_new.html.

[2] http://developer.nvidia.com/object/cuda_3_2_toolkit_rc.html#Windows XP, Windows Vista and Windows 7.

[3] C. Petzold, Programming Windows, 5th ed. (Microsoft Press, 1998).

[4] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, P. Bourne, Nucleic Acids Research 28 (2000) 235 http://www.rcsb.org/pdb.

[5] W. Humphrey, A. Dalke, K. Schulten, J. Molec. Graphics 14 (1996) 33 (http://www.ks.uiuc.edu/Research/vmd/).