Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 1542–1551

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Model-Driven Choice of Numerical Methods for the Solution of the Linear Advection Equation

Andrea Arteaga[1]*, Oliver Fuhrer[1], Torsten Hoefler[2], and Thomas Schulthess[2,3]

[1] Federal Institute of Meteorology and Climatology MeteoSwiss, Zurich, Switzerland
[2] ETH Zurich, Zurich, Switzerland
[3] Swiss National Supercomputing Centre CSCS, Lugano, Switzerland

Abstract

Designing a partial differential equations solver is a complex task which involves making choices about the solution algorithm and its parameters. Such choices are usually done on the basis of personal preference or numerical experiments, which can introduce significant bias on the selection process. In this work we develop a methodology to drive this selection process towards the optimal choices by modelling the accuracy and the performance of the solution algorithm. We show how this methodology can be successfully applied on the linear advection problem. As a result, the selection can be optimally performed with a much lower investment on the development of high-performance versions of the solvers and without using the target architecture for numerical experiments.

Keywords: accuracy modeling, performance modeling, optimization, wave propagation, linear advection, Finite Difference, Runge-Kutta, GPU

1 Introduction

Partial differential equations (PDEs) are ubiquitous in modern science and their solution is the primary goal of many high-performance computing (HPC) applications. Due to the abundance of numerical methods for the solution of PDEs, choosing an algorithm with an optimal trade-off between the computational cost and the quality of the solution can be a difficult task, and scarce literature exists on the topic of making such an opportune choice [1].

Some characteristics of the numerical methods make them more suited for certain classes of problems. Elliptic problems are, e.g., often solved using the Finite Element Method (FEM), due to its suitability for time-independent problems. Conservation equations are frequently tackled using the Finite Volume Method (FVM) [2] or the Discontinuous Galerkin Method (DG) [3], the latter allowing for simpler increase of the convergence order. However, many exceptions to this simple categorization exist [4], and Finite Difference methods are applied for many types of PDEs and still present in many simulation codes. Consequently, one has a broad choice of algorithms to solve PDEs, and each solution method comes with a broad choice of variants and

*Corresponding author: Andrea.Arteaga@meteoswiss.ch parameters: The formulation of the equations, the space discretization and the time marching scheme are just some examples of factors which influence the development of a solver [5, 6].

In this article we choose a typical problem which arises in weather and climate simulations: advection. Advection is one of the key processes that happen in the atmosphere and its accurate resolution is of primary importance. Moreover, the advection velocity is one of the main limiting factors to the time step size due to stability and accuracy constrains [7], which affects the simulation runtime. In this study, we discretize the linear advection problem using FD schemes and use accuracy and performance modeling to select the discretization order, time marching scheme and Courant number[1] that minimize the runtime under a given accuracy constraint.

The present work is a first step towards the development of a methodology to solve such optimization problems on a broader range of PDEs and solvers. The encouraging results presented here show that expensive tests performed using actual implementations on specific computer hardware for the purpose of selecting the best method can be replaced by modeling, with much lower cost due to implementation of a high-performance version of the solvers to be tested — our model requires simple prototypes without any performance optimization — and to the node-hours required to perform these tests on the target architecture.

Our main contributions are the following:

- The development of an accuracy model for linear Finite Difference solvers, with the ability to provide solid predictions of the L^2 error norm and to perform stability analysis;
- The application of a performance model for such solvers, with the ability to accurately predict the runtime of simulations on different modern computer architectures;
- The formulation and solution of an optimization problem with the goal of minimizing the cost of a simulation under the constraint of a maximum error, by considering a set of solvers and their configuration parameters; and
- The execution of experiments on parallel computers to validate our methodology.

This work was supported by the Swiss National Science Foundation under Sinergia grant CRSII2_154486/1 crClim.

1.1 Related work

Previous work which specifically addressed optimization of Finite Difference advection solvers focused mainly on stability analysis. A review of early work on stability is given by Lax and Richmyer [8]. Recent work on the optimization of these schemes has been performed by Baldauf [7], where maximum CFL numbers for a large set of linear schemes are derived, with the assumption that larger CFL numbers will provide comparable accuracy at a lower cost. Novel Runge-Kutta schemes with a large stability region have been proposed by Allampalli et al. [9]. The present work investigates more accurately the cost of such schemes in terms of a performance measure and to give an understanding of the accuracy-performance relationship. Optimized Runge-Kutta schemes with respect to accuracy, stability and performance have been dervived with FD space discretization by Calvo et al. [10], with pseudospectral methods by Tselios and Simos [11] and with DG by Toulorge and Desmet [12].

Work focused on accuracy estimation and accuracy-constrained optimization has been performed by Lockard et at. [13], where the concept of the PPW (Points Per Wavelenght) quantity is introduced and minimal PPW requirements to reach a certain accuracy is derived for multiple solvers. Also, Zingg [14] shows how to derive new Finite Difference / Runge-Kutta schemes for linear advection, with a lower order than theoretically possible with the same cost, but with accuracy features specifically tailored for the relevant wavelengths. Our research alleviates the

[1]The Courant number is the ratio of the product of the time step size and the velocity to the grid spacing

limitation of these works in the topic of the performance prediction and assessment: the cited articles use as a cost model a simple operation or stencil applications count, while we strive to be more specific in terms of the expected runtime on different architectures. Similar works, with the same limitations, have been performed by Pirozzoli et al. [15, 16]

Comparative studies for other equations and other solvers are abundant in the literature. Benzley et al. [17] review the convergence accuracy of several FEM meshes for the solution of elastic continuum problems, but do not use a performance model. Work focused on performance-accuracy relationship for FEM has been performed by Cifuentes and Kalbag [18], although it only provides empirical measurements, and no models. Castillo et al. [19] provided as insightful discussion about a-priori error estimation for the DG method. The error-controlled Fast Multipole Method [20] is well suited for error-constrained optimization. The topic has been touched by Arnold et al. [21], with empirical results, and research is currently active in that direction.

2 Accuracy modeling

In this section we present a model for the a-priori estimation of the error given by linear Finite Difference (FD) schemes. As a case study, consider the advection equation in one dimension [22]:

$$\frac{\partial \varphi}{\partial t} = -c \cdot \frac{\partial \varphi}{\partial x} \tag{1}$$

The typical FD approach to discretize this equation is to substitute the spatial derivative by a computation involving only a fixed set of neighboring values (stencil) and the temporal derivative by an integration method, such as one of the Runge-Kutta schemes. For instance, the derivative stencil could be the following third-order upwind discretization (assuming a positive advection velocity c):

$$\left.\frac{\partial \varphi}{\partial x}\right|_{x_j} \approx \frac{2\,\varphi(x_{j+1}) + 3\,\varphi(x_j) - 6\,\varphi(x_{j-1}) + \varphi(x_{j-2})}{6\Delta x} \tag{2}$$

The temporal derivative could be implemented using the classical Runge-Kutta method (named *RK41* in [23]):

$$k_1 := f\left(\varphi_j^n\right), \quad k_2 := f\left(\varphi_j^n + \frac{\Delta t}{2}\,k_1\right), \quad k_3 := f\left(\varphi_j^n + \frac{\Delta t}{2}\,k_2\right), \quad k_4 := f\left(\varphi_j^n + \Delta t\,k_3\right)$$
$$\varphi_j^{n+1} := \varphi_j^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where f is the function which computes the right hand side value $-c\cdot\frac{\partial\varphi}{\partial x}$ by means of the chosen spatial discretization stencil.

In the following we will identify each solver using the notation $\mathtt{P}\alpha\mathtt{Q}\beta$, where α and β represent the spatial and temporal discretization order of the method respectively. For instance, the method described in the previous paragraph is named `P3Q4`. We always choose the most compact upwind or centered stencil of the desired spatial order [22] and the *RK32* and *RK41* Runge-Kutta schemes as defined in [23] as the third and fourth-order integration schemes respectively.

In order to analyze the behavior of the chosen discretization, we use the Von-Neumann analysis [22]. We assume initial conditions in the form of a wave with wavenumber k:

$$\varphi_j^0 := \mathrm{e}^{i\cdot k\cdot j\,\Delta x}, \tag{3}$$

and we apply one time step of the solution algorithm, which delivers the numerical result φ_j^1. The ratio of this result to the initial condition is called *amplification factor* and denoted by $A := \varphi_j^1/\varphi_j^0$. This complex number, which is independent of j for linear schemes, gives information about the amplitude and the phase of the resulting wave through its absolute value

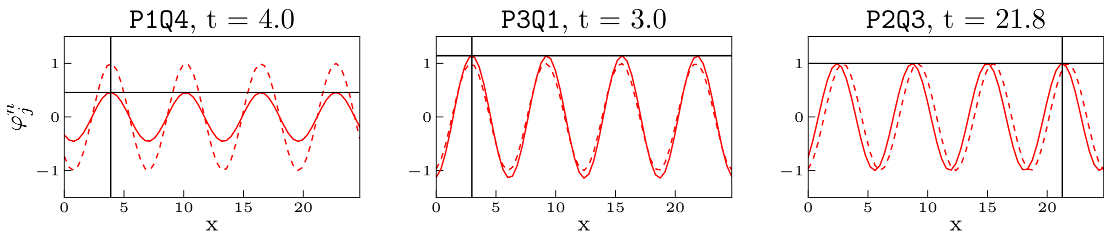

Figure 1: Comparison of exact (dashed lines) and numerical (solid lines) solution to the linear advection problem for three different solvers. The black horizontal and vertical lines show the prediction for amplitude and phase given by the numerical solver respectively.

$|A|$ and its argument $\arg(A)$ respectively. The FD scheme is only consistent, if these values converge to 1 and $k \cdot c \cdot \Delta t$ respectively when Δx and Δt tend to 0. Moreover, the amplitude should never be larger than 1 to avoid instability.

Figure 1 shows how this analysis can predict with excellent accuracy the evolution of the wave by a FD scheme. We recognize a damping scheme (`P1Q4`), an unstable amplifying scheme (`P3Q1`) and a scheme with a large phase error (`P2Q3`). The amplitude and phase of the simulated wave after n time steps will be $|A|^n$ and $\arg(A) \cdot n$ respectively. The numerical solution at the time $t = n\,\Delta t$ corresponding to the initial conditions given in equation (3) can be predicted without running the scheme by:

$$\varphi(t;x) = |A|^n \cdot \mathrm{e}^{i \cdot k \cdot (j\,\Delta x - \arg(A) \cdot n\,\Delta t)} \tag{4}$$

This allows for a good approximation of the L^2 error norm at the end of the simulation, which is defined as follows:

$$\varepsilon_{L^2} := \int_\Omega \left(\bar{\varphi}(t;x) - \varphi(t;x)\right)^2 \mathrm{d}x, \qquad \text{with} \quad \bar{\varphi}(t;x) := e^{i \cdot k \cdot (x - c \cdot t)}, \tag{5}$$

where $\bar{\varphi}(t;x)$ represents the exact solution, and $\varphi(t;x)$ is the numerical solution computed at the time t, interpolated to x. In the experiments performed in this work we approximate ε_{L^2} with a first-order quadrature using the same grid points as the spatial discretization, so that the interpolation plays no role.

To generalize, we can use the following initial conditions and respective analytic solution:

$$\varphi_j^0 := \sum_{m=1}^{M} b_m \cdot \mathrm{e}^{i \cdot k_m \cdot j\,\Delta x}, \qquad \bar{\varphi}(t;x) := \sum_{m=1}^{M} b_m \cdot \mathrm{e}^{i \cdot k_m \cdot (x - c \cdot t)} \tag{6}$$

for any $M \in \mathbb{N}$, $b_m \in \mathbb{R}$, $k_m \in \mathbb{R}^+$. The numerical solution is predicted by the model as:

$$\varphi(t;x) = \sum_{m=1}^{M} |A_{k_m}|\, b_m \cdot \mathrm{e}^{i \cdot k_m \cdot (x - \arg(A_{k_m}) \cdot t)}. \tag{7}$$

The amplification factor A depends on the wave number, and we indicate this dependency by subscribing the relevant wave number in A_{k_m}. Our model captures the different velocities and aplifications which are applied to waves of different wave numbers.

In order to validate our model, we solve the linear advection equation with the initial condition given in the first panel of Figure 2. Different solvers are used, and for each of them a convergence study is performed: the grid spacing and the time step size are progressively reduced. We compute the error norm for each instance and compare it with the prediction offered by the model. The result of this comparison is shown in the center panel. The prediction offered by the model is almost indistinguishable from the actually obtained error. The behavior exhibited by the `P5Q4` and by the `P7Q4` schemes with very small grid spacings is due to the

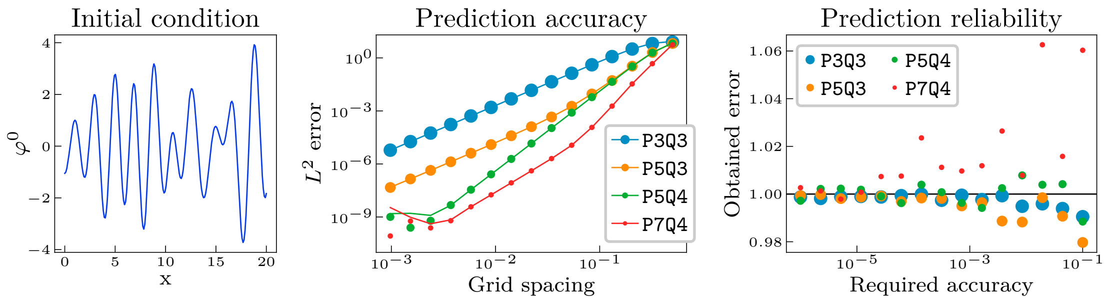

Figure 2: Usability of the accuracy model. Markers represent the experimental results. Solid lines represent the model prediction.

floating-point accuracy: no error reduction is possible with the double precision data type used in these experiments, thus the lines become almost flat. The error model captures this effect, although its prediction accuracy is deteriorated for the same reasons.

We perform an experiment to assess the model usability: For varying ε, we search for the coarsest simulation (i.e., the simulation with the largest grid spacing), whose predicted error is not higher than ε. We then run the simulation with the obtained grid spacing and compare the *Obtained error* with ε. ε will be called *Required accuracy*, i.e., the largest error which is considered acceptable. The last panel in Figure 2 presents the results of this experiment. The vertical axis represents the *overshoot* of the model, i.e., the ratio of the obtained error to the requested accuracy. This ratio should lie as close as possible to the horizontal black line representing the neutral case. In the worst case, our model produces an overshoot of 6% for the highest order scheme and less than 1% for all other schemes.

3 Performance modeling

The performance of a solver can be defined in various ways, which typically correspond to the inverse of some measure of the resources used to solve the problem under consideration. The most usual resources are time, node-hours and energy. A solver performs better than another if it solves the same task with a lower cost. The assessment and prediction of the cost is the topic of this section.

Operation counts, i.e., the number of floating point operations a solver performs to compute the result, has proven [24] to provide successful comparisons for a group of similar methods, but cannot be used to compare the expected performance of the same algorithm across architectures. Moreover, it disregards important factors like cache effects, thread synchronization, and, most importantly, memory access. The evolution of the performance of computational units and DRAM are vastly different [25]. It is therefore increasingly important to consider DRAM bandwidth as the main aspect when analyzing the performance of an application.

A more realistic and architecture-agnostic performance model is the well-known Roofline model [26]. In this model, the performance is limited by two factors, which are specific to the hardware on which the computation happens: the peak performance in terms of floating-point operations per second and the maximum bandwidth between the DRAM memory and the processing units. The arithmetic intensity of an algorithm expresses the ratio of the arithmetic operations are performed on floating-point values to the number of bytes transferred between the DRAM and the processing units. This value specifies which of the two factors is limiting the performance: if the intensity is low, memory bandwidth will be the limiting factor, while intensive algorithms will have their performance bounded by the peak performance.

Finite Difference stencil computations typically exhibit low arithmetic intensity. For instance, an Euler integration step of linear advection with the third order stencil (2), applied on arrays whose size is N will require the following steps:

- Load the initial condition for φ and the velocity u,
- Run the stencil and add the result to the initial condition to obtain the new value of φ,
- Store the newly computed φ to the memory,
- If the computation runs on an accelerator, consider a constant cost to launch the kernel.

This corresponds to transferring $3\,N$ floating point values — under the optimistic assumption of perfect cache reuse — and performing $11\,N$ floating-point operations — under the pessimistic assumption of no optimizations performed by the compiler. Since each floating-point value is either 4 or 8 bytes long, the ratio of the arithmetic operations to the transferred bytes is at most $\frac{11}{12}\,flops/byte$ for single-precision values and $\frac{11}{24}\,flops/byte$ for double-precision values. On modern architectures, such arithmetic intensity values put the mentioned algorithm clearly on the memory-bound region, where the memory bandwidth is the limiting factor. All methods used in this article exhibit similar low values of arithmetic intensity, making all of them memory-bound. The performance model used in this work predicts the runtime as follows:

$$T = n \cdot \left[\frac{1}{\beta} \cdot \sum_{s=1}^{S} (m_{s,in} + m_{s,out}) + S \cdot \alpha \right], \tag{8}$$

where n is the total number of time steps, S is the number of stages in the Runge-Kutta integration method, $m_{s,in}$ and $m_{s,out}$ $[bytes]$ are the number of bytes consumed and produced, respectively, by the s-th stage, α $[s]$ is the constant cost associated to the launch of a kernel (if any) and β $[bytes/s]$ is the hardware memory bandwidth. The result is the runtime T $[s]$. In order to assess the expected bandwidth β, the STREAM benchmark [27] is used, while the cost α is estimated by performing simple experiments.

4 Our methodology

We make use of the accuracy and performance models introduced in the previous sections to solve the the global optimization problem. Both models take as input the description of the solver, i.e., the integration scheme (*Integrator*) and the spatial derivative discretization (*Stencil*). We show a diagram of the work performed by the methodology in Figure 3.

The accuracy model produces expressions representing the amplitude and phase modifications applied by the solver on the relevant waves. This step is performed symbolically using

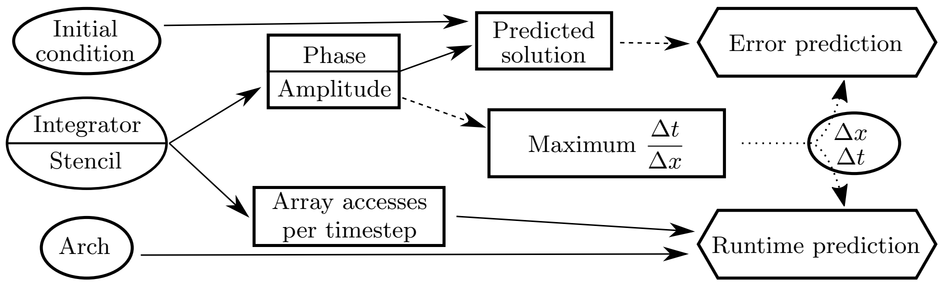

Figure 3: Workflow to the optimization of a solver. Circles represent the inputs, while boxes show the internal values used by the methodology and hexagonals denote the products of the models. Solid arrows represent symbolic manipulations and exact computations, while dashed arrows represent numerical approximations.

the SymPy library [28]. The amplitude expression is then used to perform a numerical analysis to obtain the maximum supported time step size due to stability constraints. On the other side, the amplitude and phase expressions are used to perform a prediction on the L^2 error norm of the solution produced by the solver after a certain integration time (also provided as configuration parameter — not shown in the picture). This numerical mapping from amplitude and phase expressions to error predictions requires the analysis presented in Section 2.

The performance model needs additionally some information about the architecture on which the application is to be run and the floating point type (these are indicated by the "Arch" label in Figure 3). Architecture data include the maximum achievable bandwidth and constant costs, such as the cost to execute a computational kernel. With this information, the performance model is able to provide an estimate of the total runtime by counting the total amount of accessed memory. No training is necessary for this model to be effective, besides the need to know the result of the STREAM benchmark as was explained in Section 3.

The optimization task is applied on both products of the models: the error prediction and the runtime estimate. The optimizer will return the solver configuration which minimizes the runtime as estimated by the performance model under the constraint given by the maximum error as requested by the user and predicted by the accuracy model. The dotted lines to the right of Figure 3 represent the core of the optimization machinery: the values Δx and Δt can be varied under the constraint given by the stability expression (*Maximum* $\frac{\Delta t}{\Delta x}$), and the error and runtime predictions are the optimization constraint and target respectively.

The procedure depicted thus far drives to the optimization of a single solver. When the described methodology is applied on a set of solvers, it will deliver, for each of them, the optimal configuration to reach the requested level of accuracy and the expected runtime. By comparing these runtimes one can easily choose the solver that minimizes the runtime, which represents the solution to the global optimization problem.

5 Experiments

In order to validate our methodology, we performed experiments on the linear advection equation with constant (positive) velocity, as defined in equation (1). The initial conditions were set to a superposition of waves with wavenumbers in the range $\left[\frac{1}{2}, 4\right)$ as depicted in the first panel of Figure 2. We implemented high-performance versions of multiple FD schemes for both CPU and GPU architectures. We performed a large number of simulations with varying grid spacing and Courant number for each of the chosen method. The goal of these runs was to isolate, for each *required accuracy* ε (as defined in Section 2), the method, grid spacing and Courant number which produces a solution within the accuracy constraint in the shortest runtime. This exhaustive search approach was very expensive, requiring multiple tens of node hours.

The same task can be performed using our methodology, with the advantage that the methods do not need to be implemented in a high-performance context, but only as a prototype Python version. The target architecture is not used for the optimization problem: the only relevant architecture property is the maximum achievable bandwidth, as explained in Section 3. No actual simulations have to be completed. The optimization problem is solved by our methodology on a consumer desktop computer in approximately 15 minutes.

Figures 4 and 5 show the result of the optimization task on a CPU (Xeon E5-2690 v3) and a GPU (Nvidia Tesla P100) of the Piz Daint computer at the Swiss National Supercomputing Center CSCS[2]. The best solver configuration (i.e., the grid spacing and Courant number which deliver an acceptable accuracy within the shortest runtime) is found by the exhaustive search (orange squares) and by our methodology (blue circles). Very similar results are obtained.

[2]See `http://www.cscs.ch`

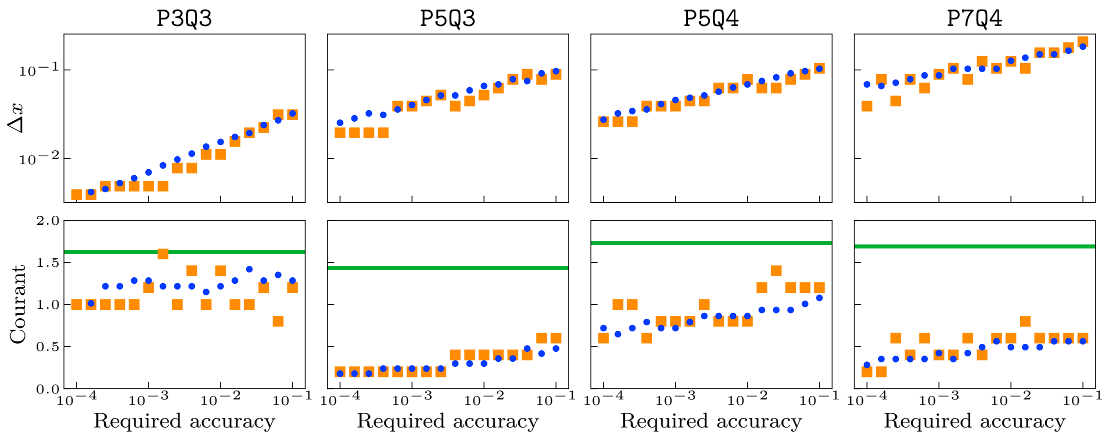

Figure 4: Optimization results on CPU: the orange squares represent the results of the exhaustive search, while the blue circles are the results given by our methodology. The thick green horizontal bar shows the highest stable Courant number for each solver.

Figures 4 and 5 show that the optimal Courant number for most schemes is considerably lower than the one allowed by the stability constraint represented by the green lines. This simple consideration is contrary to the common intuition and represents a highly interesting remark: a smaller time step on a coarser grid can be more beneficial than large time steps. However, this trade-off depends on the chosen discretization. For instance large time steps proved to be very efficient with the low-order P3Q3 scheme, while small steps performed much better on higher order schemes, like P7Q4. The comparison of P5Q3 and P5Q4 shows that, in this case, increasing the time discretization order made larger steps more efficient. One can come to the same conclusions when analyzing the experimental results and when using the modeling.

We conclude this experiment with the global optimization solution. When the optimal grid spacing and time step sizes are found for each solver and each requested error, we can easily search for the best tuple (*architecture*, *solver*, Δx, Δt) which minimizes the runtime. We again perform this task with both the exhaustive search and our methodology, with the results schematized in Figure 6. In all cases, the GPU architecture performed better, and the models captured this phenomenon, thus the diagram does not take into consideration this dimension.

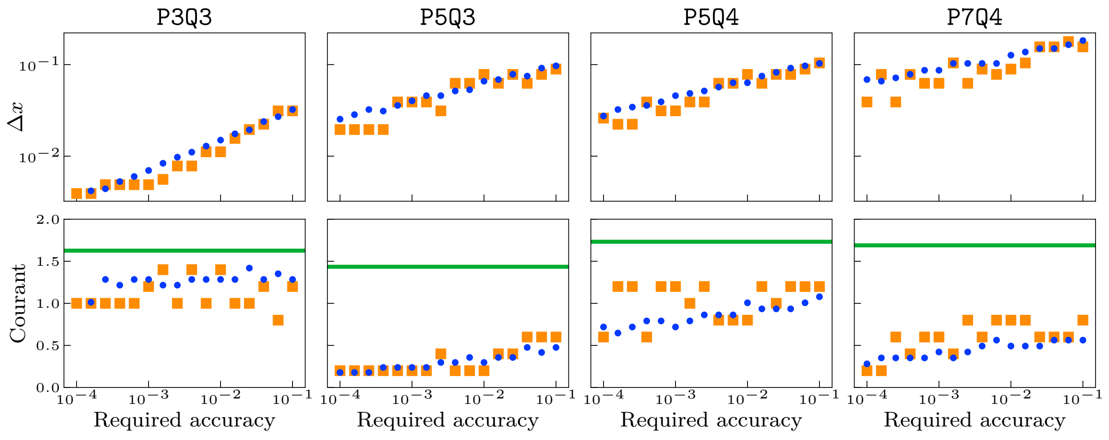

Figure 5: Optimization results on GPU

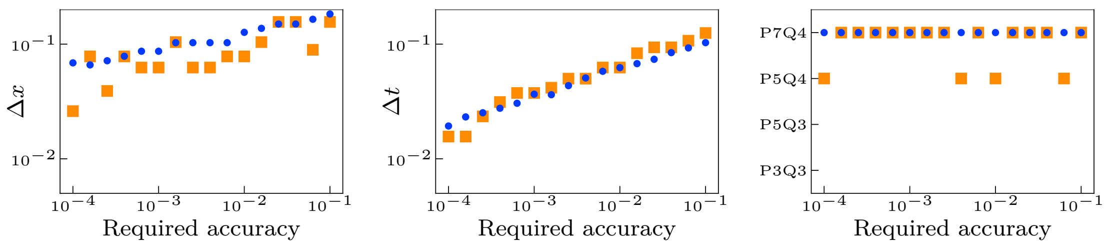

Figure 6: Results of the global optimization problem

6 Conclusions

In this work we formulate and solve an optimization problem to choose the best algorithm and its best configuration for the solution of a linear advection PDE. The algorithm that delivers a solution of acceptable accuracy within the shortest runtime is the optimization target.

We showed two approaches to solve this problem. The exhaustive search approach scans the entire space by performing a large number of simulations with the high-performance application on the target architecture, collects the results and compares them. Our new methodology does not require the development of a high-performance application, nor its execution on the target machine: It predicts the performance of the high-performance application by using a performance model, and estimates the error based on numerical analysis. A very important fact we want to highlight here is that such a methodology based upon modeling can guide the developer towards a rational and unbiased choice of the solver. This guidance should replace the need to guess the best choice for algorithms, time step sizes and other parameters.

We presented an experiment where the two approaches were used to perform the same task. While the exhaustive search required more than 32 node hours on the Piz Daint supercomputer, our methodology gave very similar results in approximately 15 minutes on a desktop computer.

We recognize the limitations of our methodology in its current form: It can just address linear equations, and the framework has been currently validated only on one-dimensional problems. Nevertheless, the promising results are encouraging our research towards the extension of this approach to more general problems and solution algorithms.

References

[1] P. H. Lauritzen et al. *Numerical Techniques for Global Atmospheric Models.* Vol. 80. Springer Science & Business Media, 2011. ISBN: 9783642116391.

[2] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics.* 2nd ed. Springer Berlin Heidelberg, 2003. ISBN: 9783540780915.

[3] W. H. Reed and T. R. Hill. "Triangular mesh methods for the neutron transport equation". In: *Los Alamos Scientific Laboratory Report* (1973).

[4] C. J. Cotter and A. T. T. McRae. *Compatible finite element methods for numerical weather prediction.* 2014. arXiv: `1401.0616`.

[5] J. Thuburn. "Horizontal Discretizations: Some Basic Ideas". In: *Numerical Techniques for Global Atmospheric Models* (2011).

[6] J. Thuburn. "Vertical Discretizations: Some Basic Ideas". In: *Numerical Techniques for Global Atmospheric Models* (2011).

[7] M. Baldauf. "Stability analysis for linear discretisations of the advection equation with Runge-Kutta time integration". In: *Journal of Computational Physics* 227.13 (2008).

[8] P. D. Lax and R. D. Richtmyer. "Survey of the stability of linear finite difference equations". In: *Communications on Pure and Applied Mathematics* 9.2 (1956).

[9] V. Allampalli et al. "High-accuracy large-step explicit Runge-Kutta (HALE-RK) schemes for computational aeroacoustics". In: *Journal of Computational Physics* 228.10 (2009).

[10] M. Calvo, J. M. Franco, and L. Rández. "A new minimum storage Runge-Kutta scheme for computational acoustics". In: *Journal of Computational Physics* 201.1 (2004).

[11] K. Tselios and T. Simos. "Runge-Kutta methods with minimal dispersion and dissipation for problems arising from computational acoustics". In: *Journal of Computational and Applied Mathematics* 175.1 (2005).

[12] T. Toulorge and W. Desmet. "Optimal Runge-Kutta schemes for discontinuous Galerkin space discretizations applied to wave propagation problems". In: *Journal of Computational Physics* 231.4 (2012).

[13] D. Lockard, K. Brentner, and H. Atkins. "High-accuracy algorithms for computational aeroacoustics". In: *AIAA journal* (1995).

[14] D. Zingg. "Comparison of high-accuracy finite-difference methods for linear wave propagation". In: *SIAM Journal on Scientific Computing* (2000).

[15] S. Pirozzoli. "Performance analysis and optimization of finite-difference schemes for wave propagation problems". In: *Journal of Computational Physics* 222.2 (2007).

[16] M. Bernardini and S. Pirozzoli. "A general strategy for the optimization of Runge-Kutta schemes for wave propagation phenomena". In: *Journal of Computational Physics* 228.11 (2009).

[17] S. E. Benzley et al. "A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis". In: *4th International Meshing Roundtable* 17 (1995).

[18] A. Cifuentes and A. Kalbag. "A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis". In: *Finite Elements in Analysis and Design* 12.3-4 (1992).

[19] P. Castillo et al. "An a priori error analysis of the local discontinuous Galerkin method for elliptic problems". In: *SIAM Journal on Numerical* (2000).

[20] I. Kabadshow and H. Dachsel. "The Error-Controlled Fast Multipole Method for Open and Periodic Boundary Conditions". In: *Fast Methods for Long-Range Interactions in Complex Systems, IAS Series, Volume 6*. 2011, pp. 85–114. ISBN: 9783893367146.

[21] A. Arnold et al. "Comparison of scalable fast methods for long-range interactions". In: *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 88.6 (2013).

[22] D. R. Durran. *Numerical Methods for Fluid Dynamics - With Applications to Geophysics*. Vol. 32. Springer Science & Business Media, 2010.

[23] J. C. Butcher. *The numerical analysis of ordinary differential equations : Runge-Kutta and general linear methods*. J. Wiley, 1987. ISBN: 0471910465.

[24] T. Colonius and S. Lele. "Computational aeroacoustics: progress on nonlinear problems of sound generation". In: *Progress in Aerospace sciences* (2004).

[25] W. A. Wulf and S. A. McKee. "Hitting the memory wall". In: *ACM SIGARCH Computer Architecture News* 23.1 (1995).

[26] S. Williams, A. Waterman, and D. Patterson. "Roofline: an insightful visual performance model for multicore architectures". In: *Communications of the ACM* 52.4 (2009).

[27] J. D. McCalpin. "Sustainable Memory Bandwidth in High Performance Computers". In: *Silicon Graphics Inc* (1995). URL: `http://www.cs.virginia.edu/stream/ref.html`.

[28] D. Joyner et al. "Open source computer algebra systems: SymPy". In: *ACM Communications in Computer Algebra* 45.177 (Jan. 2011).