2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

doi 10.26089/NumMet.v22r421

Implementation and Performance of Wave Tomography Algorithms on SIMD CPU and GPU Computing Platforms

A. V. Goncharsky
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: https://orcid.org/0000-0001-9953-611X, e-mail: gonchar@srcc.msu.ru
S. Y. Romanov
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: https://orcid.org/0000-0001-7882-2061, e-mail: romanov60@gmail.com
S. Y. Seryozhnikov
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: https://orcid.org/0000-0002-4106-2692, e-mail: s2110sj@gmail.com

Abstract: This paper is concerned with implementation of wave tomography algorithms on modern SIMD CPU and GPU computing platforms. The field of wave tomography, which is currently under development, requires powerful computing resources. Main applications of wave tomography are medical imaging, nondestructive testing, seismic studies. Practical applications depend on computing hardware. Tomographic image reconstruction via wave tomography technique involves solving coefficient inverse problems for the wave equation. Such problems can be solved using iterative gradient-based methods, which rely on repeated numerical simulation of wave propagation process. In this study, finite-difference time-domain (FDTD) method is employed for wave simulation. This paper discusses software implementation of the algorithms and compares the performance of various computing devices: multi-core Intel and ARM-based CPUs, NVidia graphics processors.

Keywords: wave tomography, inverse problem, SIMD, GPU, ARM, benchmark.

Acknowledgements: The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University. This work was supported by Huawei Technologies Co., Ltd. (Project No. OAA20100800391587A).

For citation: A. V. Goncharsky, S. Y. Romanov, S. Y. Seryozhnikov, "Implementation and Performance of Wave Tomography Algorithms on SIMD CPU and GPU Computing Platforms," Numerical Methods and Programming. **22** (4), 322–332 (2021). doi 10.26089/NumMet.v22r421.

1. Introduction.

Wave tomography technology aims to determine the internal structure of the object using the wave field scattered by the object and recorded by detectors. Tomographic imaging methods using wave sources are currently being developed in Russia, USA and Europe. Primary applications of wave tomography are medical ultrasound tomography, nondestructive testing, electromagnetic sounding [1–4]. Physical experiments on tomographic nondestructive testing [5] and medical ultrasound tomography [6–8] are being conducted by the authors.

Unlike X-ray tomography, inverse problems of wave tomography are nonlinear ill-posed problems with large number of unknowns [9, 10]. Breakthrough results in the field of solving inverse problems of wave tomography in scalar wave models were obtained in [1, 11, 12]. In these works, representations for the gradient of the residual functional, have been obtained for various formulations of the inverse problem. These results open up the possibilities for employing gradient iterative methods for solving inverse problem of wave tomography [13].

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

The residual functional is the difference between the measured and computed wave fields; thus, gradient-based methods rely on numerical simulations of wave propagation process [6–8] performed repeatedly to obtain successive approximations of the exact solution. Computing capabilities of the hardware are of great importance for practical implementation of wave tomography methods.

A scalar wave model is used for numerical simulations in this study. This model accounts for such physical phenomena as diffraction, refraction and multiple scattering. The better the wave model approximates real physical processes, the better approximate solution can be obtained. Scalar wave model accounts for waves propagating in all directions, unlike commonly used Born and Rytov approximations that are valid only for a narrow angular range [14].

The wave tomography algorithms have been implemented in open-source software published by the authors. The performance of various CPU and GPU devices on a range of typical model problems of wave tomography was assessed. ARM-based processors are being increasingly used at present time in server and HPC applications, and in many cases ARM CPUs demonstrate better energy efficiency and cost efficiency comparing to Intel-architecture CPUs [15]. One of the aims of this study is to evaluate the performance of ARM-based Kunpeng-920 CPUs manufactured by Huawei Technologies in solving inverse problems of wave tomography. The CPU computing platforms tested were Intel Haswell-EP E5-2697v3, 2.6 GHz, 14 cores, Intel 6240R dual-CPU server, 2.4 GHz, 24 cores per CPU, and TaiShan 5280 dual-CPU server equipped with ARM-based Kunpeng-920 CPUs, 2.4 GHz, 48 cores per CPU. The GPU platforms tested were NVidia Tesla P100 and NVidia Tesla V100. The computations were carried out on "Lomonosov-2" supercomputer at the Lomonosov Moscow State University [16] and the hardware provided by Huawei Technologies Co., Ltd.

2. Formulation of the inverse problem of wave tomography and its solution method. The basic scheme of a tomographic examination is shown in Figure 1a. The object being imaged occupies region G. In medical imaging, region L is filled with water with known $c_0 = \text{const}$. The detectors are located on circle Γ. The objective is to reconstruct the speed of sound in region G using the ultrasonic waves radiated from the emitters, scattered by the object and registered by the detectors.

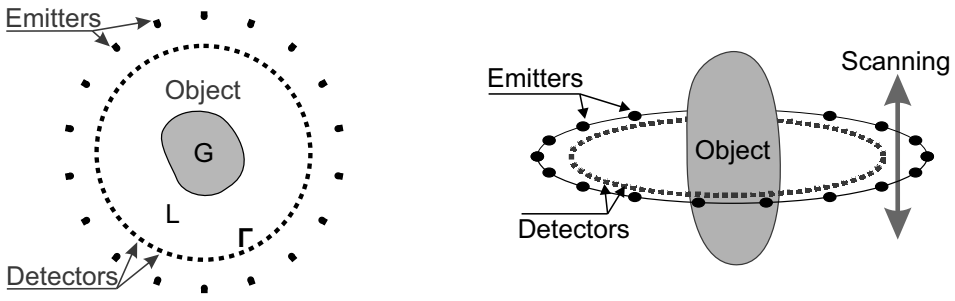
Figure 1b shows the scheme of the layer-by-layer 3D wave tomography. Emitters and detectors are located in a horizontal plane and can shift relative to the object in vertical direction. The images are acquired in multiple horizontal imaging planes. A 3D image of an object is represented in the form of a stack of 2D cross-sections in this formulation.

The inverse problem of wave tomography is posed as a coefficient inverse problem. The unknowns are the speed of sound and the absorption factor at each point of the object. A scalar wave model based on a second-order hyperbolic differential equation (1) is used for numerical simulation of wave propagation process. This model accounts for diffraction, refraction, multiple scattering and absorption of ultrasound waves:

$$c(\boldsymbol{r})u_{tt}(\boldsymbol{r},t) + a(\boldsymbol{r})u_t(\boldsymbol{r},t) - \Delta u(\boldsymbol{r},t) = 0; \tag{1}$$

$$u(\boldsymbol{r},t)\big|_{t=0} = F_0(\boldsymbol{r}), \quad u_t(\boldsymbol{r},t)\big|_{t=0} = F_1(\boldsymbol{r}). \tag{2}$$

Here, $u(\boldsymbol{r},t)$ is the acoustic pressure; $c(\boldsymbol{r}) = 1/v^2(\boldsymbol{r})$, where $v(\boldsymbol{r})$ is the speed of sound; $a(\boldsymbol{r})$ is the absorption factor; $\boldsymbol{r} = \{x, y\}$ is a point in the imaging plane, and Δ is the Laplacian operator with respect to $\boldsymbol{r}$.



a) b)

Figure 1. Scheme of tomographic examination (a), layer-by-layer 3D tomography (b)

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

Initial conditions (2) represent the wavefield at the initial time of the simulation. Approximate non-reflecting boundary conditions [17] are applied at the boundary S of the computational domain:

$$\partial_n u\big|_{ST} = -c^{-0.5}\partial_t u\big|_{ST}.$$

The inverse problem of wave tomography is an ill-posed coefficient inverse problem for the wave equation (1). The objective is to determine the speed of sound $c(\boldsymbol{r})$ and absorption factor $a(\boldsymbol{r})$ inside the medium, while the wavefield $u(\boldsymbol{r},t)$ is known only at the detector positions. An approximate solution to the inverse problem can be obtained via minimizing the residual functional

$$\Phi(u(c,a)) = \frac{1}{2}\int\limits_0^T\int\limits_\Gamma (u(\boldsymbol{s},t) - U(\boldsymbol{s},t))^2\, d\boldsymbol{s}\, dt \tag{3}$$

for its argument (c,a). Here $U(\boldsymbol{s},t)$ are the data measured at surface Γ for the time period $(0,T)$, $u(\boldsymbol{s},t)$ is the solution of the direct problem (1)–(2) for given $c(\boldsymbol{r}) = 1/v^2(\boldsymbol{r})$ and $a(\boldsymbol{r})$. The residual functional is the sum of the residuals (3) obtained for each ultrasound emitter.

The gradient $\Phi'(u(c,a)) = \{\Phi'_c(u), \Phi'_a(u)\}$ of the functional (3) with respect to the variation of the sound speed and absorption factor $\{dc, da\}$ has the form:

$$\Phi'_c(u(c)) = \int\limits_0^T w_t(\boldsymbol{r},t)u_t(\boldsymbol{r},t)\, dt, \qquad \Phi'_a(u(a)) = \int\limits_0^T w_t(\boldsymbol{r},t)u(\boldsymbol{r},t)\, dt. \tag{4}$$

Here $u(\boldsymbol{r},t)$ is the solution of the direct problem (1) – (2), and $w(\boldsymbol{r},t)$ is the solution of the "conjugate" problem with the given $c(\boldsymbol{r})$, $a(\boldsymbol{r})$, and $u(\boldsymbol{r},t)$:

$$c(\boldsymbol{r})w_{tt}(\boldsymbol{r},t) - a(\boldsymbol{r})w_t(\boldsymbol{r},t) - \Delta w(\boldsymbol{r},t) = E(\boldsymbol{r},t); \tag{5}$$

$$w(\boldsymbol{r},t=T) = 0, \qquad w_t(\boldsymbol{r},t=T) = 0; \tag{6}$$

$$E(\boldsymbol{r},t) = \begin{cases} u(\boldsymbol{r},t) - U(\boldsymbol{r},t), \text{ where } \boldsymbol{r}\in\Gamma \text{ and } U(\boldsymbol{r},t) \text{ is known;} \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

In order to solve the inverse problem of wave tomography, a direct problem is solved to obtain the boundary values and the simulated wavefield $u(\boldsymbol{s},t)$ at the detectors, given the current approximate values of the coefficients $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$. The direct problem is solved for each ultrasound emitter at each iteration of the gradient descent method.

3. Numerical method. Finite-difference time-domain method (FDTD) is employed to solve equations (1) – (2). We define a uniform rectangular finite difference grid: $x_i = ih$, $y_j = jh$, $t_k = k\tau$; $i, j = 1,\ldots,N$, $k = 1,\ldots,M$, where h is the spatial discretization step, and τ is the time step. A second-order finite difference scheme approximates equation (1):

$$c_{ij}\frac{u_{ij}^{k+1} - 2u_{ij}^k + u_{ij}^{k-1}}{\tau^2} + a_{ij}\frac{u_{ij}^{k+1} - u_{ij}^{k-1}}{2\tau} - \frac{\boldsymbol{L}_{ij}^k}{h^2} = 0. \tag{8}$$

Here, $u_{ij}^k = u(x_i, y_j, t_k)$ are the values of $u(\boldsymbol{r},t)$ at point (i,j) at the time step k; c_{ij} and a_{ij} are the values of $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$ at point (i,j). The first term approximates $c(\boldsymbol{r})u_{tt}(\boldsymbol{r},t)$, the second term approximates $a(\boldsymbol{r})u_t(\boldsymbol{r},t)$. The discrete Laplacian is denoted by $\boldsymbol{L}_{ij}^k$. A fourth-order numerical approximation [18] on a 5×5-point stencil is used for the discrete Laplacian:

$$\boldsymbol{L}_{ij}^k = \sum_{i_0=i-2}^{i+2}\sum_{j_0=j-2}^{j+2} v_{i_0 j_0}u_{i_0 j_0}^k.$$

Collecting the terms with u_{ij}^{k+1} in (8), we obtain an explicit finite-difference scheme for the wave equation (1):

$$u_{ij}^{k+1} = \left(2\frac{c_{ij}}{\tau^2}u_{ij}^k + \frac{\boldsymbol{L}_{ij}^k}{h^2} + \left(\frac{a_{ij}}{2\tau} - \frac{c_{ij}}{\tau^2}\right)u_{ij}^{k-1}\right)\left(\frac{a_{ij}}{2\tau} + \frac{c_{ij}}{\tau^2}\right)^{-1}.$$

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

This scheme allows us to compute the wavefield $u(\boldsymbol{r}, t)$ sequentially in time, starting witn initial conditions (2). The parameters h and τ are related by the Courant stability condition $c^{-0.5}\tau < h/\sqrt{2}$. For the problem considered, we used a time step equal to $\tau = 0.3\, c_0^{0.5} h$, which ensured the stability of the finite difference method. Since h and τ are proportional, the total number of operations in a numerical simulation of wave propagation is proportional to $O(N^3)$, where N is the number of grid points along spatial dimensions. The number of points N is chosen so that the wave simulation is precise enough for a selected wavelength range. Thus, computing power requirements rise as a third power of wave frequency and spatial image resolution.

An approximate solution to the inverse problem can be obtained via an iterative gradient descent method. The gradient of the residual functional is computed, and the current approximation of coefficients $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$ is updated: $\{c^{(n+1)}, a^{(n+1)}\} = \{c^{(n)}, a^{(n)}\} - \alpha \cdot \{\Phi'_c(u), \Phi'_a(u)\}$, where n is the iteration number. The process stops when the residual functional reaches the level determined by measurement errors and numerical simulation errors and does not decrease anymore. Each iteration involves solving direct (1)–(2) and conjugate (5)–(7) problems, which require simulating the wave propagation process in forward and reverse time.

The gradient descent method involves computing successive approximations of unknown coefficients over many iterations. Wave propagation and back-propagation simulations must be performed at each iteration for every ultrasound emitter, and the approximate solutions found for every cross-section of the object, resulting in tens of thousands of wave simulations in total. This makes the method very computationally expensive.

4. Software implementation. Open-source "WaveTomography" software package developed for this study implements the algorithms for solving direct and inverse problems of wave tomography for Intel x86-64, ARM and GPU computing platforms. This software can be used in research and educational projects on wave tomography, computational diagnostics, numerical simulation and supercomputer technology. The software is implemented in C++ using open-source "vectorclass" library for Intel-compatible processors and OpenCL interface for GPU computing. ARM version of the numerical algorithm uses intrinsics and GCC vector extensions so that the code translates directly to SIMD instructions.

Figure 2 illustrates the direct problem solution algorithm. The wave field is simulated using a predefined numerical phantom that specifies the parameters $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$ in a single imaging plane. The software can automatically generate randomized test phantoms for demonstration purposes. Automatic routines ensure that the generated model problems are solvable. Emitters and detectors are placed in a circular formation around the phantom (Fig. 1a). The wavefield is simulated starting from the initial pulse that is computed analytically as a wave radiating from the emitter position. The base wavelength, bandwidth and beam width of the initial pulse are determined by the simulation parameters. The wave field at the detectors for each emitter is recorded in the output dataset.

Figure 3 illustrates the inverse problem solution algorithm. An approximate solution to the inverse problem of wave tomography is computed via the iterative gradient descent method using the previously computed simulated dataset as input. A constant initial approximation of the coefficients is assumed at the beginning of the iterative process. The direct problem is solved for the current approximation. The boundary values of $u(\boldsymbol{r}, t)$ wavefield are stored in the memory buffer in order to reverse the wave propagation direction of $u(\boldsymbol{r}, t)$ in formula (4). Then, $w(\boldsymbol{r}, t)$ wave is computed from $u(\boldsymbol{r}, t)$ and the input dataset, and the gradient is computed using formula (4). The approximate solution is updated according to the gradient and the process is repeated. At the end of the process, the approximate solution is the output of the inverse problem solution algorithm.

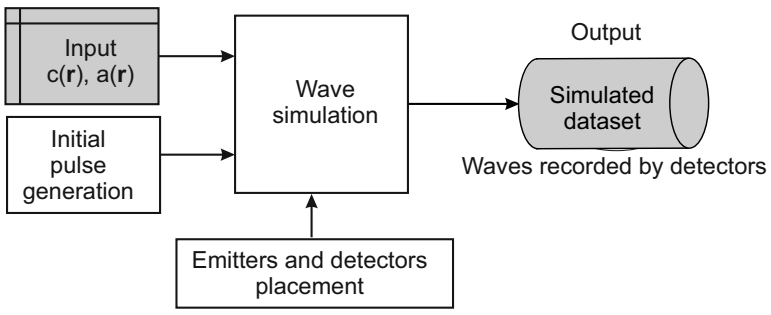


Figure 2. Direct problem solution algorithm

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

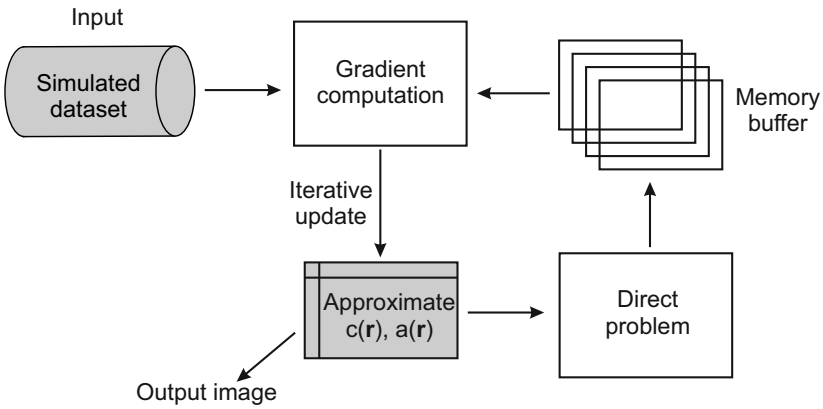


Figure 3. Inverse problem solution algorithm

The software performs a limited number of iterations of the gradient descent method of minimizing the residual functional and stops when the minimum is found or the limit on the number of iterations is reached.

The iterative gradient descent method allows for all the computations within a single iteration to be executed in parallel. SIMD-parallel algorithm implementation is employed to compute the wavefield using formula (8). The discrete Laplacian in this formula constitutes the main computational load in this numerical method.

The flowchart of the SIMD algorithm for computing the discrete Laplacian is shown in Fig. 4. The Laplacian operator is spherically symmetrical; thus, fewer operations are needed, comparing to a general convolution-type problem.

For maximum data retention in CPU registers, Y-marching method is used to compute wave data. The results are calculated sequentially in vertical direction, while holding the amount of data that fit in the registers along the horizontal dimension. Using the input data vector and horizontally adjacent cells, partial sums are computed and stored in the register matrix, which contains the data for 5 lines of the image. The result vector is computed by multiplying the register matrix by the Laplacian coefficient vector. The algorithm advances to the next line by shifting the lines in the register matrix up and reloading the last line from the input vector. The data is shifted via renaming the registers, no actual operations are needed for shifting.

Modern processors (AVX, AVX-512, ARM NEON-class FPUs) typically contain 32 SIMD registers. There are three partial sums per line and five lines in the register matrix; thus, the input vector can be two registers long for the single wave simulation ($u(\boldsymbol{r},t)$ for the direct problem) and one register long for the dual wave simulation ($u(\boldsymbol{r},t)$ and $w(\boldsymbol{r},t)$ for the conjugate problem).

The computations on multi-core CPUs are parallelized using OpenMP library. MPI interface is used for data exchange between computing nodes (CPU sockets or GPU devices). Figure 5a,b illustrate the order of computations for multi-core CPUs and GPUs, respectively.

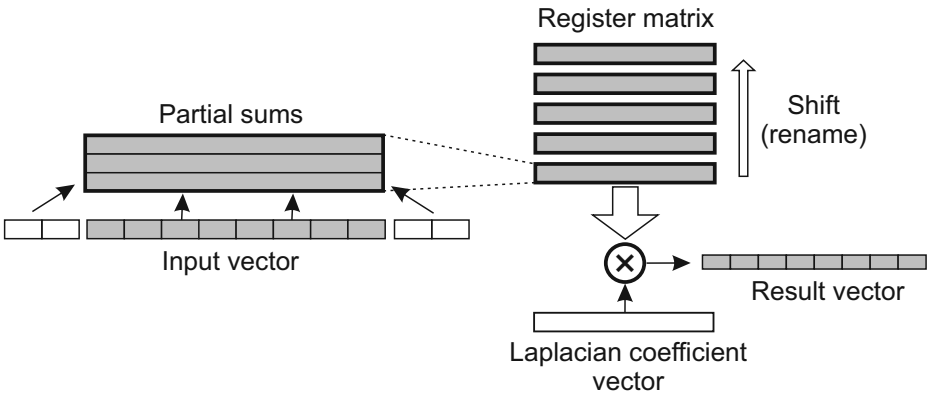


Figure 4. SIMD discrete Laplacian computation algorithm

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

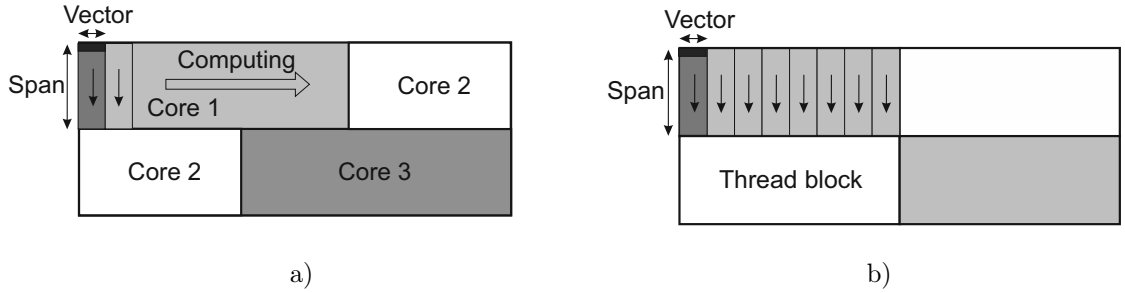


Figure 5. Parallelizing the computations on multi-core processors (a) and GPU (b)

In both cases, the calculations are performed in Y-marching order. For efficient use of cache memory, the length of the vertical segment ("Span") of Y-marching method is limited to some value specified at run-time. On multi-core CPUs, individual spans are computed in sequence along the horizontal dimension. An equal amount of data is processed by each computing core (with an accuracy of ± 1 span). For GPU, the computations are performed in parallel (automatically scheduled by the GPU) within each thread block. The optimal span length depends on image size and CPU cache and can be determined for each target system via performance tests. The better the data fits into the CPU cache, the longer span lengths are preferred. For GPUs, thread block size can also be adjusted for better performance.

Computation of the gradient of the residual functional is subdivided into independent sub-tasks for each ultrasound emitter. The total number of emitters in wave tomography typically ranges from 10 to 100. The emitters are divided evenly between the computing nodes. Partial gradients computed for each emitter are summed up within each node, and then summed between nodes using MPI interface. Data exchanges between nodes occur only once per iteration and therefore do not incur any noticeable delay.

5. Performance evaluation. The software was benchmarked on various SIMD CPU and GPU computing platforms: ARM-based Kunpeng-920, Intel x86-64, Intel 6240R and Intel Haswell-EP processors, NVidia Tesla P100 and Tesla V100 GPUs. The tests measure the output rate — the number of output (gradient) pixels computed per second. Dividing the output rate by the total number of data points per iteration, we obtain the overall performance as the number of gradient descent iterations per second:

$$\mathrm{IterationsPerSecond} = \mathrm{OutputRate}/\mathrm{ImageSize}^2/\mathrm{Emitters}/\mathrm{TimeFrames}.$$

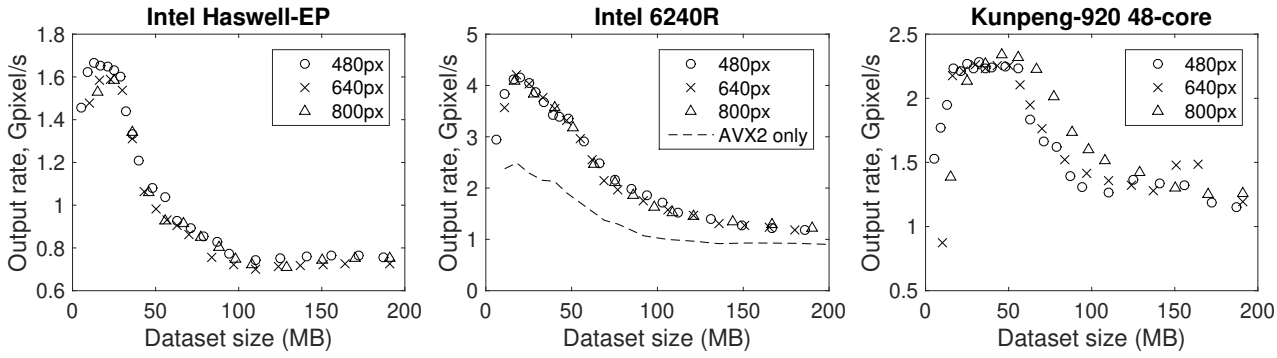
For CPUs, cache utilization has a major impact on performance. In order to optimize cache memory usage, the computations are grouped into one or more batches executed sequentially. A batch consists of the data for one or more ultrasound emitters that are processed in parallel. Too large batch sizes cause cache misses and frequent accesses to system RAM, decreasing the performance. With too small batch sizes, the batch execution time becomes very short and the thread synchronization latency becomes noticeable.

Quick tests were performed to determine the optimal batch size to hold in memory for parallel processing on each target computing system for maximum performance. The batch sizes may differ form the physical CPU cache size due to the use of an additional memory buffer (Fig. 3). Accesses to the buffer are less frequent than to the wave data, but they may offset the optimal dataset size from the physical CPU cache size.

The tests were performed for reconstructed image sizes of 480×480, 640×640 and 800×800 pixels. Each pixel of a reconstructed image uses 32 bytes of data. Test datasets of different sizes were created by varying the number of ultrasound emitters from 4 to 24. Figure 6 shows the performance test results for Intel Haswell-EP, Intel 6240R and Kunpeng-920 processors.

The optimal batch sizes were determined as 20–24 MB for Intel Haswell-EP and Intel 6240R CPUs and 50–55 MB for Kunpeng-920 processors. Intel 6240R processors feature AVX-512 vector FPU, which operates on four times larger vectors than ARM NEON-class FPU of Kunpeng-920 does. The tests showed that the performance of 48-core Kunpeng-920 CPU with 128-bit SIMD FPU is roughly equivalent to a 24-core Intel 6240R CPU operated with a vector size of 256-bit (AVX2-type). AVX2 mode performance is shown as a dashed line in Fig. 6b. While Intel 6240R CPU has a substantial computing power advantage, Kunpeng-920 processor's performance is much more stable across the cached range (15–60 MB) than that of Intel 6240R CPU. For very small datasets, a performance drop due to thread synchronization latency is noticeable, especially for

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421



a) b) c)

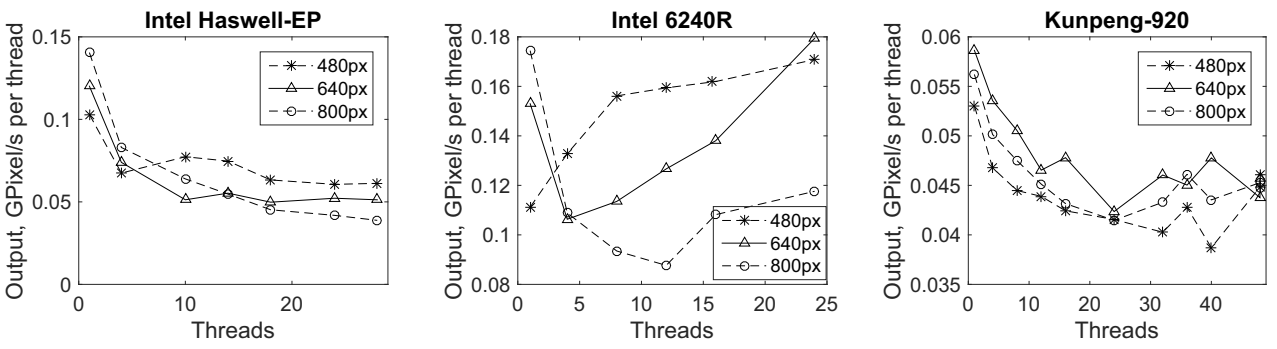
Figure 6. Performance depending on data size for Intel Haswell-EP (a), Intel 6240R (b) and Kunpeng-920 (c)

Kunpeng-920 CPUs, however, datasets that small (a low-resolution image and very few emitters) are rarely used in practice.

In order to assess the scalability and acceleration capabilities of multi-core processors, benchmarks were performed with different numbers of threads launched on the processors. Per-thread performance was measured. Figure 7 shows per-thread performance rates for Intel Haswell-EP, Intel 6240R and Kunpeng-920 processors, depending on the number of threads.

While on Intel processors a single thread can be up to 3 times faster than an average thread in a multi-threaded environment, on Kunpeng-920 processors a single thread is only 20% faster than an average thread. This result signifies that the threads in Kunpeng-920 CPU do not compete for any system resources, which is an architectural advantage of this CPU and provides stable performance for varying workloads, but also this result means that the CPU performance is limited by the computing cores: per-thread performance of Kunpeng-920 CPU is comparable to that of a much older Intel Haswell-EP CPU in a multi-threaded environment. The difference in performance for different dataset is also negligible for Kunpeng-920 CPU and more significant for Intel processors. On Intel 6240R processors, high volume of possibly simultaneous AVX-512 operations make the performance very unstable for small thread count.

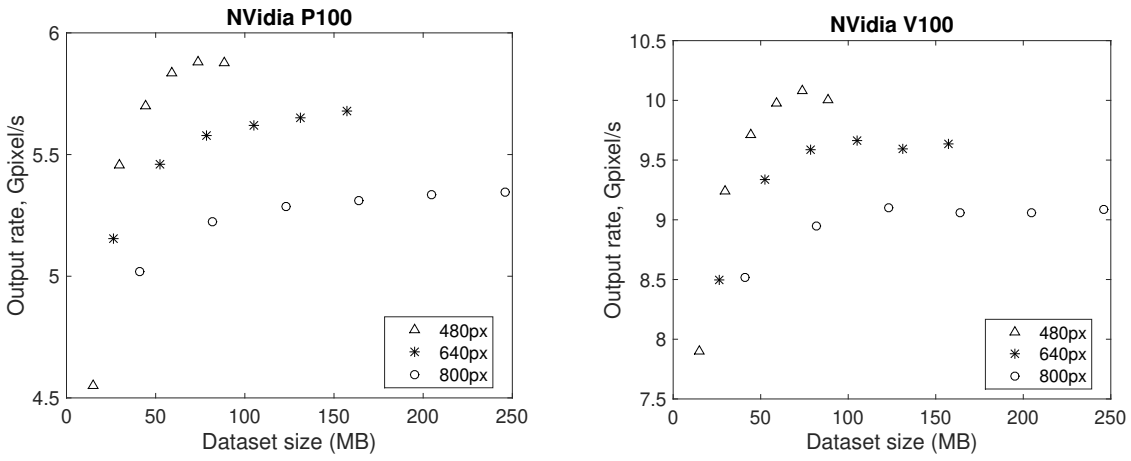
The software was tested on NVidia Tesla P100 and NVidia Tesla V100 graphics processors. Modern GPU devices are well-suited or numerical simulation algorithm like the wave simulation implemented in the developed software. The performance of GPU devices on wave simulation tasks is roughly proportional to the memory throughput of the devices. Figure 8 shows the performance of NVidia Tesla P100 (a) and NVidia Tesla V100 (b) on different input datasets. Average output rate of NVidia Tesla P100 amounted to 5.5 GPixel/s, which is 2.5



a) b) c)

Figure 7. Per-thread performance depending on the number of threads for Intel Haswell-EP (a), Intel 6240R (b) and Kunpeng-920 (c)

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421



a) b)

Figure 8. Performance of NVidia P100 GPU (a) and NVidia V100 GPU (b) on different datasets

times the rate of Kunpeng-920. Average output rate of NVidia Tesla V100 amounted to 9.2 GPixel/s, which is 4.5 times the rate of Kunpeng-920.

Figure 9 shows the overall performance of the tested devices relative to Kunpeng-920. Intel 6240R showed the best results among CPUs due to its AVX-512 FPU well-suited for computation-intensive tasks. GPU devices consistently outperform CPUs. GPUs were found to be the preferred architecture for solving direct and inverse problems of wave tomography. The computing tasks are data-parallel and do not require synchronized data exchanges between computing cores or cache coherence. Thus, the algorithms can benefit from the specific structure of graphics processors.

6. Image reconstruction results. Wave tomography image reconstruction fundamentally differs from X-ray tomographic imaging. While in X-ray imaging the detectors record a single value, in wave tomography the data recorded by detectors is a continuous waveform for some time period. The inverse problem of wave tomography is nonlinear and ill-posed. The reconstructed images represent two variables — sound speed and sound absorption factor inside the object.

The developed software is aimed to aid the research in medical imaging for breast cancer diagnosis, and the parameters of the simulations are set accordingly. In medical imaging, soft tissues have low contrast relative to water, but the imaging method must detect small inclusions, typically about 2 mm in size.

Figure 10 shows the numerical phantom used for the reconstructions. The phantom was generated automatically by the software. It has a diameter of 72 mm and random inclusions of varying sound speed and absorption factor. Two dots in the center of the image are 2 mm in size. The outer environment represents water with a speed of sound of 1.5 $\text{km}\cdot\text{s}^{-1}$ and no absorption.

Figure 11 shows the reconstruction results for low resolution: 480×480-pixel grid size, 9mm wavelength and 10 ultrasound emitters. Although the 2 mm inclusions are much smaller than the wavelength, they are

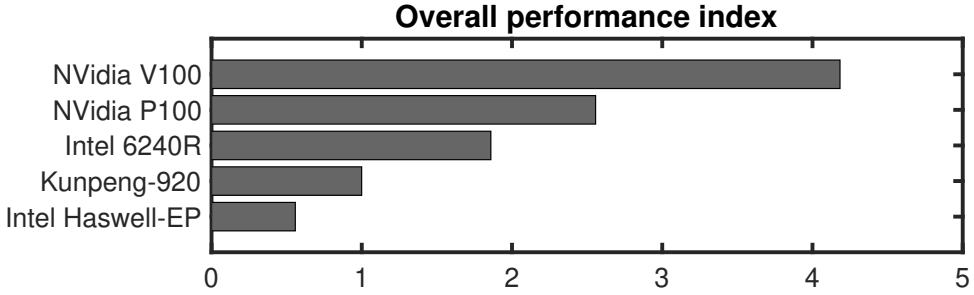
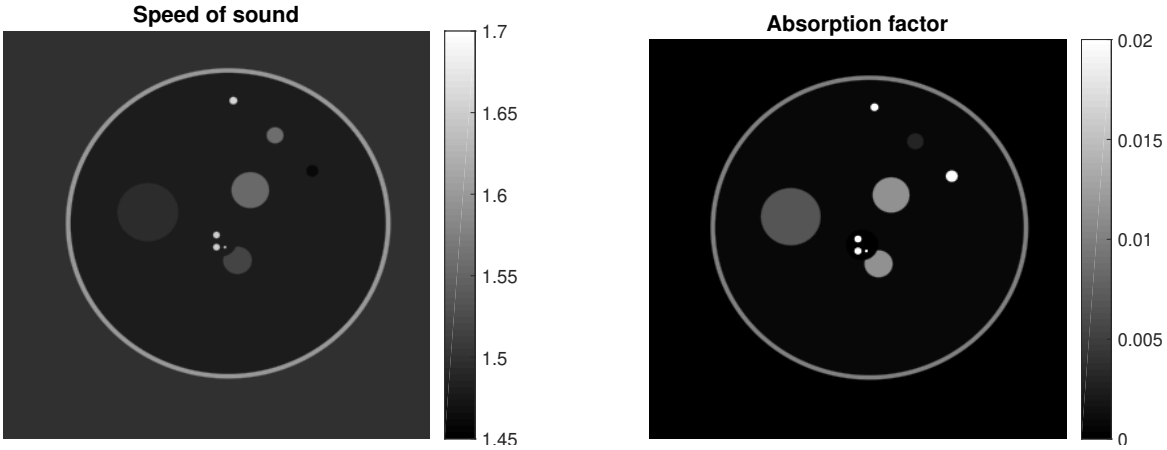


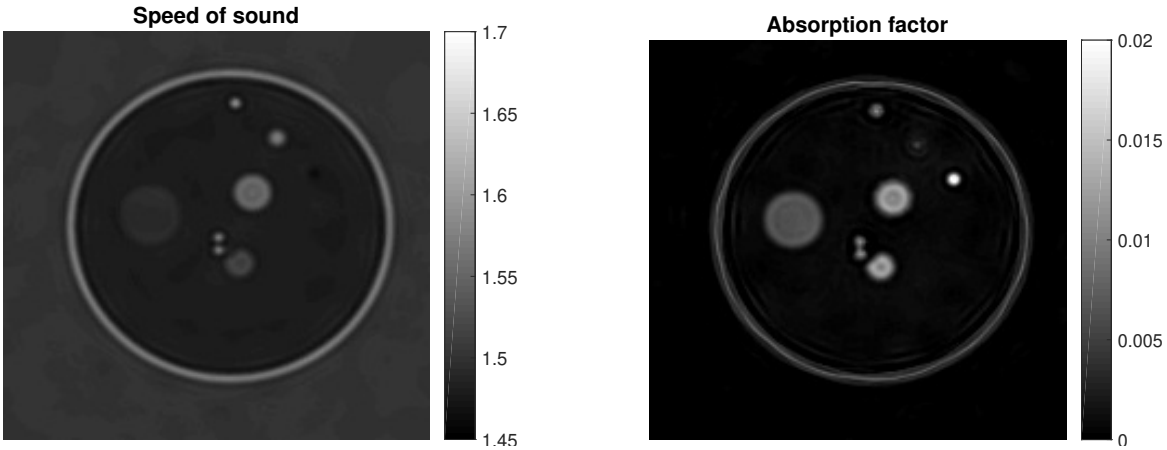
Figure 9. Overall performance of the computing devices

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421



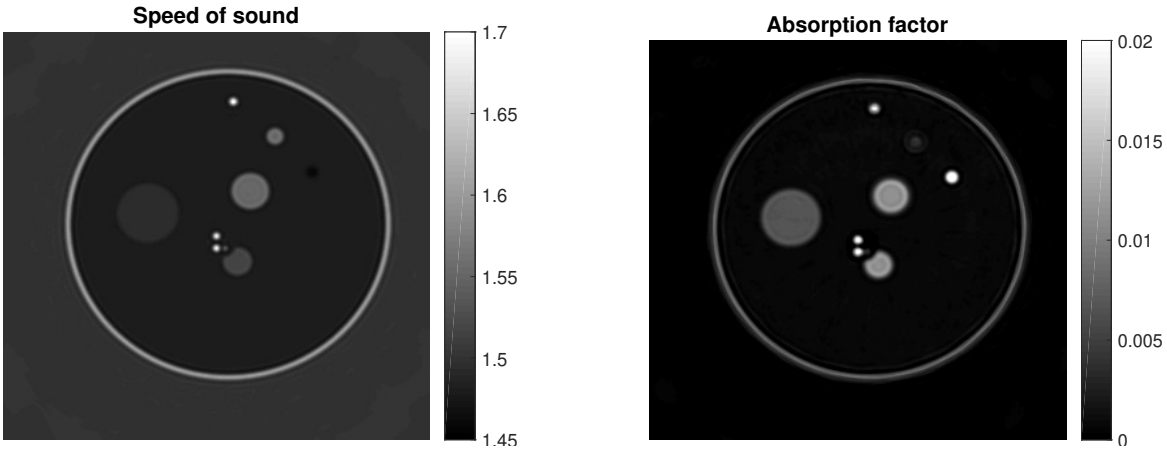
a) b)

Figure 10. Test phantom: speed of sound (a), absorption factor (b)



a) b)

Figure 11. Low-resolution reconstruction at 9 mm wavelength, 480 × 480 pixels, 10 emitters: speed of sound (a), absorption factor (b)



a) b)

Figure 12. High-resolution reconstruction at 5.6 mm wavelength, 800 × 800 pixels, 20 emitters: speed of sound (a), absorption factor (b)

visible in the image. A remaining effect of the waves can be seen at the perimeter of the phantom. Sound speed reconstruction quality (Fig. 11a) is higher than absorption factor reconstruction quality (Fig. 11b). The dataset for this example is 40 MB in size and contains data for 1039 simulation time frames.

Figure 12 shows the reconstruction results for high resolution: 800×800-pixel grid size, 5.6mm wavelength and 20 ultrasound emitters. The dataset for this example is 230 MB in size and contains data for 1733 simulation time frames. Typical computing time is 2 seconds per iteration at 480-pixel resolution and 10 seconds per iteration at 800-pixel resolution on a single Kunpeng-920 CPU. Approximately 100 iterations are needed to reconstruct the images. The initial approximation for the iterative process is a constant equal to the acoustical properties of water ($v_0 = 1.5\,\mathrm{km{\cdot}s^{-1}}$, $a_0{=}0$).

The images demonstrate that it is possible to reconstruct tomographic images from waveform data with high resolution and high sensitivity to changes in sound speed and absorption factor even if the wavelength is significantly longer than the size of an inclusion.

7. Conclusion. Open-source "WaveTomography" software for solving inverse problems of wave tomography on ARM-Based Kunpeng platform has been developed [19]. The software supports Intel x86-64, ARM and GPU processors, effectively uses SIMD FPU features present on the target systems. GPU is supported via industry-standard OpenCL interface.

A performance comparison of Kunpeng-920 CPUs to competing architectures was performed. Kunpeng-920 outperforms Intel Haswell-EP by 1.8 times. However, Intel 6240R CPUs equipped with AVX-512 FPU outperform Kunpeng-920 despite having twice less computing cores. GPU processors exhibit even higher performance. FPU and memory throughput are the most important factors for wave simulation tasks.

Performance and scalability of the algorithms have been assessed. Kunpeng-920 CPUs demonstrated the most stable performance across different datasets and the most linear scalability with respect to the number of threads. This can be attributed to a better CPU architecture, but also to a lower computing power of individual CPU cores.

The results showed that GPU is the preferred architecture for solving problems of wave tomography. Such problems can be characterized by average data volumes that can exceed CPU cache capacity, but fit into the on-board GPU RAM. Graphics processors are designed for specific data-parallel tasks with limited memory volume, which include the problem considered.

References

1. M. V. Klibanov and A. A. Timonov, *Carleman Estimates for Coefficient Inverse Problems and Numerical Applications* (De Gruyter, Berlin, 2004), doi 10.1515/9783110915549.
2. M. Birk, R. Dapp, N. V. Ruiter, and J. Becker, "GPU-based Iterative Transmission Reconstruction in 3D Ultrasound Computer Tomography," J. Parallel Distrib. Comput. **74** (1), 1730–1743 (2014). doi 10.1016/j.jpdc.2013.09.007.
3. J. Wiskin, D. Borup, M. Andre, et al., "Three-Dimensional Nonlinear Inverse Scattering: Quantitative Transmission Algorithms, Refraction Corrected Reflection, Scanner Design, and Clinical Results," J. Acoust. Soc. Am. **133** (2013). doi 10.1121/1.4805138.
4. V. A. Burov, D. I. Zotov, and O. D. Rumyantseva, "Reconstruction of Spatial Distributions of Sound Velocity and Absorption in Soft Biological Tissues Using Model Ultrasonic Tomographic Data," Acoust. Phys. **60** (4), 479–491 (2014). doi 10.1134/S1063771014040022.
5. S. Romanov, "Simulations in Problems of Ultrasonic Tomographic Testing of Flat Objects on a Supercomputer," in *Communications in Computer and Information Science* (Springer, Cham, 2020), Vol. 1331, pp. 320–331. doi 10.1007/978-3-030-64616-5_28.
6. A. V. Goncharsky and S. Y. Seryozhnikov, "Supercomputer Technology for Ultrasound Tomographic Image Reconstruction: Mathematical Methods and Experimental Results," in *Communications in Computer and Information Science* (Springer, Cham, 2019), Vol. 965, pp. 401–413. doi 10.1007/978-3-030-05807-4_34.
7. A. V. Goncharsky and S. Y. Seryozhnikov, "Three-Dimensional Ultrasound Tomography: Mathematical Methods and Experimental Results," in *Communications in Computer and Information Science* (Springer, Cham, 2019), Vol. 1129, pp. 463–474. doi 10.1007/978-3-030-36592-9_38.
8. A. V. Goncharsky, V. A. Kubyshkin, S. Y. Romanov, and S. Y. Seryozhnikov, "Inverse Problems of Experimental Data Interpretation in 3D Ultrasound Tomography," Vychisl. Metody Programm. **20** (3), 254–269 (2019). doi 10.26089/NumMet.v20r323.
9. A. Bakushinsky and A. Goncharsky, *Ill-Posed Problems: Theory and Applications* (Kluwer Academic Publishers, Dordrecht, 1994).
10. A. N. Tikhonov, A. V. Goncharsky, V. V. Stepanov, and A. G. Yagola, *Numerical Methods for the Solution of Ill-Posed Problems* (Kluwer Academic Publishers, Dordrecht, 1995).

2021, **22** (4), 322–332. doi 10.26089/NumMet.v22r421

11. A. V. Goncharsky and S. Y. Romanov, "Iterative Methods for Solving Coefficient Inverse Problems of Wave Tomography in Models with Attenuation," Inverse Probl. **33**(2), (2017). doi 10.1088/1361-6420/33/2/025003.
12. F. Natterer, "Sonic Imaging," in *Handbook of Mathematical Methods in Imaging* (Springer, New York, 2014), pp. 1253–1278. doi 10.1007/978-1-4939-0790-8_37.
13. A. Bakushinsky and A. Goncharsky, *Iterative Methods for Solving Ill-Posed Problems* (Nauka, Moscow, 1989) [in Russian].
14. R. K. Saha and S. K. Sharma, "Validity of a Modified Born Approximation for a Pulsed Plane Wave in Acoustic Scattering Problems," Phys. Med. Biol. **50** (12) (2005). doi 10.1088/0031-9155/50/12/007.
15. D. Yokoyama, B. Schulze, F. Borges, and G. Mc Evoy, "The survey on ARM processors for HPC," J. Supercomput. **75**, 7003–7036 (2019). doi 10.1007/s11227-019-02911-9.
16. V. V. Voevodin, A. S. Antonov, D. A Nikitenko, et al., "Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community," Supercomput. Front. Innov. **6** (2), 4–11 (2019). doi 10.14529/jsfi190201.
17. B. Engquist and A. Majda, "Absorbing Boundary Conditions for the Numerical Simulation of Waves," Math. Comput. **31**, 629–651 (1977). doi 10.1090/S0025-5718-1977-0436612-4.
18. B. Hamilton and S. Bilbao, "Fourth-Order and Optimised Finite Difference Schemes for the 2-D Wave Equation," in *Proc. 16th Int. Conf. on Digital Audio Effects, Maynooth, Ireland, September 2–6, 2013*, https://www.pure.ed.ac.uk/ws/portalfiles/portal/11221940/dafx2013_submission_64.pdf.
19. WaveTomography software. http://inverseproblems.ru/WaveTomography. Cited December 10, 2021.

Received
November 29, 2021

Accepted for publication
December 5, 2021

Information about the authors

Alexander V. Goncharsky — Dr. Sci., Professor, Head of Laboratory, Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, Building 4, 119991, Moscow, Russia.

Sergey Y. Romanov — Dr. Sci., Leading Researcher, Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, Building 4, 119991, Moscow, Russia.

Sergey Y. Seryozhnikov — Ph. D., Engineer, Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, Building 4, 119991, Moscow, Russia.