A New Dynamical Domain Decomposition Method for Parallel Molecular Dynamics Simulation

V. Zhakhovskii1, K. Nishihara1, Y. Fukuda1, S. Shimojo2, T. Akiyama2, S. Miyanaga2, H. Sone3, H. Kobayashi3, E. Ito3, Y. Seo4, M. Tamura4, and Y. Ueshima5

1Institute of Laser Engineering, Osaka University, Yamada-oka 2-6, Suita, Osaka, 565-0871, Japan
basil@ile.osaka-u.ac.jp
2Cybermedia Center, Osaka University, Japan
3Information Synergy Center, Tohoku University, Japan
4NEC Corporation, Japan
5Advanced Photon Research Center, JAERI, Japan

Abstract
A new material particle dynamical domain decomposition method MPD3 has been developed. The method is suitable for a large scale parallel molecular dynamic simulation on a heterogeneous computing net. Performance of the MPD3 algorithm is tested in various computing environments, such as PC clusters, super computer clusters, and Grid. It is shown that the MPD3 algorithm is highly adaptive for both computer clusters and Grid computing environments, even if other programs are running on the same computer environment.

1. Introduction
Classical molecular dynamics (MD) approach is widely applied for simulation of complex many-particle systems in many areas of physics, chemistry and biochemistry. The particles such as ions, atoms and molecules interact with each other through a given force function. By step-by-step integration of Newton’s equations of motion, trajectories of all particles are obtained. Useful information can be extracted from the particle trajectories by using a suitable averaging procedure.

Typically the interactions among the particles can be described by short-range forces, hence MD simulation has local spatial character. Therefore the particle motion for a short period can be determined only from particles located in neighborhood of a given particle. This fact was used to develop successful parallel MD algorithms based on a static spatial domain decomposition (SDD) of simulation area [1]. There are a few approaches to improve the load balancing of SDD [2, 3], which works well for simulations with uniform density distribution and without significant flows of particles.

A large scale MD simulation requires a large number of high performance computers. Recently the Grid computing has been developed and allows us to use many computers connected through a net. However performance of the computers and the net may not be the same and their performance may not be known before the use. In addition, other users may also run their programs on the same computer network without notice. If we use many computers located in different sites, some of the computers may be very busy for certain times because of other users. A dynamical domain decomposition method is therefore required to obtain a good adaptive load balancing for heterogeneous computing environments such as Grid. We have developed a new Material Particle – Dynamical Domain Decomposition method (MPD3) for large scale parallel MD simulations on a heterogeneous computing net, which results in large reduction of elapsed time.

Recently the MPD3 method was applied for simulations of hydrodynamics problems like the Richtmyer-Meshkov instability [4] and laser destruction of solids [6]. The flows of matter in both cases result in great imbalance among processors (CPUs) in SDD. The new method is also applicable for simulations of any dynamical processes with strongly nonuniform density distribution accompanied with phase transition, shock waves and cracks.

In this report we present a new MPD3 algorithm and its performance tested on real physical problems and in various computing environments, such as PC clusters consisting of different power CPU connected by low bandwidth net-
work, cluster supercomputers, and vector supercomputers connected by Grid network with Globus toolkit 2.4.

2. Adaptive MPD$^3$ Algorithm

MD simulation is carried out using many processors connected through nets, where a number of the processors is $N_p$. Performance of the processors and the nets may not be the same. Each processor calculates MD motions of the atomic particles, like atoms and molecules (hereinafter called atoms) belonging to a subdomain. We consider following computing environment: other user's programs and system programs are running on some of the $N_p$ processors and a number of other programs may change during the simulation. Our main goal is to develop a successful algorithm to accomplish a good load balance among the processors to reduce elapsed time for main loop of simulation. In other word we adopt the following definition of the well-balanced simulation. A pair of processors is well-balanced during main loop of algorithm if they are fully utilized with the minimum of communications. It should be noted that the computational loading of each processor may also change in time due to the dynamical processes (like compression wave, bubble formation and etc) in the subdomains. Therefore the load balance algorithm has to be iteratively adapted for time-dependent computing environment and physical dynamics of simulated phenomena.

As a simple example, Fig. 1 (left) shows possible two dimensional (2D) decompositions of a uniform density cylindrical body into 4 subdomains among 4 CPUs, two fast and two slow processors. The initial not well-balanced decompositions (left images) are transformed into well-balanced (right images) by the MPD$^3$ method after a few iteration steps.

Let us demonstrate another example how the algorithm should work to reach the well-balanced condition. Assume that our MD simulation of one dimensional uniform density rod (with unit length $l = 1$) consisting of one sort of atoms is loaded on two CPUs with equal processor power/speed ($P_1 = P_2 = 1$). It is clear that well-balanced decomposition of the rod is the simple division into equal parts, $l_1 = l_2 = l/2 = 1/2$. But if one of these CPUs is also loaded by other program, then available CPU power $P$ for our simulation is reduced. One may take for instance $P_2 = 1/2$, which means that only $P_2 t_2 = 1/2 t_2$ of the 2nd processor time can be utilized for our program. To set up a good balance between two CPU the algorithm has to redistribute atoms among them. The well-balanced decomposition fits the equal elapsed times for a simulation loop $t_1 = t_2$, that are proportional to parts of rod $t_1 \sim l_1/P_1$ and $t_2 \sim l_2/P_2$ consequently. Hence the well-balanced condition is $l_1/P_1 = l_2/P_2$, and finally $l_1 = P_1/(P_1 + P_2) = 2/3$ of the rod should belong to the 1st CPU and $l_2 = P_2/(P_1 + P_2) = 1/3$ to the 2nd CPU. The same algorithm can be applied to different power CPUs as well as arbitrary activity of other users.

![Fig. 1. Two dimensional dynamical domain decompositions for simulation of a crystal cylinder at rest. Each subdomain belongs to a processor. Two CPUs are faster than other 2 CPUs. The well-balanced decompositions are displayed on the right.](image)

Now we derive the MPD$^3$ method in a general form. Assume that main simulation loop/step in the algorithm can be divided into two major parts – calculation of atom motion without inter-processor communication (MD integration part) and exchange of atom data among the processors (communication part). A program must estimate both the processor dependent time (cpu.time) and the elapsed time (sys.clock) for the whole main loop as well as for the integration part of the loop. Here the cpu.time is a time duration spent only for our program in a processor. If other programs run on some of the processors, the elapsed times become longer than the cpu.times in those processors.

Available processor power $P(i)$ can be determined through the normalized MD integration time of the $i$-th processor as

$$P(i) = \frac{t_{wMD}(i)}{t_{eMD}(i)}, \quad 0 < P(i) \leq 1 \quad (1)$$

where $t_{wMD}$ is the cpu.time spent for the integration without communication with other processors, and $t_{eMD}$ is the sys.clock time for the integration. The $P(i)$ can be used for the estimation how the $i$-th processor is loaded by other users/system jobs under the assumption that activity of other programs is almost constant during the main simulation loop. If there are no other programs, $P(i)$ equals to one.

Let us define the weighting factor $W(i)$ of the $i$-th
CPU as
\[ W(i) = \frac{t_w(i)}{P(i)t_e(i)}, \quad 0 < W(i) \leq 1 \quad (2) \]

Here \( t_w \) is the cpu time of the main loop, and \( t_e \) is the elapsed time that includes the total duration for the MD integration and the communication. It is reasonable to suppose that the \( i \)-th CPU is in a good balance with the \( j \)-th CPU, if their weighting factors almost equal each other: \( W(i) \cong W(j) \). In the special case that the evaluation of the processor dependent time is impossible, one may use the simplified definition of the weighting factor, \( W(i) = t_{eMD}(i)/t_e(i) \) instead of Eq. (2).

A simulation domain can be divided into \( N_p \) simple subdomains like rectangular boxes before the MD simulation. As an example of initial domain decomposition, Fig. 1(left) shows possible decomposition of a cylinder. Each of 4 CPUs simulates its sub-domain. \( N(i) \) is a number of atoms in the \( i \)-th subdomain. We define the position of the center \( \mathbf{R}(i) \) of the \( i \)-th subdomain as following:

\[ \mathbf{R}(i) = \frac{1}{N(i)} \sum_{k=1,N(i)} \mathbf{r}(k), \quad (3) \]

where \( k \) denotes an atom number in the \( i \)-th subdomain, \( \mathbf{r}(k) \) is a position of the \( k \)-th atom. For identical atoms, \( \mathbf{R}(i) \) is a center of mass of the \( i \)-th subdomain. For each pair of the \( i,j \)-subdomains, we define a boundary plane between them at the midpoint \( \mathbf{R}_{12}(i,j) = (\mathbf{R}(i) + \mathbf{R}(j))/2 \) and perpendicular to the connecting vector \( \mathbf{R}(i,j) = \mathbf{R}(i) - \mathbf{R}(j) \). After that all of the atoms near the boundary are associated with one of the \( i,j \)-subdomains. By repeating this procedure for all pair subdomains, the simulation domain will be finally divided into \( N_p \) Voronoi polygons, see Fig. 2. The map of Voronoi polygons is known as the Dirichlet tessellation and used for grid generation in computational fluid dynamics [7].

For a system with a uniform atom density, the area of each Voronoi polygon is the same and thus the number of atoms becomes the same in each domain. Therefore the static Voronoi decomposition gives a good balance only for a uniform density distribution with a homogeneous computing net. Nevertheless the Voronoi decomposition is a good point to start MD simulation. If the atom diffusion and exchange between the polygons are forbidden, the dynamical behavior of these polygons looks as motion of Lagrangian particles from the hydrodynamics point of view. This is the reason why we term the moving Voronoi polygon as a material particle (MP).

To obtain a good load balance in the heterogeneous computing environments the MP centers should be adjust time-dependently so that for example a busy computer calculates less number of atoms compared with others. We propose the following simple and efficient iterative algorithm to calculate the displacement of the MP center. The displacement of the \( i \)-th MP center at the current \( n \) simulation step can be evaluated as

\[ \Delta \mathbf{R}(i) = \sum_{j=1}^{N_n(i)} \frac{a L(i,j)}{\sqrt{N_n(i)}} [W(i) - W(j)] \frac{\mathbf{R}(i,j)}{|\mathbf{R}(i,j)|}, \quad (4) \]

where \( N_n(i) \) is a number of neighbor MPs surrounding the \( i \)-th MP, a suitable symmetrical function \( L(i,j) \) with length dimension, and \( a \) is an adjustable dimensionless parameter of the method. We use \( L(i,j) = \min[L(i),L(j)] \), where \( L(i) \) is a linear size of the \( i \)-th MP. The possibility exists of using other forms of Eq.(4). It is clear from Fig. 2 that the pair displacement \( \Delta \mathbf{R}(i,j) \sim |W(i) - W(j)| \) affects not only pair \( (i,j) \) imbalance but all neighbors of \( i \)-th MP as well. Therefore any linear combination of independent pair displacements can not be optimal formula. Nevertheless Eq. (4) is enough efficient from a computational standpoint as it will be shown below.

At the next simulation step \( (n+1) \) a new desired position of the \( i \)-th MP is given by

\[ \mathbf{R}(i)^{n+1} = \mathbf{R}(i)^n + \Delta \mathbf{R}(i). \quad (5) \]

Once the exchange of atoms among MPs according to these desired centers has been done, the true center of the \( i \)-th MP, \( \mathbf{R}(i)^{n+1} \) is evaluated from Eq. (3). It is hoped that the differences of the weighting factor \( |W(i) - W(j)| \) in Eq. (4) may be minimized by repeating of this process at each step.

Because of iterative Eq. (5) a good load balance may not be reached within a few steps. In addition, the well-balanced
Voronoi decomposition may not exist for small numbers of CPUs \( N_p \). Nevertheless, as will be shown later in our simulations with processor number of \( N_p \) from 14 to 720, the good-balanced decomposition can be achieved within 20\% of imbalance \( |W(i) - W(j)| \) between the fastest CPU and the slowest one.

Finally the simplified MPD\(^3\) algorithm can be designed as follows:

0) **Initialization.** Any simple initial decomposition.

1) Exchange atoms between neighbor MPs according to the Voronoi method.

2) Evaluate the new MP positions and iterate 1)-step so long as every MPs reach steady shapes.

3) **Start simulation.** First exchange the MP positions Eq. (3) and timing data among the neighbor MPs. Then evaluate a new desired position of the MP center by using Eqs.(4) and (5). Do exchange atoms and etc.

4) Advance MD integration step and measure all timing data: \( t_{\text{wMD}}(i), t_{\text{cMD}}(i), t_w(i), t_c(i) \).

5) **Repeat** 3)- and 4)-steps.

It should be pointed out that the new MPD\(^3\) method shares features with the usual static spatial decomposition method as well as the particle decomposition algorithm [1]. In other words the simulated atomic particles are distributed among CPUs according to their positions in the dynamical clustered medium of subdomains/MPs that depend on atom motion. An atom keeps its number and membership of MP only for a relatively short period. To prevent the increase of cache memory missing the program renumbers atoms inside MP according to its position in a neighbor atom list. It has been observed in our MD simulations that the renumbered neighbor atoms locate within short distances from each other in computer memory on average. The renumbering significantly improves cache hitting and CPU performance.

It is also important to keep in mind that the distances and forces between pair of neighbor atoms are calculated once if they belong to the same MP subdomain and twice in either CPU if the atoms are located in the vicinity of the boundary of two different MPs as it is shown by double lines connecting the atomic particles in Fig. 2.

![Fig. 3. The normalized performance (scaled efficiency) as a function of number of processors involved to the test. Triangles denotes of 80 CPU PC cluster. Cycles correspond to 720 CPU super computer cluster. Solid line indicates the perfect scalability.](image)

![Fig. 4. Snapshots of formation of central and peripheral cracks taken from [6]. On top the initial 1D decomposition among 8 identical SX-5 CPUs. On bottom the final structure of MPD\(^3\). The load balance was perfect during all simulation.](image)

3. **Testing**

Let us consider an ensemble of many atoms interacting each other with short-range pair forces in a MD simulation box that is divided into subdomains. The pair potential for our model system is described by the quasi-Lennard-Jones (qLJ) pair potential with cut-off distance \( r_c^2 = 5r_0^2 \), where \( r_0^2 = 2 \). Here and after we use standard reduced MD units where LJ parameters \( \sigma = 1 \) is unit of length and \( \epsilon = 1 \) is energy unit. In reduced MD units, the qLJ potential is given by

\[
\phi_{qLJ}(r) = \frac{r_0^2}{25} (r^2 - r_c^2)^4 \left[r^{-10} - r^{-4}\right]
\]

Unlike to the well-known LJ potential function truncated at cut-off radius \( r = r_c \) the qLJ potential Eq. (6) is enough smooth function at \( r_c \approx 2.51 \), resulting in better energy conservation. The atom mass is assumed to be \( m = 48 \) MD units. The parameter in Eq. (4) is set to \( a = 0.5 \) for all benchmark simulations. To integrate equation of motion we use the time-reversible explicit integrator on the basis of the second order Taylor expansion of force [5].

The MPD\(^3\) method was implemented in the fully vectorized Fortran program with calls of standard MPI 1.1
Fig. 5. Snapshots of the collision of two cylinders simulated in the Grid environment on NEC SX-5 (Osaka University) and NEC SX-7 (Tohoku University) machines connected with Super-SINET. 1. Simple initial rectangular decomposition among SX-5 machine (left cylinder, 7 MPs) and SX-7 machine (right cylinder, 7 MPs). 2. Start point for MD simulation. Steady Voronoi decomposition was achieved after 14 iterations. Two MPs move from right cylinder to the left as indicated by thick line which shows the boundary between two machines. 3. Both computers work in exclusive usage mode which corresponds to the (1)st period in Fig. 6. MP subdomains are almost equal in size. 4. SX-5 machine was overloaded by other tasks/users but SX-7 was almost free. Picture is captured from the end of the (2)nd period shown in Fig. 6.

Fig. 6. Performance of the MPD$^3$ method for simulation of two bodies collision (see Fig. 5) Waiting time indicates time duration for communication time per a simulation step. (1). Both computers work in exclusive usage mode corresponding to Fig. 5.2-3. Long waiting time is due to the low bandwidth of the network line between two computers. (2). Continue of simulation in the Grid on two nodes of overloaded SX-5 and SX-7. The initial long waiting time was reduced by auto-balancing algorithm, see snapshot in Fig. 5.4.

Subroutines. Scalability of the MPD$^3$ program was measured on two different clusters. The first was a PC cluster consisting of 40 nodes connected 100 Mbps line, with LAM/MPI environment. Each node has Dual Pentium III 1.4 GHz. The benchmark was performed for simulation of a qLJ crystal cylinder consisting of 2, 551, 472 atoms. The initial atom density was $\rho = 1.035$ and temperature $T = 0.25$ in reduced MD units. We measured performance as $S = N_{\rho 0} \tau(N_{\rho 0}) / \tau(N_{\rho})$, where $\tau$ is the elapse time per simulation step. The smallest CPU number $N_{\rho 0}$ used for the normalization can not be 1 for a large MD system, because of the limitation of a local CPU memory. Figure 3 shows that the scaled efficiency of the small PC cluster is close to the ideal line. The second cluster was super computer cluster Compaq AlphaServer ES40-4 SC (180 nodes, 720 CPU) in Super Simulation Center at Advanced Photon Research Center (JAERI). The simulated system was extended to 22,961,888 atoms. Figure 3 also shows very good performance except small deviation from the perfect linearity for $N > 160$, which may be due to two reasons. At the fixed number of atomic particles the MP size decreases with the increase of the total number of CPUs, therefore the number of atoms in a vicinity of inter-MP boundary becomes relatively bigger. For these atoms the pair interaction is calculated twice in two neighbor MPs independently as it is pointed out in the end of previous section. It may result in relative reduction of the CPU efficiency for very small MP subdomain. In addition the MPs have to exchange information of boundary atoms at each simulation step, hence the communication cost increases considerably when the number of boundary atoms becomes comparable with the number of bulk atoms (which do not interact with atoms located in the neighbor MPs). The similar drop of the performance is also visible for the PC cluster near 80 CPU (see triangles in Fig. 3)

The MPD$^3$ program was also tested in various computing environments and physical problems. It was applied to MD simulation of one dimensional expansion of a laser-heated crystal consisting from $\sim 10^7$ atoms by using 8 identical CPUs on NEC SX-5 of Cybermedia Center, at Osaka University [6]. As it is shown in Fig. 4 the laser-heated crystal expands and cracks appears to destroy the continuous matter. Phase change in the crystal and the cracks may cause heavy imbalance of CPU loading, if the conventional SDD is applied. The MPD$^3$ method has shown the almost perfect load balance in 1D case due to adaptive redistribu-
tion of atoms among processors.

Fig. 7. Snapshots of two cylinders collision history, simulated by PC cluster (80 CPU). Snapshots of collision simulated by Compaq super computer (716 CPU) at JAERI look similar. 1. Decomposition at the starting point of MD simulation. 2. At near collision moment the redistribution of MP sizes starts to restore good load balance, see (2)nd stage in Fig. 8. 3. MP subdomains become almost equal in size (except high dense contact area) at the end of the (3)rd period shown in Fig. 8.

We measured performance of the MPD\(^3\) method in details for another two testing problems which demonstrated applicability of the new method to computer clusters and Grid computing. Both tests deal with MD simulations of high-speed collision of two solid cylinders with different radii. It is clear that the static domain decomposition may have a hard load imbalance because of large flow velocity of the atoms, and as we expect, the dynamical decomposition by the MPD\(^3\) algorithm will be sufficiently effective to establish a good load balance.

In the first test we use two super computers, NEC SX-5 of Cybermedia Center at Osaka University and SX-7 of Information Synergy Center at Tohoku University. They are connected with Super SINET network with the use of Globus toolkit 2.4.

Figure 5 shows a set of snapshots from the beginning to the end of the simulation. There are two crystal cylinders consisting of 2,066,700 LJ atoms in all. Initially the simplest rectangular decomposition is chosen for dividing simulation domain among 7 SX-5 CPUs and 7 SX-7 CPUs as shown in Fig. 5.1. After 14 iterations, the steady Voronoi decomposition is established. At the moment the left cylinder consists of 9 MPs, of which 7 MPs belong to SX-5 and other 2 MP to SX-7, and the right cylinder consists of 5 MPs belonging to SX-7, see Fig. 5.2. This is the starting point of the MD simulation of atom motion which corresponds to the end of step 2) of Algorithm in the previous section. After that the timing data of each CPU are measured.

According to the timing data the MPD\(^3\) algorithm starts to establish a good load balance among CPUs. In Fig. 5, the thick black line denotes the boundary between SX-5 and SX-7. The snapshot in Fig. 5.3 shows more or less uniform distribution in MP sizes at the end of the first period corresponding to (1) in Fig. 6. It reflects that the simulation is carried out in exclusive usage mode on both computers. The waiting time is near \(\sim 100\%\) of the CPU dependent time/step because the low bandwidth line connects the ma-
The next snapshot in Fig. 5 shows the adaptive deformation of the domains due to the changes of the load condition on two machines. At this time SX-5 at Osaka University is overloaded by other users/tasks while SX-7 at Tohoku University is much less loaded. As indicated in the Fig. 5.4, it results in reduction of the areas corresponding to the MPs belonging to SX-5. The adaptation from Fig. 5.3 to Fig. 5.4 states takes about 500 simulation steps as is shown in the beginning of the (2)nd period of Fig. 6. It should be stressed that MPs redistribute atoms not at each simulation step but every 16th steps.

In the second test we run simulation on cluster supercomputer Compaq AlphaServer ES40-4 SC (716 CPU was used). There are two crystal cylinders consisting of 18,599,160 qLJ atoms in all. Figure 7 shows three images of simulated domain from the beginning to the end of simulation. Initially the random decompositions are chosen independently for each cylinder. The left cylinder is partitioned into 365 MP's and the right cylinder – into 357 MP's. After 60 iterations, the steady Voronoi decomposition is established, as shown in Fig. 7.1. As it has been already pointed out this is the starting point of the MD simulation of the collision dynamics.

As can be seen from the (1)st period in Fig. 8 the well-balanced decomposition is established in the beginning despite the fact that average size of MP in small cylinder is 4 times smaller than MP size in large one. This is due to the fact that two cylinders can move independently until collision, almost without communication, due to long distance between them, see Fig. 7.1. In this period the waiting time is nearly 10% of the elapsed time per a simulation step.

When two cylinder starts to interact with each other, the communication between the two cylinders begins to exchange atoms and atom positions. It results in a jump of elapsed time. Within the (2)nd period in Fig. 8, the MPD³ algorithm recovers the load balance by redistribution of atoms among CPUs. It causes significant flow of MPs from the small cylinder to large one as shown in Fig. 7.2.

The (3)rd period in Fig. 8 indicates that the well-balanced decomposition is established again. Correspondingly Fig. 7.3. shows that the MP subdomain sizes become almost equal, except the area where density and temperature are significantly higher than their initial values due to the collision.

4. Conclusion

We have demonstrated that the MPD³ method is a highly adaptive dynamic domain decomposition algorithm for MD simulation on both computer clusters and Grid computing environments, even if other programs/tasks are running on the same environment.

It has been shown that the well-balanced decomposition results from dynamical Voronoi polygon tessellation, where its center is displaced to reach the minimum elapsed time. Our approach can be extended to other particle methods like Monte Carlo, particle-in-cell, and smooth-particle hydrodynamics if the cut-off radius of particle interaction is much smaller than simulation domain.

The MPD³ method works perfectly for 1D decomposition, but for 2D case the load balance may depend on a geometrical configuration of simulation problems and a number of CPUs in use. We have demonstrated that the MPD³ is especially optimal for large scale multi-CPU simulations.

We are grateful to the Cybermedia Center at Osaka University, Information Synergy Center at Tohoku University, and Super Simulation Center at Advanced Photon Research Center (JAERI) for organization of computer experiments.

5. References

[1] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics", J. Comput. Phys., Academic Press, 1995, vol. 117, pp. 1-19.

[2] L. Nyland et al, "Achieving Scalable Parallel Molecular Dynamics Using Dynamic Spatial Domain Decomposition Techniques", J. Parallel and Distributed Computing, Academic Press, 1997, vol. 47, pp. 125-138.

[3] Y. Deng, R. F. Peierls, and C. Rivera, "An Adaptive Load Balancing Method for Parallel Molecular Dynamics Simulations", J. Comput. Phys., Academic Press, 2000, vol. 161, pp. 250-263.

[4] V. Zhakhovskii, K. Nishihara, and M. Abe, "Molecular Dynamics Simulation on Stability of Converging Shocks", Inertial Fusion Science and Applications, IFSA 2001, Elsevier, Paris, 2002, pp. 106-110.

[5] V. Zhakhovskii, A New Time-Reversible Integrator for Molecular Dynamics Applications, ILE Annual Progress Report 2003,(Osaka Univ., 2004), pp.151-154 (see also arXiv:physics/0405124, 24 May 2004)

[6] S. I. Anisimov, V. V. Zhakhovskii, N. A. Inogamov, K. Nishihara, A. M. Oparin, and Yu. V. Petrov, "Destruction of a Solid Film under the Action of Ultrashort Laser Pulse", JETP Lett., Nauka/Interperiodica, Moscow, 2003, vol. 77, pp. 606-610.

[7] T. J. Chung, Computational Fluid Dynamics, Cambridge University Press, 2002, ch. 18