GPU-Accelerated Simulation of Elastic Wave Propagation

Ing. Kristian Kadlubiak, Ing. Jiří Jaroš, Ph.D., Faculty of Information Technology, Brno University of Techology

Introduction

The simulation of elastic wave propagation has many applications in ultrasonics, including the classification of bone diseases and non-destructive testing. In biomedical ultrasound in particular, elastic wave models have been used to investigate the propagation of ultrasound in the skull and brain, and to optimize the delivery of therapeutic ultrasound through the thoracic cage. Currently, there is an accurate elastic wave model written in MATLAB as part of the open-source k-Wave toolbox. However, the computational requirements of the model did not allow it to be deployed for realistic simulation cases. Therefore, we decided to create a native implementation in CUDA to accelerate the simulation.

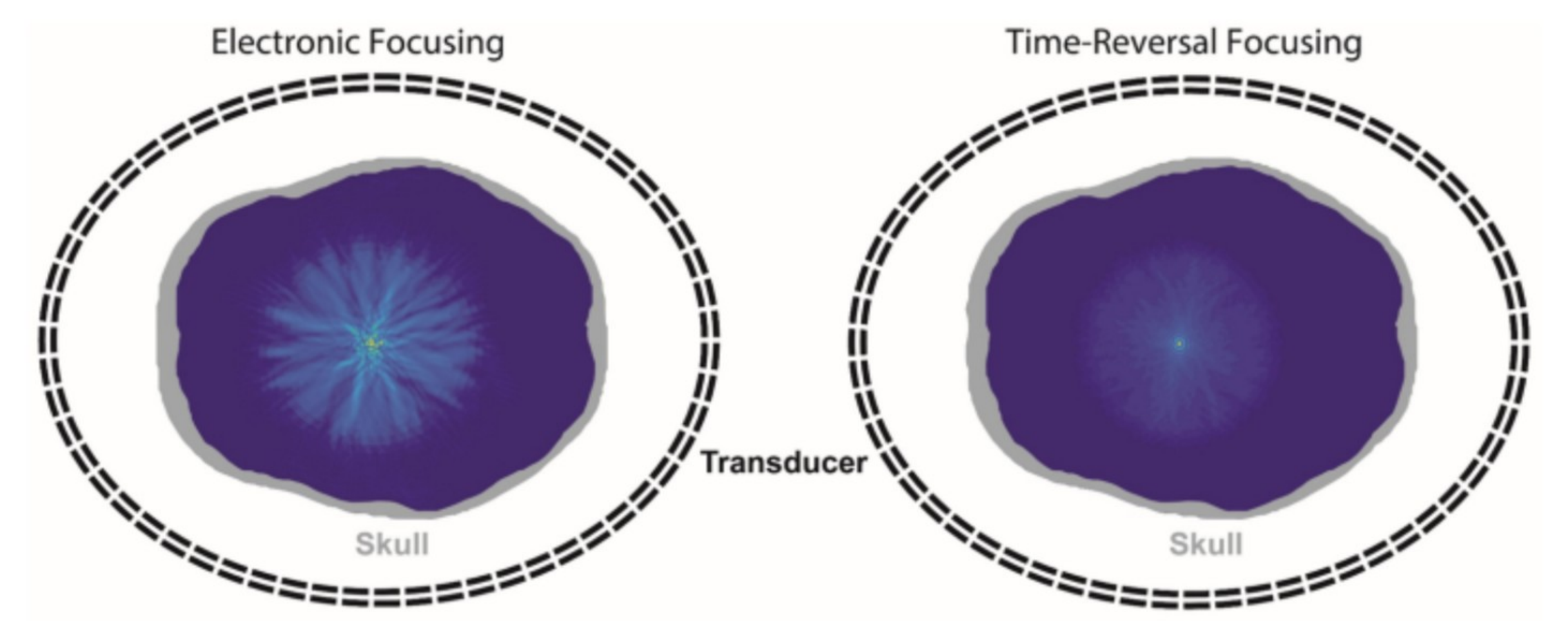


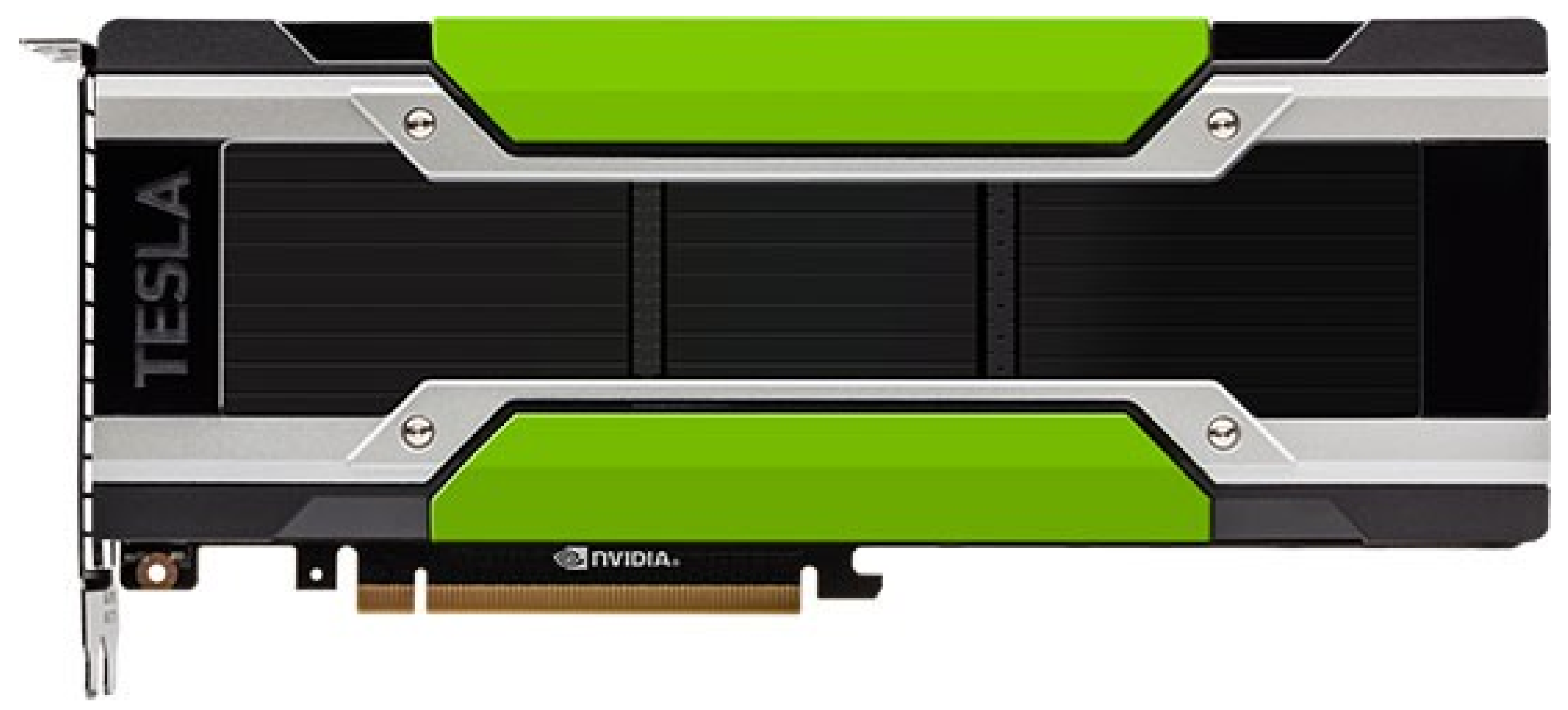
Figure 1: Comparison of electronic and simulation based time-reversal focusing.

Proposed method

The implementation of the native CUDA application follows the numerical model based on the explicit solution of coupled PDEs using the Fourier pseudospectral method. This uses the Fourier collocation spectral method to compute spatial derivatives, and a leapfrog finite-difference scheme to integrate forwards in time. An example of such a calculation to compute the derivative of the 3D stress field in x dimension is show in Eq. (1)

$$\partial_x \sigma_{xyz} = F_x^{-1}\left\{ i k_x e^{+ik_x \Delta x/2} F_x\left\{\sigma_{xyz}\right\}\right\} \tag{1}$$

Here $F\{\}$ and $F^{-1}\{\}$ are the 1D forward and inverse Fourier transforms over the x, i is the imaginary unit, k_x is the discrete set of wavenumbers in x dimension, and Δx gives the grid spacing assuming a uniform Cartesian mesh. The exponential terms are spatial shift operators that translate the output by half the grid point spacing which improves the accuracy of the model.



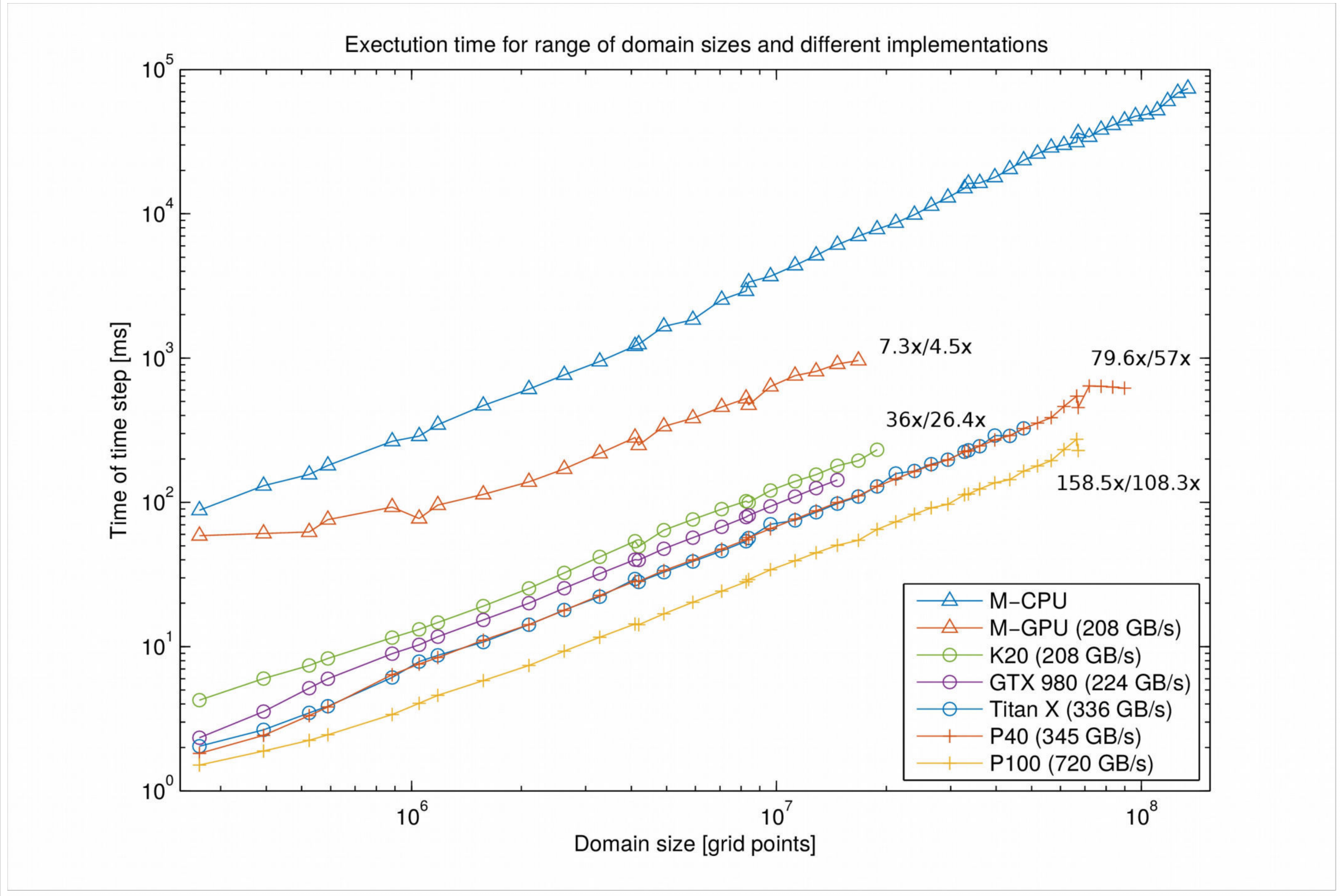


Figure 2: Exectution times for one simulation time step.

Experimental results

The performance of the developed native CUDA application was evaluated on several Nvidia GPUs based on the Kepler, Maxwell and Pascal architectures with 4 to 24 GB of on-board memory. The original Matlab implementation executed on one node of the Anselm supercomputer 2 with 2×8 Intel Haswell E5-2660 CPUs and a Kepler K20 GPU were taken as reference points. Fig. 2 shows the growth of the execution time of one simulation time step with increasing domain size from 64^3 to 512^3 grid points. Examining Fig. 2, there is a significant difference between the native CUDA application and the Matlab version caused by manual tuning of the simulation kernels, better spatial and temporal data locality, and the use of optimized FFT kernels. Interestingly, there is a remarkably low difference between different GPU architectures. Although the raw computational performance grew by an order of magnitude between Kepler and Pascal GPUs, the observed speed-up is close to 4. This suggests the code is strongly memory bound, which can be supported by the *nlogn* time complexity of the FFT, and linear complexity of other simulation kernels. The execution times copy the memory bandwidth shown in the legend of Fig. 2. To highlight the benefits of our implementation, let us mention that the performance of the native code is approx. 5.6 times higher compared to the Matlab GPU code using the same GPU. The best speed-up was achieved on Pascal P100 GPU, which was on average 108 times faster than Matlab CPU version.

Fig. 2 also reveals that some simulation domain sizes cannot be executed on particular GPUs. This is given by the memory requirements. Therefore, we derived an analytic model, Eq. (2) predicting the GPU memory consumption

$$Mem[GB] = \frac{(45+A)N_xN_yN_z + 6(N_xN_yN_z) + C + S}{1024^3/4} \tag{2}$$

where A takes a value from $<0, 8>$ depending on the heterogeneity of the medium, C represents scratch place taken by the cuda FFT library and S describes the geometry of the input source and the driving signal. This model was compared with the maximal amount of memory consumed on the Pascal P40 GPU. Fig. 3 shows a close agreement where the small difference is caused by the size of the executable and the CUDA driver.

Conclusion

This paper has presented a GPU-accelerated implementation of the elastic wave propagation in biological materials. The code has both significantly decreased the simulation time and extended the range of manageable simulation scenarios. For example, a elastic wave propagation simulation with 448^3 grid points and 4,655 time steps can now be completed in 48 minutes, instead of several hours.

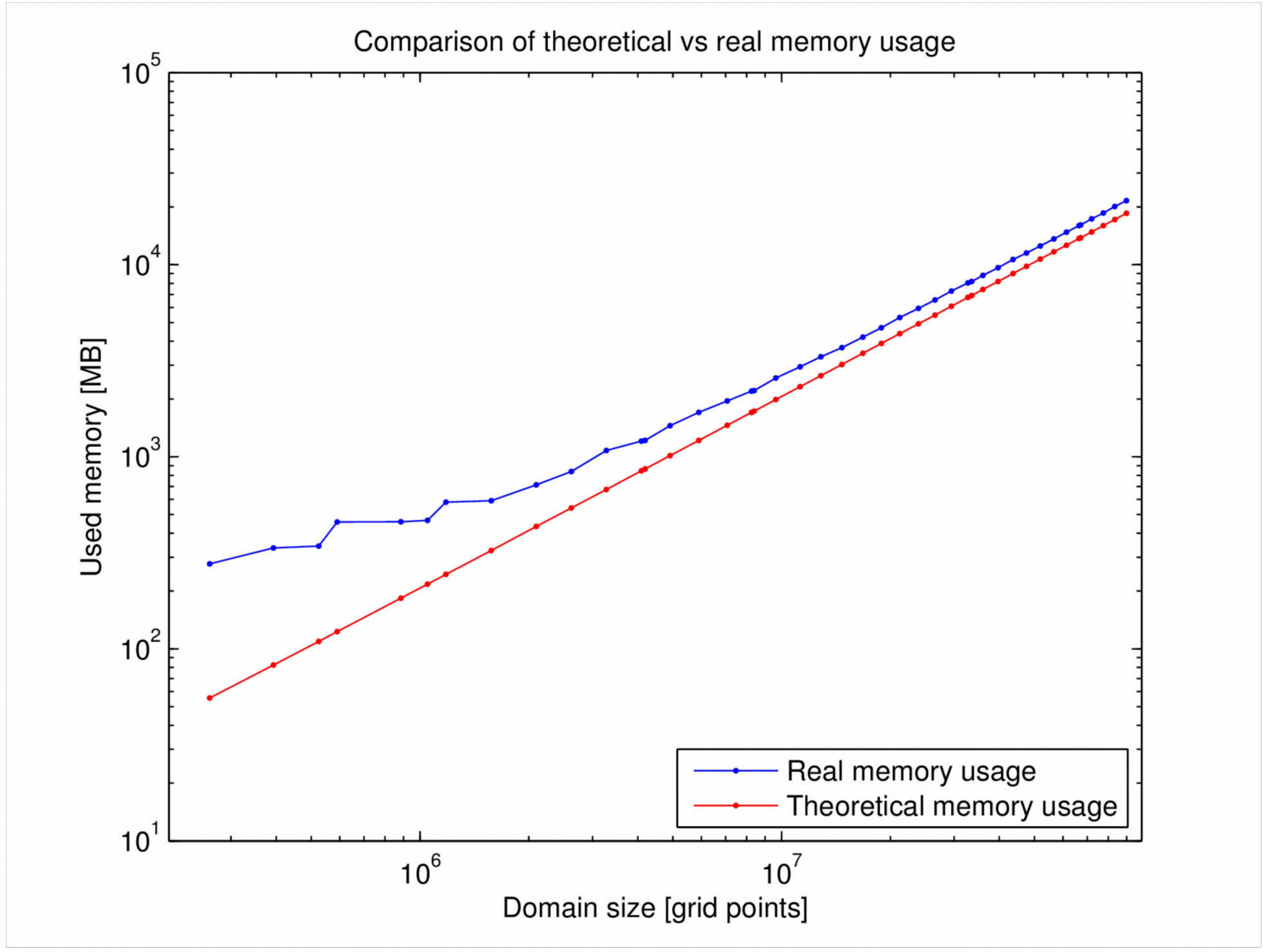


Figure 3: Comparison of the predicted and measured GPU memory consumption

Acknowledgement

This work was supported by The Ministry of Education,Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602" and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project IT4Innovations National Supercomputing Center - LM2015070". This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID pr14.