Optimization of a distributed finite difference code for simulation of acoustic wave propagation

This bachelor thesis project (G2E-level, 10 weeks, 15 credits) will be done at the Division of Scientific Computing at Uppsala University. The project will be supervised by Vidar Stiernström, PhD student in Scientific Computing. Suggested subject reader is Jarmo Rantakokko, associate professor in Scientific Computing.

Background: Finite difference methods for simulation of acoustic waves

Computer simulations are a vital tool for understanding our world, predicting the behavior of complex systems, and in guiding design processes. Many physical processes can compactly be described via partial differential equations (PDEs), where the unknown physical quantities of the medium (for instance pressure or particle displacement) are related via their rate of change. Typically the unknowns are functions of the spatial dimensions of the medium (e.g (x, y, z) in three dimensions) and of time. However, it is in general not possible to obtain an analytic (exact) solution to the PDE, and one therefore turns to approximations computed via numerical methods.

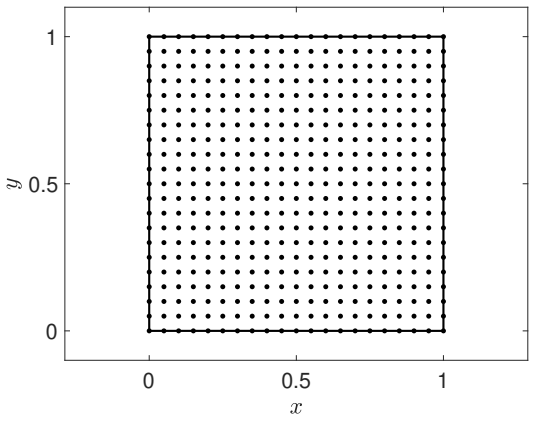
In this project we will focus on improving the performance of a code used for the simulation of acoustic waves (e.g sound). For simplicity we will restrict ourselves to a acoustic waves in a two-dimensional (2D) domain. The extension to 3D is straightforward but will not be considered in the project, due to time restrictions. The propagation of acoustic waves in a 2D domain Ω can be described by the following set of PDEs

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} &= -\frac{1}{\rho}\frac{\partial p(x,y,t)}{\partial x} && (x,y) \in \Omega, \quad t \geq 0 \\ \frac{\partial v(x,y,t)}{\partial t} &= -\frac{1}{\rho}\frac{\partial p(x,y,t)}{\partial y} && (x,y) \in \Omega, \quad t \geq 0 \\ \frac{\partial p(x,y,t)}{\partial t} &= -\kappa\left(\frac{\partial u(x,y,t)}{\partial x} + (\frac{\partial v(x,y,t)}{\partial y}\right) && (x,y) \in \Omega, \quad t \geq 0 \end{aligned} \tag{1}$$

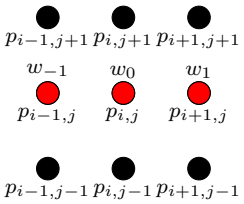
where the unknowns u, v are the velocities in the x- and y-direction and the unknown p is the pressure. The density ρ and incompressibility κ of the medium are known parameters.

As a first step in obtaining an approximate solution to (1) the domain Ω is represented as a grid, with a fixed number of equidistantly spaced points per coordinate direction. As an example, see Figure 1a where a grid for the square domain $\Omega = [0, 1]^2$ is constructed using 21 points in each coordinate direction. Assuming we use N points per coordinate direction, each of the unknowns u, v, p are now represented on the grid as N^2 element arrays where e.g $p_{i,j} = p(x_i, y_j, t) \quad i, j = 0 \ldots N$. Next, the spatial derivatives are approximated using a finite difference method (FDM). In FDMs derivatives are approximated by applying stencils to each grid point, i.e weighted sums taken over neighboring grid points. For instance, the partial derivative of p in the x-direction, evaluated at grid point (x_i, y_j) using a 3-point stencil is approximated as $\frac{\partial p(x_i,y_j)}{\partial x} \approx \sum_{k=-1}^{k=1} w_k p_{i+k,j}$, where w_k are the stencil weights. See Figure 1b. The shape of the stencils depend on the approximated derivative, and the order of accuracy of the approximation (where more accurate approximations typically utilize wider stencils). The above procedure (the introduction of a computational grid, and approximating spatial derivatives by stencils) takes (1) from a system of PDEs with 3 unknowns u, v, p to a system of coupled ordinary differential equations (ODEs) with a total of $3N^2$ degrees of freedom (DOF). Given an initial condition (i.e the initial state of the medium), an approximate solution to the system of ODEs (and thereby to (1)) can then be obtained at a future time t using a suitable time integration method.

For a given order of accuracy of the numerical method, the accuracy in the approximate solution is related to the spatial accuracy which grows as $\mathcal{O}(N^D)$ where D is the dimensionality of the problem. It is then clear that for fine resolutions in 2D, or even moderate resolutions in 3D, there is a need for efficient methods suitable for distributed memory systems. For wave dominated problems, high-order accurate FDMs have long been known to be very efficient, allowing for a considerable



(a)



(b)

Figure 1: (a) Grid representation of the domain $\Omega = [0.1]^2$; (b) 3-point stencil approximating $\frac{\partial p}{\partial x}$ at grid point (x_i, y_j). The points in red are included in the stencil.

reduction in degrees of freedom for a given error tolerance [5]. In contrast to many other numerical methods requiring matrix-vector calculations, FDMs are also naturally implemented using the stencil-based approach described above, allowing for a reduced memory footprint. Parallelizing a FDM is done by simply partitioning the domain over the processes and communicating the data needed for the stencil applications. Moreover, the overhead in communication can in many cases largely be overlapped by computations allowing for good parallel scaling. For this scaling to be relevant, the serial performance of the implemented finite difference method is however of great importance, which brings us to the present project.

Objective

The starting point for this bachelor thesis project is a FDM implemented in the C/C++ programming languages. The code is parallelized for distributed memory computations using the scientific software library PETSc [2, 3, 4, 1] which also provides time integration routines. The spatial discretization is based on a high-order accurate FDM developed at the Division of Scientific Computing, Uppsala University. Although the current implementation is functioning in the sense that it produces correct results, many questions regarding its efficiency are yet to be addressed. The main objective of the project is therefore to apply the techniques studied in the course High Performance Programming (10 credits) in order to investigate and optimize the performance of the code. For instance, no efforts have been made to improve data access patterns, e.g. via cache blocking. In addition, the data structures and API:s of PETSc are mostly tailored towards matrix-vector computations, and not matrix-free stencil computations. It is therefore not clear if other data structures would be more beneficial to employ. Typically, stencil computations tend to be memory bound [6] and correct data layout and accessing patterns is therefore crucial for performance. Other questions related to compiler optimizations, such as function inlining and auto vectorization of hot loops also need to be investigated. If time permits one may also investigate whether further performance gains may be obtained by utilizing shared memory parallelization via OpenMP on node local data (avoiding the cost of message passing). The objectives of the project are summarized below

- Profile existing code to identify performance bottle necks.
- Investigate and implement improvements to memory usage, such as cache efficiency and data layout.

- Investigate compiler optimizations of performance critical parts, focusing on auto vectorization and function inlining of hot loops.
- (Optional) Investigate if shared memory parallelization using OpenMP can be utilized to increase performance of node-local computations.
- Profile optimized code to demonstrate improvements (if any).

Methodology

- Meetings will be held by appointment or on urgent request by the student or supervisor(s).
- The supervisor will provide the existing unoptimized software, and access to the UPPMAX compute center.
- The thesis will be written in English using LaTeX.
- The software produced will be developed using the C/C++ programming language(s).
- All documents and software produced during this work will be archived in a GitHub repository, and referenced in the thesis.

Relevant courses

- High Performance Programming
- Scientific Computing and Calculus
- Computer Architecture

Delimitations

We stress that the emphasis on the present project is mainly on optimization techniques for improved serial performance, and not on distributed parallel computations. The project will utilize functionality from the PETSc software library as a starting point, but if this becomes a hindrance for the completion of the project, PETSc will be omitted.

Time plan

The equivalent of 10 full-time weeks will be spent on this work, beginning in week 12 of 2021. We aim at completion by mid June 2021, upon a public presentation of the results and revision of the thesis according to the comments by the audience and the reviewer and subject reader. The time will be spent according to the following time plan:

1. Background study on FDMs (0.5 weeks).
2. Background on existing code base, including PETSc (1 weeks).
3. Background study on optimization techniques for FDMs (1 weeks).
4. Profiling, implementation and evaluation (5 weeks).
5. Writing of the thesis (2.5 weeks).

References

[1] Shrirang Abhyankar, Jed Brown, Emil M Constantinescu, Debojyoti Ghosh, Barry F Smith, and Hong Zhang. Petsc/ts: A modern scalable ode/dae solver library. *arXiv preprint arXiv:1806.01437*, 2018.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. `https://www.mcs.anl.gov/petsc`, 2019.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.14, Argonne National Laboratory, 2020.

[4] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[5] Heinz-Otto Kreiss and Joseph Oliger. Comparison of accurate methods for the integration of hyperbolic equations. *Tellus*, 24(3):199–215, 1972.

[6] Mathias Louboutin, Michael Lange, Felix J. Herrmann, Navjot Kukreja, and Gerard Gorman. Performance prediction of finite-difference solvers for different computer architectures. *Computers & Geosciences*, 105:148–157, 2017.