Optimization of solver for gas flow modeling

D Savichkin¹, O Dodulad, Yu Kloss
National Research Centre "Kurchatov Institute"
1 Akademika Kurchatova pl, Moscow 123182, Russian Federation
E-mail: denis.savichkin@gmail.com

Abstract. The main purpose of the work is optimization of the solver for rarefied gas flow modeling based on the Boltzmann equation. Optimization method is based on SIMD extensions for x86 processors. Computational code is profiled and manually optimized with SSE instructions. Heat flow, shock waves and Knudsen pump are modeled with optimized solver. Dependencies of computational time from mesh sizes and CPU capabilities are provided.

1. Introduction
In present work we solve the Boltzmann equation by finite-difference method by using the conservative projection method for calculation the collision integral [1]. The method states that the laws of mass conservation, momentum and energy are strictly observed, and the collision integral of the Maxwellian distribution equals to zero. Real molecular potentials can be used in simulation. The method can be applied to modeling a single component gas, a gas mixture and a gas with internal degrees of freedom.

A multidimensional modeling uses much memory and computing resources. Accuracy of computational method depends on mesh of covering. To model the rarified gas we use Boltzmann equation [2]. Problem-Solving Environment [3] designed for simulation of rarefied gas flows was developed on the basis of conservative projection method and supports multi-processor systems based on MPI [4]. The Problem-Solving Environment (PSE) allows one to perform calculations on personal computer, including the use of graphics processors [5].

Due to limited resources some bottlenecks are located using performance analyzers. The code is optimized with SSE and AVX instructions. A version for Intel MIC architecture is now under development. The work presents results of simulation of some slightly disturbed flows as well as supersonic flows. Examples of the slightly disturbed flows show modeling of Knudsen pump [6] and its’ multistage modification. The accuracy of the obtained results is proved by parametric computations and is only limited by computational resources.

2. Kinetic equation
The Boltzmann equation is written as [2],

\[
\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{x}} = I(\vec{\xi})
\]  

Equation (1) is solved on a grid \(x_i\) and on a uniform grid consisting of \(N_0\) equidistant velocity nodes \(\xi_i\) with mesh size of \(h\) introduced in domain \(\Omega\) of volume \(V\). The velocity grid is
constructed as follows. A cube is defined in the velocity space and a uniform grid is constructed in it. Then, a sphere is inscribed in the cube, and all the outer nodes are eliminated. As a result, the number of grid nodes is roughly halved.

We use of the free-stream density $n_0$ and $T_0$ temperature values as the reference values [1]. The scales of velocity $v_0$, length $\lambda$, and time $\tau_0$ are defined as

$$v_0 = \sqrt{kT_0/m}, \lambda = 1/\sqrt{2\pi n_0 \sigma_{eff}^2}, \tau_0 = \frac{\lambda}{v_0}$$

Here, $v_0 = v_T/\sqrt{2}$, where $v_T$ is the mean thermal velocity of molecules according to [7]. $\lambda$ is the free path of molecules, $\tau_0 = \tau_T/\sqrt{2}$, where $\tau_T$ is the mean free time. Knudsen number is $Kn = \lambda/L$, where $L$ is the characteristic size of the flow. Proceeding to dimensionless variables and a dimensionless distribution function $\xi^* = \xi/v_0$, $t^* = t/\tau_0$, $b^* = b/\sigma_{eff}$, $x^* = x/\lambda$, $f^* = f/(n_0 v_0^{-3})$, and retaining the previous notation, we obtain the following equation in dimensionless variables: [6]

$$\frac{\partial f}{\partial t} + \xi \frac{\partial f}{\partial x} = \frac{1}{2\pi} I(\xi)$$

The hard sphere model of a molecular potential is considered in this paper. The effective diameter $\sigma_{eff}$ equals to $\sigma_\infty$ and $b_m$ equals to $\sigma_\infty$ for this model.

Equation (1) is split into advection equation and collision operator. Splitting is written as $S_r = A_r/2C_rA_r/2$ where $S_r$ is operator for iterations $t_0$ through $t_0 + \tau$. $A_r/2$ is free molecular flow

$$\frac{\partial f}{\partial t} + \xi \frac{\partial f}{\partial x} = 0$$

with time step $\tau/2$ and $C_r$ is operator of collision relaxation

$$\frac{\partial f}{\partial t} = I(f,f)$$

with time step $\tau$. The boundary conditions are used to solve advection equations. The same cycle is repeated at the next time step, but a new grid $S_r$ of integration nodes is generated for computing the collision operator. This grid is obtained by periodic shift of the original one through a random 8-dimensional vector. The advection equation is solved using first- and second-order (TVD scheme [7]) accurate conservative difference scheme. The method for solving Eq. (5) ensures that solution is nonnegative.

The collision integral is written as

$$I(f,f) = \int_{-\infty}^{\infty} \int_{0}^{2\pi} b_m \int_{0}^{2\pi} \phi \left(f'f_* - ff_*\right) db db d\xi$$

Where $g = |\xi - \xi_\epsilon|$, $b_m$ is the target distance, $\epsilon$ is the collision scattering angle. After introducing

$$\phi(\xi_\gamma) = \delta(\xi - \xi_\gamma) + \delta(\xi_\epsilon - \xi_\gamma) - \delta(\xi' - \xi_\gamma) - \delta(\xi_* - \xi_\gamma)$$

at the point $\xi_\gamma$ we obtain

$$I(\xi_\gamma) = \int_{-\infty}^{\infty} \int_{0}^{2\pi} b_m \int_{0}^{2\pi} \phi \left(f'f_* - ff_*\right) db db d\xi d\xi_*$$

To evaluate Eq. (7), we define the domain $\Omega \times \Omega \times [0, 2\pi] \times [0, b_m]$ with mesh $\xi_\alpha$, $\xi_\beta$, $\epsilon_\nu$, $b_\nu$ and use Korobov’s uniform grid [8]. Note that, points after collision $\xi'_\alpha$, $\xi'_\beta$ do not coincide with velocity nodes. For velocity $\xi'_{\alpha_{\nu}}$, two nearest points $\xi_\lambda$, $\xi_{\mu}$ are determined. Two points $\xi_{\lambda_{\nu}, \mu}$
and $\xi_{\mu_s-s_\nu}$ are located symmetrically from sum $g_\nu = \xi_{\alpha_\nu} + \xi_{\beta_\nu}$ and determined for velocity $\xi'_{\beta_\nu}$. Since that delta-functions in Eq. (7) turn to $\delta(\xi'_{\alpha_\nu} - \xi_{\gamma}) = (1 - r_\nu)\delta(\xi_{\alpha_\nu} - \xi_{\gamma}) + r_\nu \delta(\xi_{\mu_\nu} - \xi_{\gamma})$. Where the coefficient $r_\nu$ is obtained from the conservation laws: $(\xi'_\alpha + \xi'_{\beta_\nu})^2 = (1 - r_\nu)(\xi^2_{\alpha_\nu} + \xi^2_{\mu_\nu}) + r_\nu(\xi^2_{\alpha_\nu + s_\nu} + \xi^2_{\mu_\nu - s_\nu})$.

The distribution functions $f'_{\alpha_\nu} f'_{\beta_\nu}$ after collision can be found by interpolation

$$(f_{\lambda_\nu} f_{\mu_\nu})^{1-r_\nu} (f_{\lambda_\nu + s_\nu} f_{\mu_\nu + p_\nu})^{r_\nu}$$

This interpolation gives us zero for the collision operator if $f_\beta$ are the values of Maxwellian function at the velocity nodes. Using of the logarithmic interpolation Eq. (8) is important to modeling slightly disturbed flows. Modeling supersonic flows with existing computational resources becomes extremely time consuming. Eq. (9) is a transformation to replace the resource-intensive operation of multiplication by less expensive sum.

$$\exp [(1 - r_\nu) \ln (f_{\lambda_\nu} f_{\mu_\nu}) + r_\nu \ln (f_{\lambda_\nu + s_\nu} f_{\mu_\nu + p_\nu})]$$

After a predetermined number of steps PSE saves the set of macro-parameters which are calculated according to the formulas for density $n$, velocity $u$, temperature $T$ and pressure $P$

$$n = \int f d\xi, \quad u = \frac{1}{n} \int \xi f d\xi, \quad T = \frac{1}{3n} \int \xi^2 f d\xi - u^2, \quad P = nT$$

3. Optimization methods

3.1. Compilers’ comparison and operation time for GPU-based solver

Structure of PSE is shown on Figure 1. There are some components of multiform input, GPU-adapted calculation core and solver with unstructured meshes and various outputs.

Heat flow problem [9] modeling was compiled with GNU g++ 4.4.3 with –O3 and Intel icpc 13.0.1 with a similar option. Korobov’s grid equaled to $10^5$. The number of spatial cells varied from 30 to 1000. Operation time of the code is shown in Table 1. The comparison result shows that there are no significant differences between two compilers. However, there are many optimization options in modern compilers some parts of code have to upgrade manually.

![Figure 1. Structure of the problem solving environment.](attachment:image.png)
Table 1. Compilers' comparison.

| Compiler | Nodes | 30  | 50  | 100 | 200 | 500  | 1000 |
|----------|-------|-----|-----|-----|-----|------|------|
| g++      |       | 130s| 173s| 272s| 463s| 1066s| 2124s|
| icpc     |       | 122s| 163s| 272s| 479s| 1109s| 2193s|

For GPU-based code there were pre-generated collision meshes and one of them was randomly chosen on each time step. Comparison of optimization based on graphics processors is listed in Table 2.

Table 2. Operation time for GPU-based solver.

| Mesh in physical space | 64x64 | 96x96 | 128x128 | 160x160 |
|------------------------|-------|-------|---------|---------|
| Acceleration of transfer | 210   | 347   | 395     | 422     |
| Acceleration of relaxation | 32.5  | 35.6  | 39.5    | 33      |
| Total acceleration (times) | 105   | 139   | 150     | 135     |

Result was produced with a single core CPU (2.66GHz) and 240 cores GPU (GTX285). Speedup for calculations transfer part of solver grown for mesh sizes 64x64, 96x96 and 128x128. However, mesh size didn’t govern on speedup for the relaxation part of solver. For small meshes transfer part influenced on general operation time. Then operation time of transfer part became negligible. And due to weak paralyzation potential for collision integral algorithm there was speedup decreasing for 160x160 mesh.

3.2. SIMD optimization

Based on profiling shown on Figure 2 the function ci::iter must be taken into account. That function calculates Equation (8). Function calc_int_node generates Korobov’s grid and collision nodes. Function layer calculates transfer with boundary conditions by Eq. (4). Function main saves macro-parameters by Eq. (10). A pow is a function from standard C++ library which calculates power in Eq. (8).

![Figure 2. Intel VTune profiling output.](image)

There are two distribution functions are needed for algorithm. The data storage of each distribution function is 6-dimensional array in the physical and velocity space. A linear array
matches to a 3-dimensional array for each space. Solver has modules for transfer and relaxation part of solver. The code solves Eq. (5) is given below:

```c++
// loop by the physical space
for (size_t it = 0; it < Nr; ++it)
    
// loop by the velocity space
for (typename Nodes::const_iterator p = nodes.begin(); p != nodes.end(); ++p)
    
    // get data
    double g3 = f(i1m + it);
    double g4 = f(i2m + it);

    // main calculation
    double d = (-g3*g4 + g1*g2) * n.c;
```

According to profiling in Fig. 2 the hardest part of calculation is Eq. (9), which is optimized manually. The collision integral algorithm have weak thread parallelism. It’s explain speedup limit of GPU-optimized solver for Eq. (5). SIMD extensions provides us to use data with parallelism. In listing the loaded data are not contiguous in the application memory space. For the use SSE instructions the data access pattern have to be a unit-stride. The data storage structure must be inverted. The order of loop by the velocity and physical space is changed. This process allows to operate with the prepared vector of data and shown in listing bellow:

```c++
// loop by the velocity space
for (typename Nodes::const_iterator p = nodes.begin(); p != nodes.end(); ++p)
    
    // loop by the physical space
    for (size_t it = 0; it < Nr; it += 2)
        
        // load data in registers
        __m128d fast11 = _mm_load_pd(&f(i1l));
        __m128d fast21 = _mm_load_pd(&f(i2l));

        // main calculation
        __m128d ex = sse::exp(_mm_add_pd(_mm_mul_pd(_mm_add_pd(_mm_mul_pd(sse::log(f1l21), mrf), _mm_mul_pd(sse::log(f1m2m), rf)), _mm_add_pd(sse::log(f1m21), rf)));

        // store data from registers
        _mm_store_pd(&f(i1l), fast11);
        _mm_store_pd(&f(i2l), fast21);
```

In each memory access, the core loads not one byte, but a whole cache line, which is 16 bytes wide. With SSE instructions processor core load two different cache lines for double precision numbers. The dimension of the data vector depends on the device processor architecture. Using
SSE halves the length of the loop in a physical space. This optimization lets us to avoid the constant movement of the data from memory to the cache and back. Thus more CPU time is used optimally. The theoretical speedup for Eq. (9) is 2 (SSE), 4 (AVX), 8 (MIC).

The dependence of the operation time from the number of cores is shown in Figures 3 and 4. Calculation is done until the hundredth time step. The standard Linux tool (time) is used to determine parameters.

Figure 3. Operation time for different meshes of standard algorithm based code.

Figure 4. Operation time for different meshes of tuned algorithm based code.

Figure 3 is a picture of the old unmodified algorithm and Figure 4 is a tuned solver core. The average gain in time is about 10-20%. The linear growth can be seen in the number of spatial cells for both cases. A similar dependence is observed on the power of Korobov’s grid [8].

Figure 5. Acceleration of solver from number of processor threads.

Figure 5 shows dependencies of acceleration from number of threads. Data for this plot is obtained on one node of cluster (two Xeon X5680 processors) as a solution to a shock wave problem.

4. Physical problems
The problem of calculating the parameters of the shock wave was examined in detail in [1]. Density jump from $n_1 = 1$ to $n_2 = 3$ happened in the space of a lot bigger than the common sizes. Parameters of gas before and after the shock associated by Rankine-Hugoniot conditions. There are same conditions the border of the Maxwell function. The simulation result is shown in Figure 6. Korobov’s grid equaled to $10^6$, the number of spatial cells is $N_x = 480$. The transfer equation was approximated by the first order scheme. The calculation was performed by 16 cores (2 cluster nodes).
Next problem was modeling parameters of the gas in thermos with the wall being heated. The temperature of the left wall increased linearly according to \( T = T_0 + \alpha t \), where \( \alpha = 0.1 \), \( T_0 = 1 \). Korobov’s grid equaled to \( 10^5 \). The number of spatial cells was \( 10^3 \). The transfer equation was approximated by the first order scheme. The Knudsen number was \( Kn = 0.0015 \). The result for propagating density at different time steps are shown on Figure 7. The \( x \)-axis was normalized by 1 and started from point 0.06. Similar results with the work of A. Sacurai et al. [10] was gained.

![Figure 6. Density(0), velocity(1), temperature(2) and \( T_{xx}(3) \) depend on space coordinates for steady shock wave.](image1)

![Figure 7. Propagating density wave from \( x \) at different times for heat flow problem.](image2)

The calculation of the parameters of the gas at the Knudsen pump [6] is shown on Figure 10. There was a complex geometry of Knudsen pump consisting of two tanks connected by a set of eighteen narrow tubes and tubes with a tenfold larger diameter. The central (narrow and wide) tubes had temperature gradient with a linear doubling. The device had a monatomic gas inside with the initial density equal 1 and temperature with a leaner doubling. Capacity was divided into four parts mirror boundary conditions due to the symmetry. On the remaining walls of the diffuse boundary conditions with the accommodation coefficient equaled to 1. Korobov’s grid equaled to \( 5 \times 10^4 \) the number of spatial cells of the order was \( 1.5 \times 10^4 \). The transfer equation was approximated by the first order scheme. The calculation was performed by 120 cores (10 cluster nodes).

![Figure 8. Geometry of Knudsen pump with unstructured mesh.](image3)

The task of calculating the parameters of the gas in the cavern was modeled. The bottom wall of the container had a 0.001 of the speed of thermal molecular velocities. Mesh of speed was 20 grids by radius, the total number of nodes was 4224, spatial grid was 128X128. Power Korobov’s grid equaled to 50000.

Simulation time of all the tasks can be predicted by comparing the characteristics of computational grids in Figure 4. The results of measurements of the code speed are shown in Table 3.

There are satisfactory result for heat flow problem. Small cash size and old processors architecture may have a significant effect on the performance gain. For Knudsen pump modeling
Figure 9. Conditions of gas modeling into cavern and streamlines in equilibrium.

Figure 10. The ratio of pressure in the two boxes of Knudsen pump from time steps.

Table 3. Operation time of solver.

| Problem           | simple | tuned  | acceleration | node     |
|-------------------|--------|--------|--------------|----------|
| Heat flow         | 2h 7min| 1h 29min| 30%          | Core-i5  |
| Shock wave        | 10h 44min| 9h 13min| 14%          | 2× Xeon 5365 |
| Knudsen pump      | 53min  | 59min  | –            | 20× Xeon 5680 |

with $5 \times 10^4$ Korobov’s grid means that main part of calculation is transfer and data transposition function may result in a significant performance loss. It leads to grow the operation time for such problems.

5. Conclusions
More efficient code was obtained and tested on different problems. Simulations of slightly disturbed flows and supersonic flows were provided. Examples of the slightly disturbed flows were shown by modeling of Knudsen pump and its multistage modification.

The total operation time achieved on the architecture of Intel Xeon was 3min 18s which is 1.44 times faster than in previous code version (4min 46s). Speedup 1.73 for the architecture of the Intel Phi with wide vector processing units gained from 29min 37s up to 17min 8s. As a result, approximately 30 - 40% performance increase of the total time of the code was received for supersonic flows.

Acknowledgments
Results were obtained with using resources Supercomputer Simulation Laboratory SUSU (http://supercomputer.susu.ac.ru) and resources MCS NRC Kurchatov institute (http://computing.kiae.ru). Special thanks are extended to D. Babokin of Intel for providing Intel Phi coprocessor.

References
[1] Tcheremissine F G 2006 Comp. Math. and Math. Phys. 46 329–343
[2] Kogan M N 1967 Rarefied Gas Dynamics (Moscow: Nauka) p 440
[3] Anikin Y A, Dodulad O I, Kloss Y Y, Martynov D V, Shuvalov P V and Tcheremissine F G 2012 Vacuum 86 1770–1777
[4] Chapman B, Jost G and van der Pas R 2008 Using OpenMP (MIT press) ISBN 978-0-262-53302-7
[5] Kloss Y, Shuvalov P and Tcheremissine F 2010 Procedia Computer Science 1 1077–1085
[6] Takata S and Umetsu H 2011 AIP Conference Proceedings
[7] LeVeque R J 2007 Finite Volume Methods for Hyperbolic Problems (Cambridge University Press) ISBN 978-0-521-00924-9
[8] Korobov N M 1989 Trigonometric Sums and Applications (Moscow: Nauka)
[9] Ohwada T 1996 Phys. Fluids 8
[10] Sakaguchi G, Tsukamoto M and Sakurai A 2012 28th International Symposium on Rarefied Gas Dynamics 1501 131–136