Received: 7 October 2022 | Accepted: 3 January 2023

DOI: 10.1002/pamm.202200163

Modelling transient stresses in dynamically loaded elastic solids using the Lattice Boltzmann Method

Erik Faust[1,*], **Felix Steinmetz**[1], **Alexander Schlüter**[1], **Henning Müller**[2], and **Ralf Müller**[2]

[1] Lehrstuhl für Technische Mechanik, Technische Universität Kaiserslautern, Postfach 3049, D-67653 Kaiserslautern, Germany

[2] Institut für Mechanik, Technische Universität Darmstadt, Franziska-Braun-Straße 7, D-64287, Germany

In solids subjected to transient loading, inertial effects and S- or P-wave superposition can give rise to stresses which significantly exceed those predicted by quasi-static models. It pays to accurately predict such stresses – and the failures induced by them – in fields from mining to automotive safety and biomechanics. This, however, requires costly simulations with fine spatial and temporal resolutions.

The Lattice Boltzmann Method (LBM) can be used as an explicit numerical solver for certain appropriately formulated conservation laws [1]. It encodes information about the field variables to be simulated in distribution functions, which are modified locally and propagated across a regular lattice. As the LBM lends itself to finely discretised simulations and is easy to parallelise [2, p.55], it is an intriguing candidate as a solver for dynamic continuum problems.

Recently, Murthy et al. [3] and Escande et al. [4] adopted LBM algorithms to model isotropic, linear elastic solids. We extended these algorithms using local boundary rules that allow us to model arbitrary-valued Dirichlet and Neumann boundaries. Here, we illustrate applications of the LBM for solids and the proposed additions by way of a simple numerical example – a glass pane subject to a sudden impact load.

1 Introduction

Over the past three decades, the Lattice Boltzmann Method (LBM) has become established as a fluid solver due to its computational efficiency and the relative ease with which it may be parallelised [2, p.v]. These characteristics also make the LBM attractive in solid mechanics – especially for simulations of highly transient phenomena, which require fine spatial and temporal resolutions. Eventually, developments in LB solvers for solid mechanics might also allow for simulations of fluid-structure interactions using the LBM on both sides of a solid-fluid interface.

Several publications approach solid mechanics with LB schemes – see e.g. [5–7] – via wave equation formulations for linear elastodynamics. An alternative approach – used, for example, in [3] and [4] – is to formulate LB schemes based on a moment chain of balance laws. Recently, [8] also proposed a promising quasi-static scheme based on the Navier-Cauchy equation and an artificial damping term. A challenge common to all LB schemes is the consistent implementation of boundary rules: these need to be formulated with respect to the LBM's distribution functions, and, at the same time, to be consistent with the target boundary values. Some early developments of boundary conditions for solid-mechanical LB schemes can be found in [9–11].

2 The LB algorithm

The LBM operates on a lattice consisting of lattice sites $\vec{x}$. These are arranged regularly with lattice spacing Δx in each of the principal coordinate directions. Each lattice site is connected to its neighbours via lattice links, and several lattice velocity vectors $\vec{c}_i, i \in 0, .., q-1$ cover the distance between any lattice site $\vec{x}$ and its neighbours $\vec{x} + \vec{c}_i \Delta t$ along a lattice link i in one time step Δt [2, p.94]. For ease of visualisation and validation, we only consider two-dimensional D2Q9 lattices here, in which each lattice sites is connected to its neighbours, in the coordinate directions and along the main spatial diagonals.

The LBM encodes information about the tensor fields to be modelled in distribution functions $\bar{f}_i$. At each lattice site, an array of distribution functions is introduced, with one entry per lattice velocity $\vec{c}_i$. We utilise distribution functions resulting from the second-order accurate discretisation by He et al. [12], which is denoted by the overbar $\bar{\cdot}$.

The LB algorithm performs explicit, computationally efficient operations on these distribution functions in each iteration. Firstly, the $\bar{f}_i$ at each lattice site are *locally* relaxed toward the value of the equilibrium distribution function f_i^{eq} [2, p.64], and a contribution due to a source term ψ_i is added,

$$\bar{f}_i^{col} = \bar{f}_i - \frac{\Delta t}{\bar{\tau}}(\bar{f}_i - f_i^{eq}) + \Delta t\Big(1 - \frac{\Delta t}{2\bar{\tau}}\Big)\psi_i \,, \tag{1}$$

* Corresponding author: e-mail efaust@rhrk.uni-kl.de

where $\bar{\tau}$ is the relaxation time and Δt the time step [2, p.239]. The results of this operation are then streamed to neighbouring lattice sites

$$\bar{f}_i(x + c_i\Delta t, t + \Delta t) = \bar{f}_i^{col}(x, t)\,, \tag{2}$$

which amounts to a simple *swap in memory* [2, p.66]. The desired field quantities can finally be determined from the distribution functions (and forcing terms) as moments [1], with scalar, vector, and second-order tensor moments ${}^0\Pi$, ${}^1\vec{\Pi}$, and ${}^2\mathbf{\Pi}$

$${}^0\Pi = \sum_i \bar{f}_i + \frac{1}{2}\Delta t S\,, \quad {}^1\vec{\Pi} = \sum_i \bar{f}_i\vec{c_i} + \frac{1}{2}\Delta t\vec{S}\,, \quad \text{and} \quad {}^2\mathbf{\Pi} = \sum_i \bar{f}_i\vec{c_i}\otimes\vec{c_i} + \frac{1}{2}\Delta t\boldsymbol{S}\,. \tag{3}$$

This again amounts to *local* matrix multiplications and the addition of forcing terms S, $\vec{S}$, and $\boldsymbol{S}$.

In the limit as $\Delta t \to 0$, the LB algorithm behaves as a second-order Crank-Nicolson scheme for moment chains of the form

$$\frac{d}{dt}\Big({}^N\mathbf{\Pi}\Big) + \mathrm{div}\Big({}^{N+1}\mathbf{\Pi}\Big) = {}^N\boldsymbol{S}\,, \tag{4}$$

where ${}^N\mathbf{\Pi}$ represents some N-th order tensor field, while ${}^N\boldsymbol{S}$ is the N-th order forcing term [1]. Naturally, this allows the LBM to be used as a solver for solid-mechanical balance laws, if the considered balance laws may be written in an appropriate moment chain form.

Note that the operations outlined above – a strictly local relaxation in (1), a slightly nonlocal memory swap in (2), and a local matrix multiplication in (3) – can be evaluated very efficiently and parallelised very easily. Boundary conditions, on the other hand, pose a considerable challenge: generally, a boundary rule formulation with respect to the distribution functions $\bar{f}_i$ is sought, but these rules must be chosen consistently with boundary values in the tensors appearing in (4) [2, p.155].

3 The LBM for solids

Murthy et al. [3] found that, by choosing the equilibrium distribution function

$$f_i^{eq} = w_i\Big(\rho + \frac{1}{c_s^2}c_{i\alpha}j_\alpha + \frac{1}{2c_s^4}(P_{\alpha\beta} - \rho c_s^2\delta_{\alpha\beta})(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})\Big)\,, \tag{5}$$

and the source term

$$\psi_i = \frac{1}{c_s^2}w_i c_{i\alpha}\Big(F_\alpha + \frac{\mu-\lambda}{\rho}\partial_\alpha\rho\Big)\,, \tag{6}$$

they could model the moment chain

$$\begin{aligned}
&\partial_t\rho + \partial_\alpha j_\alpha = 0\,,\\
&\partial_t j_\alpha + \partial_\beta P_{\alpha\beta} = F_\alpha + \frac{\mu-\lambda}{\rho}\partial_\alpha\rho\,,\\
&\partial_t P_{\alpha\beta} + \partial_\gamma\Big(\frac{\mu}{\rho}(j_\alpha\delta_{\beta\gamma} + j_\beta\delta_{\alpha\gamma} + j_\gamma\delta_{\beta\alpha})\Big) = 0\,,
\end{aligned} \tag{7}$$

using the LBM. Under certain smallness assumptions, this moment chain results in the Lamé-Navier equation;

$$\rho\partial_t^2 u_\alpha = (\lambda+\mu)\partial_\alpha\partial_\beta u_\beta + \mu\partial_\beta\partial_\beta u_\alpha + F_\alpha\,, \tag{8}$$

i.e. the equations of motion for the small-strain case. Roughly speaking, only terms at the leading order of smallness are considered in each equation in (7) in the derivation. This stands in slight contrast with formal small-strain theory, in which the density ρ is assumed constant and (7.1) vanishes [13, p.4]. The resulting difference in the target equations (8), however, is of third order in the displacement derivatives and assumed negligible at small strains [11].

The moments resulting from (5) and (6) are given by [3]

$$\rho = \sum_i \bar{f}_i\,, \quad j_\alpha = \sum_i \bar{f}_i c_{i\alpha} + \frac{1}{2}\Delta t S_\alpha\,, \quad \text{and} \quad P_{\alpha\beta} = \sum_i \bar{f}_i c_{i\alpha}c_{i\beta}\,.$$

Here, the Poisson stress tensor $\boldsymbol{P}$ accounts for the Poisson ($\nu = 0.25$) part of the material law [11]. The remainder of the material law – due to the difference between the Lamé parameters λ and μ, and containing a dilatational contribution – is accounted for in the vector source term $\vec{S}$. Murthy et al. embed information about this dilatational contribution in the

gradient of the density $\mathrm{grad}(\rho)$, which is available via the zeroth moment [3]. This does away with the need to compute the displacement $\vec{u}$ via integration during the iterative procedure. The algorithm does, however, involve a finite difference evaluation to compute this gradient, which negatively impacts the computational efficiency and the stability of the method.

As we found in a recent preprint [11], the Cauchy and Poisson stress tensors can be conveniently calculated from one another via

$$P_{\alpha\beta} = -\sigma_{\alpha\beta} + (\lambda - \mu)\partial_\gamma u_\gamma \delta_{\alpha\beta}\,,$$

where information about the displacement divergence can again be obtained from the density via the first-order accurate kinematic relation

$$\partial_\alpha u_\alpha \approx \frac{\rho_0 - \rho}{\rho_0}\,.$$

Finally, Dirichlet

$$j(\vec{x},t) = j^*(\vec{x},t) = \rho(\vec{x},t)\partial_t u^*(\vec{x},t), \quad \vec{x} \in \partial\Omega\,,$$

and Neumann boundary conditions

$$P_{\alpha\beta}(\vec{x},t) = P^*_{\alpha\beta}(\vec{x},t) = -\sigma^*_{\alpha\beta}(\vec{x},t) + (\lambda - \mu)\partial_\gamma u_\gamma(\vec{x},t)\delta_{\alpha\beta} \quad \vec{x} \in \partial\Omega\,.$$

are reformulated with respect to the tensors appearing in the moment chain (7). In [11], the latter conditions are rotated into a coordinate system that is aligned with the boundary, allowing the values of certain stress components to be identified directly from the traction vector acting on the boundary. Boundary conditions can then be imposed using one of the many boundary handling schemes available for the LBM – as a simple first approach, we utilised the local bounce-back [2, p.175] and anti-bounce-back [2, p.200] rules.

4 Glass pane under sudden loading

To exemplify a possible application of the solid-mechanical LBM, we consider a simple model of a glass pane under sudden loading (see figure 1). We assume a Young's modulus of $E = 70\mathrm{GPa}$, Poisson's ratio of $\nu = 0.24$, and density $\rho = 3000\,\mathrm{kg/m^3}$, which is typical for a range of glasses [14, p.197]. The chosen values amount to Lamé parameters $\mu \approx 2.82\mathrm{GPa}$ and $\lambda \approx 2.61\mathrm{GPa}$, as well as an S-wave speed of $c_s \approx 3.07 \times 10^3\,\mathrm{m/s}$ and a P-wave speed of $c_d \approx 5.24 \times 10^3\,\mathrm{m/s}$. We subject a glass pane with width $w = 0.2\mathrm{m}$, height $h = 0.02\mathrm{m}$, and the chosen material constants to a sudden load of $t^* = 10\mathrm{MPa}$ at time $t_0 = 0\mu\mathrm{s}$. The load is applied to the lower surface, at $-0.01\mathrm{m} \leq x_1 \leq 0.01\mathrm{m}$, and removed at time $t_1 = 10\mu\mathrm{s}$. The left and right edges of the material domain at $x_1 = -0.1\mathrm{m}$ and $x_1 = 0.1\mathrm{m}$ are fixed throughout the simulation, and we assume a plane strain state. All simulations are run until $t_2 = 0.2\mathrm{ms}$.

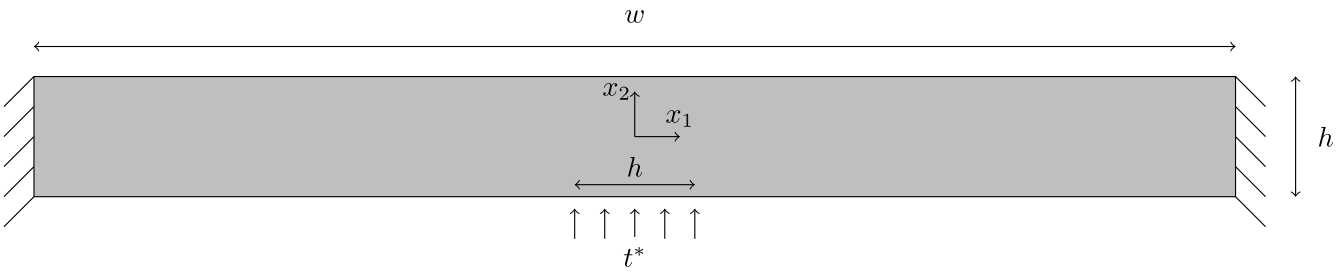

Fig. 1: Glass pane subjected to a sudden load.

We perform LB and Finite Element (FE) simulations to model the response of the glass pane to this highly transient load case. In all LB simulations, a relaxation time of $\bar{\tau} = 0.55\Delta t$ is chosen. We use FEniCS [15] to perform the FE calculations, utilising an implicit constant acceleration Newmark method. The elements are rectangular and the shape functions linear.

The left plot in figure 2 shows the vertical displacements u_2 computed by the LBM and the FEM throughout the simulation, at several points in the glass pane, for one example discretisation[1]. Similarly, second plot in figure 2 shows the von Mises stresses at a subset of these points, for $t \in [0, 50]\mu\mathrm{s}$. We chose only to plot this subinterval in time because the high-frequency stress fluctuations – caused by the reflection of P-waves between both edges of the glass pane – are difficult to discern otherwise, and the highest stress peaks occur within this interval.

[1] The discretisation used here corresponds to the second-finest in the convergence study below. Details of the Finite Element computation are outlined below.

The LBM seems to capture the transient displacement overshoot roughly as well as the FEM. Generally, the LBM appears to produce solutions that are slightly delayed when compared to the FE reference, but both simulation methods yield very similar peak displacement values. Similarly, the FEM and the LBM predict very similar peak values for the von Mises stress, and the stress evolution determined by both simulation methods matches qualitatively and quantitatively. Here, the LBM appears to produce a slightly smoother stress profile, and a slight delay is again apparent relative to the FE reference solution. The considerable transient peak stresses (in the range of $\sigma^{\mathrm{vM}} \approx 12\mathrm{MPa}$) in the glass pane are apparent in both solutions. At $x_1 = 0.0\mathrm{m}$, the arrival of P-waves accounts for the sudden spike in σ^{vM}, while at $x_1 \approx 0.094\mathrm{m}$, the stresses reach their critical value more gradually, as the clamped edge of the glass pane arrests the displacement overshoot.

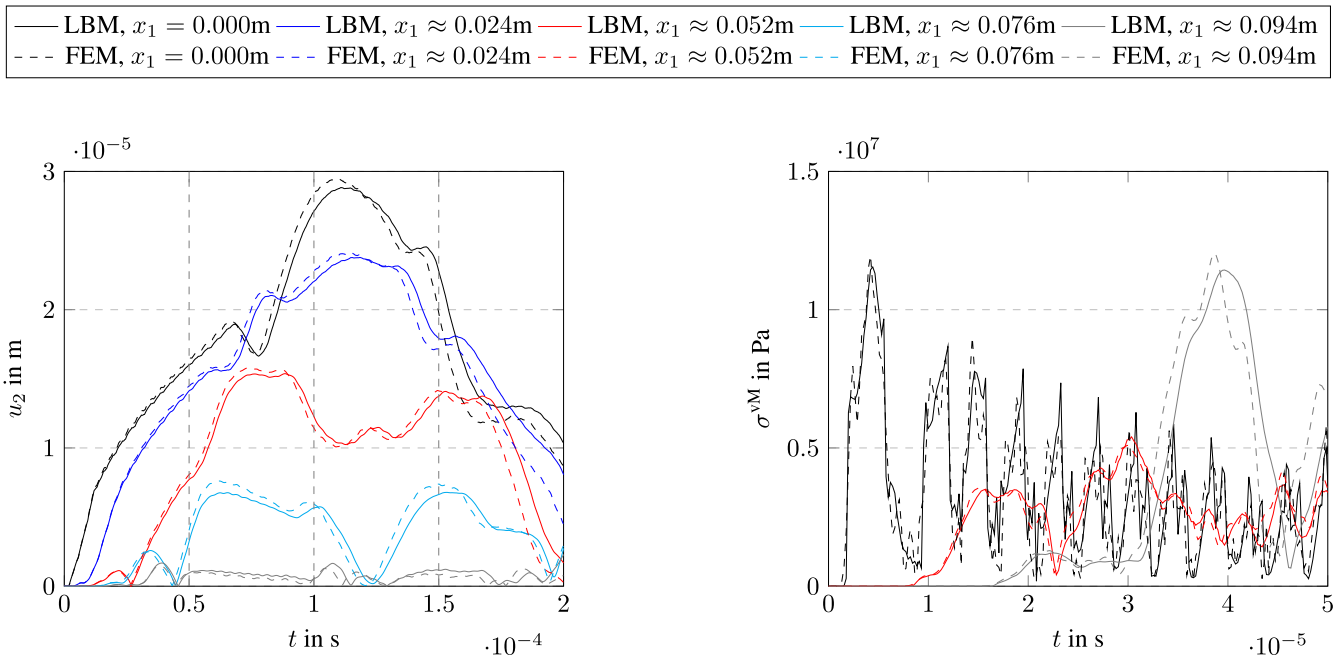

Fig. 2: Vertical displacements (left) and von Mises stresses (right) postprocessed from LBM (continuous lines) and FEM (dashed lines) simulations of a suddenly loaded glass pane, for several points along $x = 0\mathrm{m}$, with $\Delta x = 0.8 \times 10^{-3}\mathrm{m}$. The grey, dashed lines in the displacement plot indicate the time values for which convergence studies are evaluated below. The stress plot is truncated at $t = 50\mu\mathrm{s}$, as the high-frequency oscillations are hard to discern otherwise.

Furthermore, a mesh refinement study is performed for the LB simulation, with $\Delta x \approx \{5\,, 2.22\,, 1.18\,, 0.8\,, 0.61\} \times 10^{-3}\mathrm{m}$. In each case, the time step Δt is determined from the lattice spacing via $\Delta t = {}^{\Delta x}/_{\sqrt{3} c_s}$. In the reference FE simulation, a time step of $\Delta t \approx 1.15 \times 10^{-7}\mathrm{s}$ is chosen. This FE discretisation is analogous to the finest among the discretisations used for the LBM.

Note that the chosen time step size is typical for FE simulations of highly transient phenomena: time steps in transient FE calculations are typically determined to meet the CFL condition via $\Delta t = \alpha_c {}^{l_c}/_{c_c}$, with critical element length l_c, and wave speed $c_c = \sqrt{{}^{E}/_{\rho_0}}$ [16, p.335]. α_c is a step reduction factor typically chosen as $\alpha_c \approx 0.9$ for well-behaved simulations, but values of as low as $\alpha_c = 0.25$ [17] or $\alpha_c = 0.162$ [18] are common for impact and shock wave problems. With $l_c = \Delta x$, the chosen time step amounts to a reduction factor of $\alpha_c \approx 0.909$. Thus, when compared to transient Finite Element simulations, it is not necessary to chose an abnormally low time step to use the LBM for transient solid modelling[2].

In the absence of an analytical solution, the FE simulation with the finest discretisation is used as a benchmark estimate, and the convergence study is carried out with respect to it. After the simulations runs, we compute the (normalised) Euclidean norm of the error in the displacement field e at a range of points in the domain[3] obtained from the LB simulations at times $t_{\mathrm{comp}} = \{50, 100, 150\}\mu\mathrm{s}$, relative to the reference FE solution. The resulting error estimates are used for a convergence study, the results of which can be found in the double-logarithmic plot in figure 3. There, the norm of the error at several times $t \in t_{\mathrm{comp}}$ during the simulation is displayed over the lattice spacing. For reference, gradients corresponding to linear and quadratic convergence rates as well as the superlinear convergence rate of the FEM reference simulation are also shown[4].

[2] We also attempted to run the example considered here using the FEM and an explicit central difference time integration scheme. The simulations became unstable when run with the chosen time step reduction factor; a value of $\alpha_c \approx 0.45$ seems to be necessary for stable explicit FE simulations. More extensive investigations are necessary to verify this observation.

[3] For all discretisations, the error is evaluated at points spaced approximately $0.02\mathrm{m}$ apart along the glass pane. The error is normalised with the number of points used in the error computation.

[4] As the convergence rate of the FEM reference simulations is evaluated with respect to FEM simulation with the finest discretisation, we only show the convergence rate here (rather than the error values, which are meaningless). For the FEM, the best-case convergence rate for $t = 50\mu\mathrm{s}$ is used for the plot in figure 3.

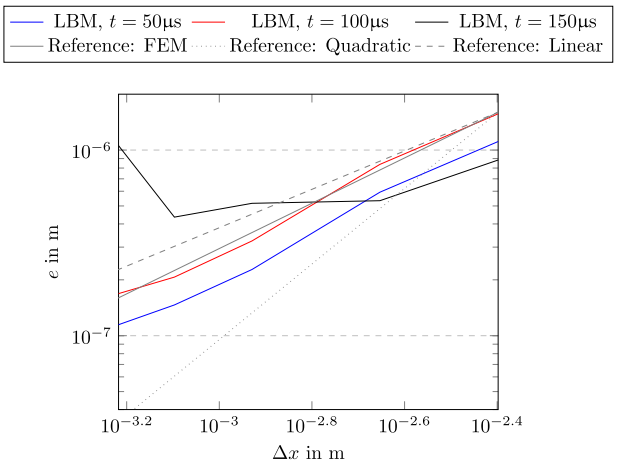

Fig. 3: Convergence study for several times t during the LBM simulation of a suddenly loaded glass pane. The Euclidean norm of the error e (normalised with the number of evaluation points) is plotted over the lattice spacing Δx.

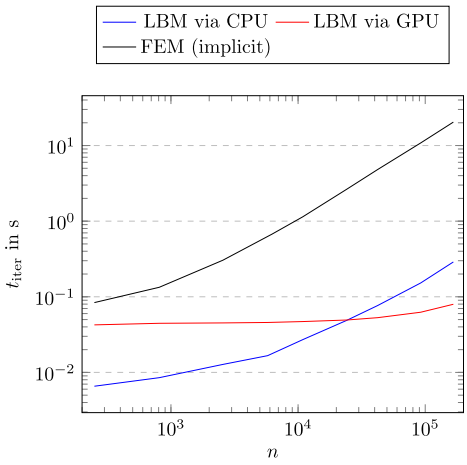

Fig. 4: Time cost of one iteration over number of lattice sites/elements for the LBM (on CPU and GPU) and the FEM.

For $t = 50\mu s$ and $t = 100\mu s$, the LBM converges superlinearly, with a very similar rate to that exhibited by the FEM. For $t = 150\mu s$, the LBM solution diverges with increasing resolution due to the appearance of instabilities. These seem to be more closely tied to the time step number than the absolute value of the time t, meaning that instabilities are exhibited at lower values of t for finer resolutions. In part, the artificial forcing term, which is computed via finite differences and accounts for the non-Poisson part of the material law, might be the cause of these instabilities. However, the simulation remains stable for sufficiently long to capture the transient displacement overshoot (and the peak stress and displacement values) for all discretisations.

Overall, the LBM produces satisfactory results in this example: the simulations predict the displacement evolution well quantitatively and qualitatively. The convergence rates exhibited for different times during the simulation are only slightly lower than those achieved with the FEM. The simulations eventually become unstable, but remain stable for long enough to capture the behaviour of interest.

The LBM algorithm described above is implemented in a vectorised fashion using an in-house python package. A CPU and a GPU version are implemented using NumPy and CuPy respectively. We do not make use of explicit parallelism such as MPI parallelism (see e.g. [19]) to date. In figure 4 the necessary computation time required for a single time step t_{iter} is shown over the number of lattice sites n (nodes in case of the FEM) [5]. The NumPy implementation outperforms the FEM implementation by slightly more than one order of magnitude regardless of the problem size n and scales similarly to the FEM with n. Above the threshold of roughly $n \approx 10^4$, the GPU version is faster, since the problem size is large enough for the algorithm to benefit from the implicitly parallel execution of commands on the GPU. For small problem sizes $n < 2 \times 10^4$, the one-time overhead of the GPU dominates the necessary computation time and in such cases, t_{iter} is nearly invariant over the problem size. Overall, the LBM performs promisingly and the straightforward execution on the GPU leaves room for more extensive simulations.

5 Limitations

While the results are generally encouraging, the LBM presented above is still plagued by considerable limitations. Firstly, the algorithm is not sufficiently stable for materials with a Poisson's ratio deviating too far from $\nu = 0.25$ – the simulation outlined above could not, for example, have been run successfully with material parameters corresponding to steel, i.e. $\nu \approx 0.3$. In part, this instability is caused by the source term computed via finite differences in (7). The use of this source term also interferes with the runtime advantages offered by the LBM, as it introduces non-locality into the collision step in (1) and interferes with the LBMs attractive parallelisation characteristics. This issue could perhaps be overcome by incorporating the remainder of the material law via the third moment of the equilibrium distribution function and making the source term redundant.

Furthermore, the time step Δt used by the LB algorithm is inextricably linked to the lattice spacing Δx and the S-wave speed $c_s = \sqrt{\mu/\rho}$. This means that the use of a finer discretisation in space always necessitates the use of a proportionally smaller time step; and the high wave speeds in relevant engineering materials mean that very fine temporal resolution are required generally. This issue is unavoidable (with the LBM or the FEM) if wave fronts are to be resolved accurately. If only quasi-static/quasi-dynamic phenomena (featuring, perhaps, dynamic overshoots in the displacement but no wave phenomena) are of interest, however, this fine time step simply constitutes a waste of computational resources. This second issue is

[5] The LBM and FEM benchmarks are performed on an Intel Xeon E5-1650 v3 and a Nvidia K40M. In all cases, 10 simulation runs are performed with 300 time steps each to compute the computational cost.

addressed via mass scaling methods in state of the art explicit FEM software; perhaps a similar methodology could provide a workaround for quasi-static simulations using the LBM.

6 Conclusions

Despite the limitations, the early results are promising: the LBM produces roughly the same the results as the FEM, with a convergence rate that is only slightly lower than that exhibited by the reference FEM simulations. Even though part of the material law is (suboptimally) implemented via a forcing term and the bounce-back type boundary rules are not adapted to highly transient behaviour, the simulations remain stable for long enough to capture the relevant transient overshoots and peak stresses with satisfactory accuracy.

Furthermore, in the example considered here, our in-house LB simulation software (developed in Python) outperforms the implicit constant acceleration Newmark method (implemented in FEniCS) by more than an order of magnitude. With the setup used here, we thus obtain comparable results (perhaps with slightly lower accuracy) in less than one tenth of the simulation time. A comparison with state-of-the-art explicit FEM software would be interesting – preliminary results in this investigation seem to indicate that an FEM scheme with explicit time integration requires smaller time steps than the LBM to obtain useful results for the highly transient example considered here.

The results seem to us sufficiently encouraging to motivate further research into Lattice Boltzmann Methods for solid mechanics. One crucial topic for future research is the extension of the stable domain to a wider range of material parameters, so that engineering materials like steel or concrete can be modelled. Boundary rules also require further attention, as they do not deal well with highly transient loading and may give rise to errors and instabilities. Finally, it may prove interesting to consider more than simple linear elastic material behaviour and model nonlinear material behaviour or damage caused by highly transient effects – a field in which computational efficiency is crucial.

Acknowledgements We are very grateful to Ramses Sala for his helpful comments.

The authors gratefully acknowledge the funding by the German Research Foundation (DFG) within the project 423809639. Open access funding enabled and organized by Projekt DEAL.

References

[1] G. Farag, S. Zhao, G. Chiavassa, and P. Boivin, Physics of Fluids **33**(3), 037101 (2021).
[2] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, The Lattice Boltzmann Method: Principles and Practice, 1st edition (Springer, Heidelberg, 2017).
[3] J. Murthy, P. Kolluru, V. Kumaran, and S. Ansumali, Communications in Computational Physics **23**(08) (2017).
[4] M. Escande, P. K. Kolluru, L. M. Cléon, and P. Sagaut, CoRR **abs/2009.06404** (2020).
[5] S. Marconi and B. Chopard, International Journal of Modern Physics B **17**(01n02), 153–156 (2003).
[6] G. S. O'Brien, T. Nissen-Meyer, and C. J. Bean, Bulletin of the Seismological Society of America **102**(3), 1224–1234 (2012).
[7] A. Schlüter, S. Yan, T. Reinirkens, C. Kuhn, and R. Müller, PAMM **20**(1), e202000119 (2021).
[8] O. Boolakee, M. Geier, and L. De Lorenzis, A new lattice boltzmann scheme for linear elastic solids: periodic problems, 2022.
[9] A. Schlüter, C. Kuhn, and R. Müller, Computational Mechanics **62**(5), 1059–1069 (2018).
[10] A. Schlüter, H. Müller, and R. Müller, Archive of Applied Mechanics(Aug) (2022).
[11] E. Faust, A. Schlüter, H. Müller, and R. Müller, Dirichlet and neumann boundary conditions in a lattice boltzmann method for elastodynamics, 2022.
[12] X. He, S. Chen, and G. D. Doolen, Journal of Computational Physics **146**(1), 282–300 (1998).
[13] V. B. Poruchikov, Methods of the classical theory of elastodynamics, 1st edition (Springer, Heidelberg, 1993).
[14] A. Varshneya and J. Mauro, Fundamentals of inorganic glasses (Elsevier, Netherlands, January 2019).
[15] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, Archive of Numerical Software **3**(100) (2015).
[16] T. Belytschko, W. Liu, B. Moran, and K. Elkhodary, Nonlinear Finite Elements for Continua and Structures (Wiley, 2014).
[17] G. Rio, A. Soive, and V. Grolleau, Advances in Engineering Software **36**(4), 252–265 (2005).
[18] C. Lim, V. Shim, and Y. Ng, International Journal of Impact Engineering **28**(1), 13–31 (2003).
[19] L. Pastewka and A. Greiner, HPC with Python: An MPI-parallel implementation of the Lattice Boltzmann Method (Universitätsbibliothek Tübingen, 2019).

