About using analog computers in today's largest computational challenges

Sven Köppel[1], **Bernd Ulmann**[1], **Lars Heimann**[1], **and Dirk Killat**[1,2]

[1]Anabrid GmbH, Am Stadtpark 3, 12167 Berlin, Germany
[2]Microelectronics Department, Brandenburg University of Technology, 03046 Cottbus, Germany

Correspondence: Sven Köppel (koeppel@anabrid.com)

Abstract. Analog computers perceive a revival as a feasible technology platform for low precision, energy efficient and fast computing. We quantify this statement by measuring the performance of a modern analog computer and comparing it with traditional digital processors. General statements are made about ordinary and partial differential equations. As an example for large scale scientific computing applications, computational fluid dynamics are discussed. Several models are proposed which demonstrate the benefits of analog and digital-analog hybrid computing.

1 Introduction

Digital computing has transformed many if not close to all aspects of industry, humanities and science. Turing completeness allows for making statements about computability and decidability of problems and "computational power" of machines. Digital storage has undergone numerous technological advances and is available in increasingly vast amounts. Nevertheless, contemporary computing is not the last word in computing, despite it's dominance in the consumer market for the last 40+ years.

Fundamental research about non-traditional (also referred to as *unconventional* or *exotic*) computing is taking place in material physics, chemistry but also in more exotic branches such as biology and life sciences. Amongst others, beyond-Turing computing (Siegelmann, 1995), natural computing (Calude et al., 1999), neuromorphic computing (Schuman et al., 2019; Ziegler, 2020) or quantum computing (Zhou et al., 2020; Georgescu et al., 2014; Kendon et al., 2010) are fields of active investigation. Being fundamental research at heart, these disciplines come along with technological challenges. For instance, computing with DNA still requires

the use of large scale laboratory equipment and machinery (Deaton et al., 1998). Currently, not only the low-temperature laboratory conditions but also the required error correction schemes challenge practical quantum computers (Wilhelm et al., 2017). This negates any practical advantage over silicon based digital computing as of now. Furthermore, all of these alternative or "exotic" computer architectures share the characteristic that they are fundamentally *non-portable*. That means they will be stuck to large facilities and dedicated special-purpose computing centers for a long time, if not forever. This is not necessarily a practical drawback, since the internet allows for delocalization of systems.

In contrast to this, silicon based *electronic analog computing* is a technology with a vivid past, which does not require exotic physical conditions to be maintained (Ulmann, 2020). Analog computers got overtaken by their digital counterparts in the last century, mostly thanks to their ever-increasing digital clock speed and their flexibility due to the algorithmic approach in conjunction with the possibility of using these machine in a time-shared environment. However, nowadays Moore's law is coming to a hard stop and processor clock speeds have not increased since a decade. Manycore architectures and vectorization come with their own share of problems, given their fundamental limits as described, for instance, by Amdahl's law. GPGPUs and specialized digital computing chips concentrate on vectorized and even data flow-oriented programming paradigms but are still limited by parasitic capacitances which determine the possible clock cycle time and provide a noticeable "energy barrier".

Thanks to their properties, analog computers gained interest by many research groups. For surveys of applications see for instance Bournez and Pouly (2018). In this paper, we study the usability of analog computers for applications in science. The fundamental properties of analog computers are

low power requirements, low resolution computation and intrinsical parallelism. Two opposite uses cases/scenarios can be identified: High performance computing (HPC) and low energy portable computing. The energy and computational demands in both scenarios are diametral and we mostly concentrate on HPC in the following text.

The paper is structured as following: In section 2, we review the general assumptions about digital and analog computing. In section 3, small scale benchmark results are presented for a simple ordinary differential equation. In section 4, an exemplary partial differential equation is considered as an example for a large scale problem. Spatial discretization effects and computer architecture design choices are discussed. Finally, section 5 gives a conclusion about the findings.

2   A linear model for work: Modeling analog and digital performance

In this paper, we study different techniques for solving differential equations computationally. Due to the different conventions in algorithmic and analog approaches, a common language has to be found and is to be introduced in this section. Here, the term *algorithmic approach* addresses the classical Euler method or classical quasi-linear techniques in PDEs, *i.e.*, in general methods of numerical mathematics. The term *analog approaches* addresses the continous time integration with an operational amplifier having a capacitor in the feedback loop. The fundamental measures for computer performance under consideration is the time-to-solution T, the power consumption P and the energy demand E.

2.1   Time to solution

The time-to-solution T is the elapsed real time (lab time or wall clock time) for solving a differential equation $\partial_t u = f(u)$ from its initial condition $u(t_0)$ at time t_0 to some target simulation time t_{final}, *i.e.*, for obtaining $u(t_{\text{final}})$. The *speed factor* $k_0 := T/t_{\text{final}}$ is the ratio of elapsed simulation time per wall clock time. In this work, we typically study $t_{\text{final}} = 1$ without loss of generality (*i.e.*, in general depending on the measurement units and scaling chosen). This choise allows on analog computers to identify $1/k_0 = 1/T$ with the *cutoff frequency* $f_c := 1/k_0$ of the analog circuit components. On digital computers, the time-to-solution is used as an estimator (in a statistical sense) for the average k_0. Relating this quantity to measures in numerical schemes will be a main discussion point in this paper.

Given a most simple problem such a $\mathrm{d}y/\mathrm{d}t = f(y) := y$, one can study the analog/digital computer performance in terms of the *complexity* of $f(y)$. For a problem M times as big as the given one, the naturally fully *parallel* analog computer exhibits a constant time-to-solution, *i.e.*, $T_A(M) = T_A = \mathcal{O}(1)$, with $\mathcal{O}$ being part of the Bachmann Landau

Table 1. A linear model for work: The computational cost C of evaluating $f(u)$ in the ODE $\partial u/\partial t = f(u)$ is expected to grow as $C \in \mathcal{O}(M)$. The effects on time-to-solution T, power P and energy E demands are shown.

(Quantity)	Digital	Analog
$T(M)$ [s]	$\mathcal{O}(M)$	$\mathcal{O}(1)$
$P(M)$ [W]	$\mathcal{O}(1)$	$\mathcal{O}(M)$
$E(M)$ [J]	$\mathcal{O}(M)$	$\mathcal{O}(M)$

Big O notation. In contrast, a single core (*i.e.*, nonvectorized, also not considering superscalar architectures) digital computer operates in a *serial* fashion and can achieve a time-to-solution $T_D(M) = M \cdot T_D$, *i.e.*, $T_D = \mathcal{O}(M)$ at best. Here, $M \in \mathbb{N}$ is the measure for the algorithmic complexity of $f(y)$ but not for the number of evolved quantities. The number of computational elements required to implement $f(y)$ on an analog computer or the number of instructions required for computing $f(y)$ on a digital computer could provide numbers for M. This is because the evaluation of $f(y)$ can hardly be numerically parallelized, while multiple quantities can be numerically evolved by parallel digital processors in time. That is, for a system of N coupled ODEs $\mathrm{d}y_i/\mathrm{d}t = f_i(y_1, \ldots, y_N)$, the vector-valued $\boldsymbol{f}$ can be either assigned an effective complexity $\mathcal{O}(NM)$, but an overall complexity $\mathcal{O}(M)$ is realistic, given the MIMD (multiple instruction, multiple data) parallelization to be expected.

Also multi-step schemes implementing high order numerical time integration can exploit digital parallelization (however, in general the serial time-to-solution of a numerical Euler scheme is the barrier for the fastest possible digital time integration). Digital parallelization is always limited by the serial parts of a problem (Amdahl's law, Rodgers, 1985), which makes the evaluation of $f(y)$ the hardest part of the problem. Section 4 discusses complex functions $f(y)$ in the context of the *method of lines* for PDEs.

2.2   Power and energy scaling for the linear model

For a given problem with time-to-solution T and average power consumption P, the overall energy is estimated by $E = PT$ regardless of the computer architecture.

In general, an analog computer has to grow with the problem size M. Given constant power needs per computing element and neglecting increasing resistances or parasitic impedances in general, one can assume $P_A(M) = P_A \cdot M$. In contrast, a serial single node digital computer in principle can compute a problem of any size serially by relying on dynamic memory (DRAM), *i.e.*, $P_D(M) = P_D$. Typically, the DRAM energy demands are one to two orders of magnitude smaller than those of a desktop or server grade processor and are therefore negligible for this estimate.

Interestingly, this model suggests that both E_D and $E_A = \mathcal{O}(M)$, *i.e.*, the analog-digital energy ratio remains constant despite the fact that the analog computer computes linearly faster with increasing problem size M. In this model,

$$\frac{E_A}{E_D} = \frac{M P_A}{P_D} \frac{T_A}{M T_D} = \frac{P_A}{P_D} \frac{T_A}{T_D} = \text{const}. \tag{1}$$

The *orthogonal* performance features of the fully-parallel analog computer and the fully-serial digital computer are also summarized in table 1.

When comparing digital and analog computer power consumption, the power consumption under consideration should include the *total* computer power including administrative parts (like network infrastructure or analog-to-digital converters) and power supplies. In this work, data of heterogenous sources are compared and definitions may vary.

2.3   Criticism and outlook

Given that the digital and analog technology (electric representation of information, transistor-based computation) is quite similar, the model prediction of a similarly growing energy demand is useful. Differences are of course hidden in the pre-factors of the asymptotic notation $\mathcal{O}(M)$. Quantitative studies in the next sections examine this prefactor.

The linear model is already limited in the case of serial digital processors when the computation gets *memory bound* (instead of *CPU-bound*). Having to wait for data leads to a performance drop and might result in a worsened superlinear $T_D(M)$.

Parallel digital computing as well as *serial analog* computing has not yet been subject of the previous discussion. While the first one is a widespread standard technique, the second one refers to analog-digital hybrid computing which allows for instance to utilize a small analog computer repeatedly on a large problem, effectively rendering the analog part as an analog accelerator or co-processor for the digital part. Parallel digital computing suffers from a theoretical speedup which is limited by the non-parallel parts of the algorithm (Amdahl's and Gustafson's law), which has exponential impact on $T_D(M)$. This is where the intrinsically parallel analog computer can unlock its biggest advantages. Section 4 will discuss this aspect of analog computing.

3   A performance survey on solving ordinary differential equations (ODEs)

In this section, quantitative measurements between contemporary analog and digital computers will be made. We adopt the *Analog Paradigm Model-1* computer (Ulmann, 2019, 2020), a modern modular academic analog computer as well as an ordinary Intel© Whiskey Lake "ultra-low power mobile" processor (Core i7-8565U) as a representative of a typical desktop-grade processor. Within this experiment,

we solve a simple[1] test equation $\mathrm{d}^2 y/\mathrm{d}t^2 = \lambda y$ (with real-valued y and $\lambda = \pm 1$) on both a digital and analog computer.

3.1   Time to solution

The simple ordinary differential equation (ODE) is evolved in time on the digital computer with simple text-book level scalar toy codes written in C and Fortran and compiled with GCC. We implement an explicit integrator method (Euler/Runge-Kutta). $N = 2 \times 10^3$ timesteps with timestep size $\Delta t = 5 \times 10^{-4}$ are computed (see also section 4 for a motivation for this time step size) thus $t_{\text{final}} = N \Delta t = 1$. No output[2] is written during the benchmark to ensure the best performance. The time per element update (per integration step) is roughly $(45 \pm 35)\,\text{ns}$. For statistics, we repeat the computation 10^5 times and average. Depending on the order of the integration scheme, we conclude the overall wall clock time to run up to t_{final} as $T_D = (75 \pm 45)\,\mu\text{s}$.

In contrast, the equation is implemented with integrating (and negating, if $\lambda = -1$) operational amplifiers on the *Model-1*. The machine approaches $t_{\text{final}} = 1$ in a wall-clock time $T_A = 1\,\text{sec}/k_0$ with $k_0 \in \{1, 10, 10^2, 10^3, 10^4\}$ the available integration speed factors on the machine (Ulmann, 2019). Therefore, the *Model-1* can obtain the result of $y'' = y$ at $t_{\text{final}} = 1$ in a wall-clock time $T_A = 100\,\mu\text{s}$ at best.

Note how $T_A/T_D \approx 1$, *i.e.*, in the case of the smallest possible reasonable ODE, the digital computer (2020s energy efficient desktop processor) is roughly as fast as the *Model-1* (modern analog computer with an integration level comparable to the 1980s).

For an outlook, given the limited increase in clock frequency, with a faster processor one can probably expect an improvement of T_D down to the order of $1\,\mu\text{s}$. For an analog computer on a chip, one can expect an improvement of T_A down to the order of $1\,\mu\text{s}$-$10\,\text{ns}$. This renders $T_A/T_D \approx 10^{-(1\pm1)}$ as a universal constant.

Summing up, with the given numbers above, as soon as the problem complexity grows, the analog computer outshines the digital one, and this advantage increases linearly.

3.2   Energy and power consumption

We use the performance measure codes `likwid` (Hager et al., 2010; Röhl et al., 2017; Gruber et al., 2020) and `perf` (de Melo, 2010) to measure the overall floating-point operations (FLOP) and energy usage of the digital processor. For the Intel mobile processor, this provides a power consumption of $P_D = 10\,\text{W}$ during computing. This num-

---

[1]This equation is inspired by the Dahlquist (1979) *test equation* $y' = \lambda y$ used for stability studies. The advantage of using an oscillator is the self-similarity of the solution which can be observed over a long time.

[2]Both in terms of *dense output* or any kind of evolution tracking. A textbook-level approach with minimal memory footprint is adopted which could be considered an *in-place* algorithm.

Table 2. Small scaling summary: Measured benchmark (Intel[©] processor vs. Analog Paradigm *Model-1*) and expected/projected analog chip results.

	Measured		Projected
	Digital	Analog (*M1*)	Analog Chip
T [μs]	75 ± 45	100	$10^{-(0.5\pm0.5)}$
$k_0 \sim 1/\Delta t$ [Hz]	3×10^4	10^4	$10^{6.5\pm0.5}$
P [W]	10	0.4	10^{-2}
$E = P \cdot T$ [μJ]	900 ± 600	40	$10^{-(2.5\pm0.5)}$
F [FLOP/sec]	10^9	$3 \times 10^{(4\pm1)}$	7×10^5
F/E [FLOP/J]	10^8	$7.5 \times 10^{8\pm1}$	3×10^{11}

ber stems directly from the CPU performance counters. For the overall computer power, an increase of at least 50% is expected. The overall energy requirement is then $E_D = P_D T_D = (0.9 \pm 0.6)\,\text{mJ}$.

The analog computer energy consumption is estimated as $P_A \approx 400\,\text{mW}$. The number is based on measurements of actual Model-1 computing units, in particular $84\,\text{mW}$ for a single summer and $162\,\text{mW}$ for a single integrator. The overall energy requirement is then $E_A = P_A T_A = 40\,\mu\text{J}$.

Note that $P_D/P_A \approx 25$, while $E_D/E_A \approx (2.25\pm1.5)$. We conclude that the analog and digital computer require a similar amount of energy for the given computation, a remarkable result given the 30 year technology gap between the two architectures compared here.

For power consumption, it is hard to give a useful prospect due to the accumulating administrative overhead in case of parallel digital computing, such as data transfers, non-uniform memory access (NUMA) and switching networking infrastructure. We assume that this will change the ratio E_D/E_A further in favor of the analog computer given both a larger digital and analog computer. Furthermore, higher integration levels lower E_A: the *Model-1* analog computer is realized with an integration level comparable with 1980s digital computers. We can reasonably expect a drop of two to three orders of magnitude in power requirements with fully integrated analog computers.

3.3   Measuring computational power: FLOP per Joule

For the digital computer, the number of computed *floating-point operations* (FLOP[3]) can be measured. The overall single core nonvectorized performance is measured as $F \approx 1\,\text{GFLOP/sec}$. A single computation until t_{final}

---

[3]sic! We either argue with *overall* FLOP and Energy (Joule) or *per second* quantities such as FLOP/sec (in short FLOPs) and Power (Watt). In order to avoid confusion, we avoid the abbreviation "FLOPs" in the main text. Furthermore, SI prefixes are used, *i.e.*, kFLOP = 10^3 FLOP, MFLOP = 10^6 FLOP and GFLOP = 10^9 FLOP.

required roughly $F_D = 3\,\text{kFLOP}$. The ratio $F_D/P_D = 100\,\text{MFLOP/J}$ is a measure for the number of computations per energy unit on this machine. This performance is one to two orders lower than typical HPC numbers. This is because we benchmark an energy-saving desktop CPU and not a high-end processor. Furthermore, this benchmark is by purpose single-threaded.

In this non-vectorized benchmark, one can ignore the reduced resolution of the analog computer (which is slightly lower than an IEEE 754 half precision floating-point, compared to the double precision floating-point numbers in the digital benchmark). One can then assign the analog computer a *time-equivalent* floating-point operation performance

$$F_A := F_D \frac{T_A}{T_D} \approx 10^{(1\pm1)}F_D = 3 \times 10^{(4\pm1)}\,\text{FLOP}\,. \qquad (2)$$

We then achieve the analog computer FLOP-per-Joule ratio (note that FLOP/J = FLOPs/W),

$$\frac{F_A}{E_A} = \frac{3 \times 10^{(4\pm1)}\,\text{FLOP}}{40\,\mu\text{J}} = 7.5 \times 10^{8\pm1}\,\text{FLOP/J}\,. \qquad (3)$$

That is, the analog computer's "FLOP per Joule" is slightly larger than for the digital one. Furthermore, one can expect an increase of F_A/E_A by 10-100 for an analog computer chip (see for instance Cowan, 2005; Cowan et al., 2005, claiming $20\,\text{GFlop/sec}$).

Keep in mind that the FLOP/sec or FLOP/J measures are (even in the case of comparing two digital computers) always problem/algorithm-specific (*i.e.*, in this case a Runge Kutta solver of $y'' = y$) and therefore controversial as a comparable figure.

4   PDEs and many degrees of freedom

In this section, forecasts about large scale differential equations are made. No benchmarks have been carried out, since a suitable integrated analog computer on chip does not yet exist. For the estimates, we assume an analog computer on chip with an average energy consumption of about $P_N = 4\,\text{mW}$ per computing element (*i.e.*, per integration, multiplication, etc.)[4] and cutoff frequency $f_c = 100\,\text{Mhz}$, which we refer to as the *analog* cutoff frequency f_c^A in the following. These numbers are several orders of magnitude better then the $P_N = 160\,\text{mW}$ and $f_c = 100\,\text{kHz}$ of the *Model-1* computer discussed in the previous section. For the digital part, different systems then before are considered.

4.1   Solving PDEs on digital and analog computers

Partial differential equations (PDEs) are among the most important and powerful mathematical frameworks for describ-

---

[4]Summation will be done implicitly on chip by making use of Kirchhoff's law so that no explizit computing element are required for this operation.

ing dynamical systems in science. PDE solutions are usually fields $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{r}, t)$, *i.e.*, functions[5] of spatial position $\boldsymbol{r}$ and time t. In the following, we concentrate on initial value boundary problems (IVBP). These problems are described by a set of PDEs valid within a spatial and temporal domain and complemented with field values imposed on the domain boundary. For a review of PDEs, their applications and solutions see for instance Brezis and Browder (1998). Furthermore, in this text, we use computational fluid dynamics (CFD) as a representative theory for discussing general PDE performance. In particular, classical hydrodynamics (Euler equation) in a flux-conservative formulation, as introduced in appendix 4.2, is described by an hyperbolic conservation laws. Such PDEs have a long tradition of being solved with highly accurate numerical schemes.

Many methods exist for the spatial discretization. While finite volume schemes are popular for their conservative properties, finite difference schemes are in general cheaper to implement. In this work, we stick to simple finite differences on a uniform grid with some uniform grid spacing $\Delta \boldsymbol{r}$. The evolution vector field $\boldsymbol{u}(\boldsymbol{r}, t)$ is sampled on G grid points per dimension and thus replaced by $\boldsymbol{u}_k(t)$ with $0 \leq k \leq G$. It is worthwhile to mention that this approach works in classical orthogonal "dimension by dimension" fashion, and the number of total grid points is given by G^D. The computational domain is thus bound by $\Omega = [\boldsymbol{r}_0, \boldsymbol{r}_G]^D$. A spatial derivative $\partial_i f$ is then approximated by a central finite difference scheme, for instance $\partial_i f_k \approx (f_{k+1} - f_{k-1})/(2\Delta x) + \mathcal{O}(\Delta x^2)$ for a second order accurate central finite difference approximation for some function F at grid point k.

Many algorithmic solvers implement numerical schemes which exploit the *vertical method of lines* (MoL) to rewrite the PDE into coupled ordinary differential equations (ODEs). Once applied, the ODE system can be written as $\partial_t u^k = G^k(\boldsymbol{u}, \boldsymbol{\nabla u})$ with u^k denoting the time evolved (spatial) degrees of freedom and G^k functions containing spatial derivatives $(\partial_i u^j)$ and algebraic sources. A standard time stepping method determines a solution $u(t_1)$ at later time $t_1 > t_0$ by basically integrating $u^k(t_1) = \int_{t_0}^{t_1} G^k(\boldsymbol{u}(t)) \mathrm{d} t + u^k(t_0)$. Depending on the details of the scheme, G^k is evaluated (probably repeatedly or in a weak-form integral approach) during the time integration of the system. However, note that other integration techniques exist, such as the arbitrary high order ADER technique (Titarev and Toro, 2002, 2005). The particular spatial discretization method has a big impact on the computational cost of G^i. In the following, we focus on the (most simple) finite difference technique, where the numbers of neighbor communications per dimension grows linearly with the *convergence order* of the scheme.

---

[5]The explicit dependency on $\boldsymbol{r}$ and t is omitted in the following text.

4.2 Classical Hydrodynamics on analog computers

The broad class of *fluid dynamics* will be discussed as popular yet simple type of PDEs. It is well known for its efficient description of the flow of liquids and gases in motion and is applicable in many domains such as aerodynamics, in life sciences as well as fundamental sciences (Sod, 1985; Chu, 1979; Wang et al., 2019). In this text, the most simple formulation is investigated: the Newtonian hydrodynamics (also refered to as *Euler equations*) with an ideal gas equation of state. It is given by a nonlinear PDE describing the time evolution of a mass density ρ, it's velocity v^i, momentum $p^i = \rho v^i$ and energy $e = t + \varepsilon$, with the kinetic contribution $t = \rho \, \boldsymbol{v}^2/2$ and an "internal" energy ε, which can account for forces on smaller length scales than the averaged scale.

Flux conservative *Newtonian hydrodynamics* with an ideal gas equation of state is one of the most elementary and textbook level versions of fluid dynamics (Toro, 1998; Harten, 1997; Hirsch, 1990). The PDE system can be written in a dimension agnostic way in D spatial dimensions (*i.e.*, independent of the particular choice for D) as

$$\frac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla} \cdot \boldsymbol{f} = \boldsymbol{S} \quad \text{with} \quad \boldsymbol{\nabla} \cdot \boldsymbol{f} = \sum_i^{n_d} \frac{\partial \boldsymbol{f}^i}{\partial x^i}, \tag{4}$$

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ p^j \\ e \end{pmatrix} \;, \; \boldsymbol{f}^i = \boldsymbol{f}^i(\boldsymbol{u}, \boldsymbol{v}) = \begin{pmatrix} p^i \\ p^i v^j - p\,\delta^{ij} \\ v^i\,(e + p) \end{pmatrix}, \tag{5}$$

with $i, j \in [1..D]$. Here, the pressure $p = \rho\varepsilon(\Gamma - 1)$ defines the ideal gas equation of state, with adiabatic index $\Gamma = 2$ and δ^{ij} is the Kronecker delta. A number of vectors are important in the following: The integrated state or *evolved* vector $\boldsymbol{u}$ in contrast to the primitive state vector or *auxiliary* quantities $\boldsymbol{v}(u) = (p, v^i)$, which is a collection of so called *locally reconstructed* quantities. Furthermore, the right hand sides in (4) do not explicitly depend on the spatial derivative $\partial^i \rho$, thus the conserved flux vector $\boldsymbol{f} = \boldsymbol{f}(\nabla q, \boldsymbol{v})$ is only a function of the derivatives of the *communicated* quantities $\boldsymbol{q} = \boldsymbol{q} = (e, p^i)$ and the auxiliaries $\boldsymbol{v}$. Furthermore, $\boldsymbol{q}$ and $\boldsymbol{v}$ are both functions of $\boldsymbol{u}$ only.

$\boldsymbol{S} = 0$ is a source term. Some hydrodynamical models can be coupled by purely chosing some nonzero $\boldsymbol{S}$. An example are the popular *Navier Stokes* equations which describe viscous fluids. Compressible Navier Stokes equations can be written with a source term $\boldsymbol{S} = \boldsymbol{\nabla} \cdot \boldsymbol{F}^v$, with

diffusion fluxes $\boldsymbol{F}^v = (0, \tau^{ij}, \sum_k \tau^{ik} v^k - q^j)^T$,

viscous stress tensor $\tau^{ij} = \mu(\partial^i v^j + \partial^j v^i - \frac{2}{3}(\partial^k v^k)\delta^{ij})$,

and heat flux $q^i = -(c_p\mu/Pr)\partial^i T$,

with specific heats c_p, c_v, viscosity coefficient μ, Prandtl number Pr and temperature T determined by the perfect gas

equation of state, *i.e.*, $T = (e - \boldsymbol{v}^2)/(2c_v)$. The (local) computational cost from Euler equation to Navier Stokes equation is roughly *doubled*. Furthermore, the partial derivatives on the velocities and temperatures also *doubles* the quantities which must be communicated with each neighbor in every dimension. For the following text, we restrict to Euler equations for the sake of simplicity.

4.3 Spatial discretization: Trading interconnections vs. computing elements

In general, we allow to investigate schemes of order F, which requires to involve F neighbor elements. For instance, a $F = 4$th order accurate stencil requires to communicate and/or compute $\boldsymbol{f}_{k-2}, \boldsymbol{f}_{k-1}, \boldsymbol{f}_{k+1}, \boldsymbol{f}_{k+2}$. Empirically, $F = 4 - 6$ is sufficient for a long-time evolution.

For a $F = 2$nd order stencil, we identify three different subcircuits

$$\boldsymbol{u}_k(\boldsymbol{f}_{k+1}, \boldsymbol{f}_{k-1}) := \int (\boldsymbol{f}_{k+1} - \boldsymbol{f}_{k-1}) \, \mathrm{d}t / (2\Delta x), \qquad (6)$$

with $\boldsymbol{f}_{k\pm 1} := \boldsymbol{f}_k(\boldsymbol{q}_{k\pm 1}, \boldsymbol{v}_k)$ and $\boldsymbol{v}_k := \boldsymbol{v}_k(\boldsymbol{u}_k)$ according to their respective definitions before. Figure 1 shows this "building block" for a single grid point, exemplary for up to $D = 2$ dimensions with an $F = 2$nd order finite difference stencil. The circuit names a number of intermediate expressions which are labeled as in the following:

$$\partial_t \begin{pmatrix} \rho_{i,k} \\ p_{i,k}^x \\ p_{i,k}^y \\ e_{i,k} \end{pmatrix} \qquad (7)$$

$$= \frac{\overbrace{\begin{pmatrix} p_{i+1,k}^x \\ p_{i+1,k}^x v_{i+1,k}^x - p_{i+1,k} \\ p_{i+1,k}^x v_{i+1,k}^y \\ v_{i+1,k}^x(e_{i+1,k} + p_{i+1,k}) \end{pmatrix}}^{\boldsymbol{f}_{i+1,k}^x} - \overbrace{\begin{pmatrix} p_{i-1,k}^x \\ p_{i-1,k}^x v_{i-1,k}^x - p_{i-1,k} \\ p_{i-1,k}^x v_{i-1,k}^y \\ v_{i-1,k}^x(e_{i-1,k} + p_{i-1,k}) \end{pmatrix}}^{\boldsymbol{f}_{i-1,k}^x}}{2\Delta x}$$

$$+ \frac{\overbrace{\begin{pmatrix} p_{i,k+1}^y \\ p_{i,k+1}^y v_{i,k+1}^x \\ p_{i,k+1}^y v_{i,k+1}^y - p_{i,k+1} \\ v_{i,k+1}^y(e_{i,k+1} + p_{i,k+1}) \end{pmatrix}}^{\boldsymbol{f}_{i,k+1}^y} - \overbrace{\begin{pmatrix} p_{i,k-1}^y \\ p_{i,k-1}^y v_{i,k-1}^x \\ p_{i,k-1}^y v_{i,k-1}^y - p_{i,k-1} \\ v_{i,k-1}^y(e_{i,k-1} + p_{i,k-1}) \end{pmatrix}}^{\boldsymbol{f}_{i,k-1}^y}}{2\Delta y}$$

$$= \frac{\begin{pmatrix} \Delta\rho_{i+1,k} \\ \Delta p_{i+1,k}^x \\ \Delta p_{i+1,k}^y \\ \Delta e_{i+1,k} \end{pmatrix} - \begin{pmatrix} \Delta\rho_{i-1,k} \\ \Delta p_{i-1,k}^x \\ \Delta p_{i-1,k}^y \\ \Delta e_{i-1,k} \end{pmatrix}}{2\Delta x} + \frac{\begin{pmatrix} \Delta\rho_{i,k+1} \\ \Delta p_{i,k+1}^x \\ \Delta p_{i,k+1}^y \\ \Delta e_{i,k+1} \end{pmatrix} - \begin{pmatrix} \Delta\rho_{i,k-1} \\ \Delta p_{i,k-1}^x \\ \Delta p_{i,k-1}^y \\ \Delta e_{i,k-1} \end{pmatrix}}{2\Delta y}$$

Just like in figure 1, all expressions which are vanishing in a single spatial dimension are colored in red. Furthermore note how the index i is for x-direction and k for y-direction, and that there are different fluxes $\boldsymbol{f}^j$ in the particular directions.

(7) is closed with the element-local auxiliary recovery

$$\begin{pmatrix} v_{i,k}^x \\ v_{i,k}^y \\ p_{i,k} \end{pmatrix} = \begin{pmatrix} p_{i,k}^x / \rho_{i,k} \\ p_{i,k}^y / \rho_{i,k} \\ e_{i,k} - \rho_{i,k}\left((v_{i,k}^x)^2 + (v_{i,k}^y)^2\right)/2 \end{pmatrix}. \qquad (8)$$

Note that one can trade in neighbor communication (*i.e.*, number of wires between grid points) for local recomputation. For instance, it would be mathematically clean to communicate the conservation quantities $\boldsymbol{u}$ only and reconstruct $\boldsymbol{v}$ whenever needed. In order to avoid too many recomputations, some numerical codes communicate also parts of $\boldsymbol{v}$. In an analog circuit, it is even thinkable of communicating parts of the finite differences such as the $\Delta\boldsymbol{v}_{i,k}$ quantities in equation (7).

We determine the number of analog computing elements required to solve the Euler equation on a single grid point as $N_{\text{single}} = 5D + 5F(D + 2) + 9$, with D being the number of spatial dimensions and F the convergence order (*i.e.*, basically the finite difference stencil size). Typical choices of interest are convergence orders of $F \in [2, 6]$ in $D \in [1, 3]$ spatial dimensions. Inserting the averaged $F = 3 \pm 1$ and $D = 2 \pm 1$ into N_{single} yields an averaged $N_{\text{single}} \approx (84 \pm 40)$ computing elements per spatial degree of freedom (grid point) required for implementing Euler equations.

Unfortunately, this circuit is too big to fit on the *Model-1* computer ressources available. Therefore the following text is based on a future implementation using a large number of interconnected analog chips. It is noteworthy that this stage of integration is neccessary to implement large scale analog computing applications. With $P_N = 4\,\text{mW}$ per computing element, the average power per spatial degree of freedom (*i.e.*, single grid point) is $P_{ND} = (336 \pm 160)\,\text{mW}$.

4.4 Time to solution

Numerical PDE Solvers are typically benchmarked in a *wall-clock time per degree of freedom update* measure T_{DOF}, where "element update" typically means a time integration timestep. In this measure, the overall wall clock time is normalized (divided) by the number of spatial degrees of freedom as well as the number of parallel processors involved.

The fastest digital integrators found in literature carry out a time per degree of freedom update $T_{\text{DOF}} = 10^{1\pm 1}\,\mu\text{s}$. Values smaller then $1\mu\text{s}$ require already the use of sophisticated communication avoiding numerical schemes such as discontinuous Galerkin (DG) schemes[6]. For instance, Dumbser et al. (2008) demonstrate the superiority of so called $P_N P_M$ methods (polynomial of degree N for reconstruction and M for time integration, where in the limit $P_0 P_M$ is standard high-order finite volume) by reporting $T_{\text{DOF}} =$

---

[6]$h - p$ methods, which provide both mesh refinement in grid spacing h as well as a "local" high order description typically in some function base expansion of order p. For reviews, see for instance Cockburn and Shu (2001) or Shu (2016).
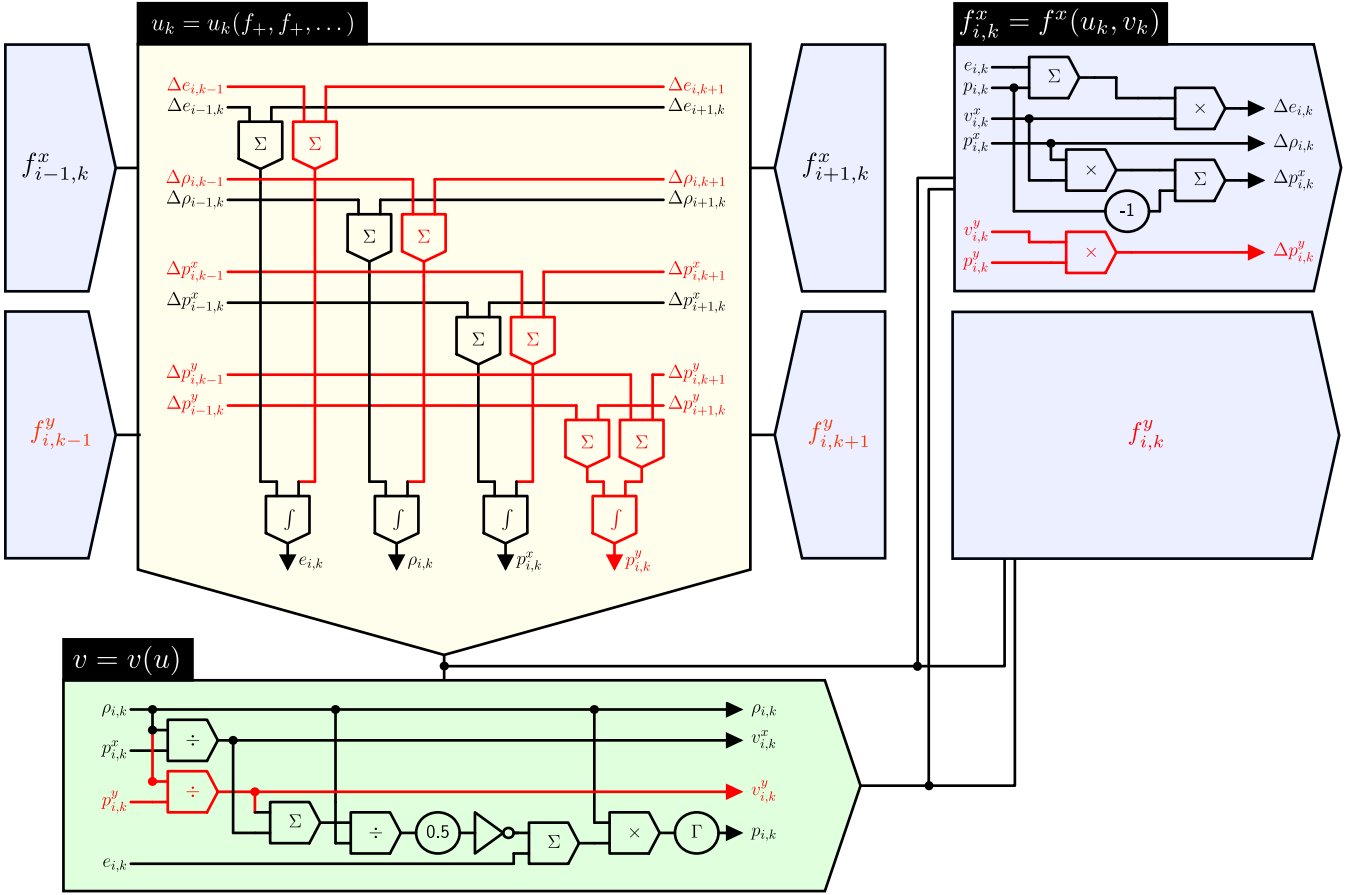
Figure 1. Overview circuit showing the blocks $\boldsymbol{f}$, $\boldsymbol{u}$ and $\boldsymbol{v}$. The three labeled blocks are distinguished by colour. Information flow is indicated with arrows. The overall circuit is given for lowest order (RK1) and in one spatial dimension. The red circuitry is the required addition for two spatial dimensions. All computing elements are drawn "abstractly" and could be directly implemented with (negating) operational amplifiers on a very large *Model-1* analog computer.

0.8 μs for a P_2P_2 method when solving two-dimensional Euler equations. Diot et al. (2012) report an adaptive scheme which performs not faster than $T_{\text{EU}} = 30$ μs when applied at three-dimensional Euler equations. The predictor-corrector arbitrary-order ADER scheme applied by Köppel (2018) and Fambri et al. (2018) on the *general-relativistic magnetodynamic* extension of hydrodynamics is also reported at $T_{\text{DOF}} = 41$ μs the fastest. The non-parallelizable evaluation of more complex hydrodynamic models is clearly reflected in the increasing times T_{DOF}.

Recalling the benchmarking result of $T_{\text{DOF}} \sim 45$ ns from section 3.1, the factor 1000 however is mainly caused by the inevitable communication required for obtaining neighbor values when solving $f(y, \nabla y)$ in $\partial_t y = f(y)$. Switched networks have an intrinsic communication latency and one cannot expect T_{DOF} to shrink significantly even for newer generations of supercomputers. Here is a key advantage of analog computing where grid neighbor communication happens continuously in the same time as in the grid-local circuit. That is, no time is lost for communication.

One can do a comparison with the analog computer without knowing the simulation time step size Δt. The reasoning is based on the cutoff frequency, *i.e.*, the shortest wavelength which can be resolved with a (first order in time[7]) numerical scheme is $f_{\text{sim}} := 1/(10\Delta t)$, *c.f.*, figure 2. The prefactor $10 = 2 \cdot 5$ includes a factor 2 due to the Nyquist-Shannon sampling theorem, while the factor 5 is chosen to take into account that a numerical scheme can marginally reconstruct a wave at frequency $f = 1/(2\Delta t)$ by two points while it can be displayed perfectly on the analog computer (down to machine precision without any artifacts). The integration of waves beyond the cutoff frequency results in a nonlinear response which heavily depends on the electrical details of the circuit (which are beyond the scope of the analog computer architecture discussed in this paper). One can demand the numerical integrator time resolution to be good enough to re-

---

[7]For a high order time integration scheme, the cutoff increases formally linearly as $f_0 \sim p/(10 T_{\text{DOF}})$. That is, for a fourth order scheme, the digital computer is effectively four times faster in this comparison.
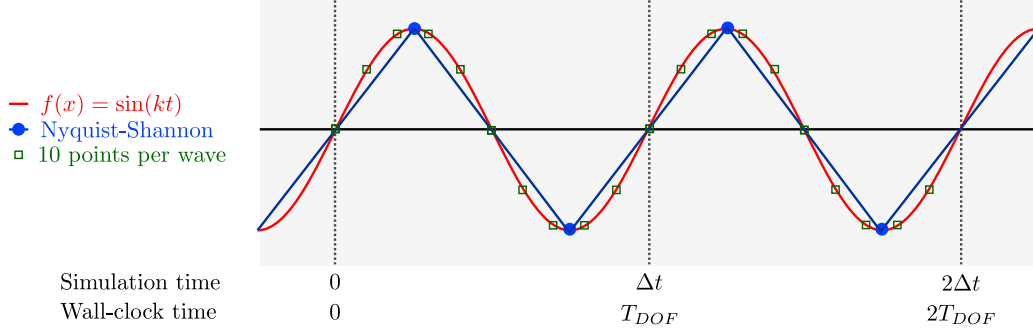
Figure 2. Analog signal sampling vs. numerical time integration: The time evolved sine with *cutoff frequency* $k = 2\pi/\lambda$ has just the wavelength $\lambda = \Delta t$, with Δt the timestep size of the explicit Euler scheme. Nyquist-Shannon theorem allows to determine wave length and phase position with two sampling points per wave length. However, a first order reconstruction of numerical data shows a zigzag function. In contrast, the full wave is clearly visible at analog integration. More sampling points close the gap between analog and numerical representation.

construct a signal *without* prior knowledge on the wave form even at the cutoff frequency. This motivates to demand for 5 additional sampling points per half-wave in order to make analog and digital outcome comparable (see also figure 2).

By assigning the numerical PDE solver the cutoff frequency identical to the highest frequency which can be evolved by the scheme in a given time, one introduces an *effective digital computer cutoff frequency* of

$$f_c^D = 1/(10T_{\text{DOF}}) = 10^{1\pm1}\,\text{khz}. \tag{9}$$

Note how the mapping of simulation time (span) Δt to wall-clock time (span) T_{DOF} results in a mapping of simulation frequency f_{sim} to wall-clock frequency f_c^D (figure 2).

The calculated $f_D = 10^{-2\pm1}\,\text{Mhz}$ has to be contrasted with $f_c^A = 100\,\text{Mhz}$ of the analog computer chip. One can conclude that analog computers can solve large scale high performance computing at least $f_c^A/f_c^D = 10^{3\pm1}$ times faster then the digital ones, when T_A and T_D are the analog and digital time to solution. Since $T \sim 1/f_c$, the resolution time reduces accordingly and $T_A/T_D = 10^{-3\pm1}$.

This is a remarkable result as it already assumed the fastest numerical integration schemes on a *perfectly* scaling parallel digital computer. In practical problems, these assumptions are hardly ever met: The impossibility of (ideal) parallelization is one of the major drawbacks of digital computing. Nevertheless, the above results show that even without these drawbacks, the analog computer is orders of magnitude faster. Notably, while it needs careful adjustment of problem and code for a high-performance computer to achieve acceptable parallel performance, when using an analog computer these advances come effortless. The only way to reduce the speed or timing advantage is to choose a distadvantageous or unsuitable number scaling.

In this study the low resolution of an analog computer (which is effectively IEEE 754 half precision floating-point) has been neglected. In fact, high order time integration schemes can invest computing time in order to archieve machine level accuracy which a typical error $\Delta f_{\text{digital}} \sim 10^{-10}$

on some evolved function or field f and an error definition $\Delta f_{\text{simulation}} := (f_{\text{simulation}} - f_{\text{exact}})/f_{\text{exact}}$. An analog computer is limited to the built-in accuracy with a typical error $\Delta f_{\text{analog}} \sim 10^{-(4\pm1)}$ (averaging over the *Model-1* and the analog chip we have in mind).

4.5 Energy and power consumption

One expects the enormous speedup T_A/T_D of the analog computer to result in a much lower energy budget $E_D = (T_D/T_A)E_A = 10^{3\pm1}E_A$ for a given problem. However, as the power requirement is proportional to the analog computer size, $P_A = NP_{ND}$, the problem size (number of grid points) which can be handled by the analog computer is limited by the overall power consumption. For instance, with a typical high performance computer power consumption of $P_A = 20\,\text{MW}$, one can simultaneously evolve a grid with $N = P_A/P_{ND} = 10^{11\pm0.5}$ points. This is in the same order of magnitude than the largest scale computational fluid dynamics simulations evolved on digital high performance computer clusters (*c.f.*, Green 500 list, Subramaniam et al., 2013, 2020). Note that in such a setup, the solution is obtained on average $10^{3\pm1}$ times faster with an purely analog computer and therefore also the energy demand is $10^{3\pm1}$ times lower.

Just to depict an analog computer of this size: Given 1000 computing elements per chip, 1000 chips per rack unit, 40 units per rack still requires 2,500 racks to build such a computer in traditional design. This is one order of magnitude larger then the size of typical high performance centers. Clearly, at such a size the interconnections will also have a considerable power consumption, even if the monumental engineering challenges for such a large scale interconnections can be met.

4.6 Hybrid techniques for trading power vs. time

The analog computers imagined so far have to grow with problem size (*i.e.*, with grid size, but also with equation

complexity). Modern chip technology could make it theoretically possible to build a computer with 10^{12} computing elements, which is many orders of magnitude larger than any analog computer that has been built so far. The idea of combining an analog and a digital computer thus forming a *hybrid computer* featuring analog and digital computing elements is not new. With the digital memory and algorithmically controlled program flow, a small analog computer can be used on a larger problem repeatedly under control of the digital computer it is mated to. Many attempts for solving PDEs on hybrid computers utilized the analog computer for computing the element-local updated state with the digital computer looping over the spatial degrees of freedom. In such a scheme, the analog computer fulfills the role of an "accelerator" or "co-processor". Such attempts are subject of various historical (such as Nomura and Deiters, 1968; Reihing, 1959; Vichnevetsky, 1968, 1971; Volynskii and Bukham, 1965; Bishop and Green, 1970; Karplus and Russell, 1971; Feilmeier, 1974) and contemporary studies (for instance Amant et al., 2014; Huang et al., 2017).

In the following, a simple back-of-the-envelope estimation with a modern hybrid computer tackling the $N = 10^{11}$ problem should be made. The aim is to trade the sheer number of computing elements with their electrical power P, respectively, against solution time T. It is assumed that the analog-digital hybrid scheme works similar to numerical parallelization: The simulation domain with N degrees of freedom is divided into Q parts which can be evolved independently to a certain degree (for instance in a predictor-corrector scheme). This allows to use a smaller analog computer which only needs to evolve N/Q degrees of freedom at a time. While the power consumption of such a computer is reduced to $P_A \to P_A/Q$, the time to solution increases to $T_A \to QT_A$. Of course, the overall required energy remains the same, $E_A = P_A T_A = (P_A/Q)(QT_A)$.

In this simple model, energy consumption of the digital part in the hybrid computer as well as numerical details of the analog-digital hybrid computer scheme are neglected. This includes the time-to-solution overhead introduced by the numerical scheme implemented by the digital computer (negligible for reasonably small Q) and the power demands of the ADC/DAC (analog-to-digital/digital-to-analog) converters (an overhead which scales with $(D+2)G^D/Q$, *i.e.*, the state vector size per grid element).

Given a fixed four orders of magnitude speed difference $f_c^D/f_c^A = 10^4$ and a given physical problem with grid size $N = 10^{11}$, one can build an analog-digital hybrid computer which requires less power and is reasonably small so that the overall computation is basically still done in the analog domain and digital effects will not dominate. For instance, with Q chosen just as big as $Q = f_c^D/f_c^A$, the analog computer would evolve only $N/Q = 10^7$ points in time, but run 10^4 times "in repetition". The required power reduces from cluster-grade to desktop-grade $P_A = (N/Q)P_{ND} =$

3.3 kW. The runtime advantage is of course lost, $T_D/T_A = (Qf_c^A)/f_c^D = 1$.

Naturally, this scenario can also be applied to solve larger problems with a given grid size. For instance, given an analog computer with the size of $N = 10^{11}$ grid points, one can solve a grid of size QN by alternatingly evolving Q parts of the computer with the same power P_A as for a grid of size N. Of course, the overall time to solution and energy will grow with Q. In any case, time and energy remain (3 ± 1) orders of magnitude lower then for a purely digital computer solution.

5 Summary and outlook

In section 2, we have shown that analog computers are *orthogonal* with respect to time and power needs to digital ones. In section 3, we performed an actual miniature benchmark of a readily available *Model-1* computer versus a mobile Intel© processor. The results are remarkable in several ways: The modern hand-crafted analog computer *Model-1*, using integrated circuit technology comparable to the 1980s, achives competitive results in computational power and energy consumption compared to a mature cutting-edge digital processor architecture, developed by one of the largest companies in the world. We also computed a problem-dependent effective FLOP/sec value for the analog computer. For the key performance measure for energy-efficient computing, namely *FLOP-per-Joule*, the analog computer obtains again remarkable results.

Note that while FLOP/sec is a popular measure in scientific computing, it is always application and algorithm specific. Other measures exist, such as transversed edges per second (TEPS) or synaptic updates per second (SUPS). Cockburn and Shu (2001) propose for instance to measure the *efficiency* of a PDE solving method by computing the inverse of the product of the (spatial-volume integrated) L^1-error times the computational cost in terms of time-to-solution or invested resources.

In section 4, large scale applications were discussed on the example of fluid dynamics and by comparing high performance computing results with a prospected analog computer chip architecture. Large scale analog applications can become power-bound and thus require the adoption of analog-digital hybrid architectures. Nevertheless, with their $\mathcal{O}(1)$ runtime scaling analog computers excel for time integrating large coupled systems where algorithmic approaches suffer from communication costs. We predict outstanding advantages in terms of time-to-solution when it comes to large scale analog computing. Given the advent of chip-level analog computing, a "gigascale" analog computer (a device with $\sim 10^9$ computing elements) could become a "game changer" in this decade.

Furthermore, in the domain of partial differential equations, there exist a number of different approaches which might be even better suited to analog computing. For in-

stance, solving PDEs with artificial intelligence has become a fruitful research field in the last decade (see for instance Michoski et al., 2020; Schenck and Fox, 2018), and analog neural networks might be an interesting candidate to tackle digital approaches. One reason why *indirect* alternative approaches such as neural networks could be better suited than *direct* analog computing networks is the no longer relevant problem of number scaling (unit scaling or number representation). Number representation on analog computers can be nontrivial when the dynamical range is large. This is frequently the case with fluid dynamics, where large density fluctiations are one reason why perturbative solutions fail and numerical simulations are carried out in the first place. Furthermore, the demand for high accuracy in fluid dynamics can not easily fulfilled by low resolution analog computing. Especially for ADCs (or rather the digital-analog interface), one bit of resolution doubles the energy consumption (Sarpeshkar, 1998). In the end, it is quite possible that a small-sized analog neural network might outperform a large-sized classical pseudo-linear time evolution in terms of time-to-solution and energy requirements. Most of these engineering challenges have not been discussed in this work and are subject to future studies.

References

Amant, R., Yazdanbakhsh, A., Park, J., Thwaites, B., Esmaeilzadeh, H., Hassibi, A., Ceze, L., and Burger, D.: General-purpose code acceleration with limited-precision analog computation, vol. 42, pp. 505–516, https://doi.org/10.1109/ISCA.2014.6853213, 2014.

Bishop, K. and Green, D.: Hybrid Computer Impelementation of the Alternating Direction Implicit Procedure for the Solution of Two-Dimensional, Parabolic, Partial-Differential Equations, AIChE Journal, 16, 139–143, 1970.

Bournez, O. and Pouly, A.: A Survey on Analog Models of Computation, CoRR, abs/1805.05729, http://arxiv.org/abs/1805.05729; http://nomura1968, 2018.

Brezis, H. and Browder, F.: Partial Differential Equations in the 20th Century, Advances in Mathematics, 135, 76–144, https://doi.org/10.1006/aima.1997.1713, http://www.sciencedirect.com/science/article/pii/S0001870897917138, 1998.

Calude, C. S., Păun, G., Tătărâm, M., (a, C. S. C., and (b, G. P.: A glimpse into natural computing, J. Multi Valued Logic, 7, 2001, 1999.

Chu, C.: Numerical Methods in Fluid Dynamics, vol. 18 of *Advances in Applied Mechanics*, pp. 285–331, Elsevier,

https://doi.org/10.1016/S0065-2156(08)70269-2, http://www.sciencedirect.com/science/article/pii/S0065215608702692, 1979.

Cockburn, B. and Shu, C.-W.: Journal of Scientific Computing, 16, 173–261, https://doi.org/10.1023/a:1012873910884, https://doi.org/10.1023/a:1012873910884, 2001.

Cowan, G., Melville, R. C., and Tsividis, Y. P.: A VLSI analog computer/math co-processor for a digital computer, ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005., pp. 82–586 Vol. 1, 2005.

Cowan, G. E. R.: A VLSI analog computer/math co-processor for a digital computer, Ph.D. thesis, Columbia University, 2005.

Dahlquist, G.: Generalized disks of contractivity for explicit and implicit Runge-Kutta methods, Tech. rep., CM-P00069451, 1979.

de Melo, A. C.: The New Linux 'perf' Tools, Tech. rep., http://vger.kernel.org/~acme/perf/lk2010-perf-paper.pdf, 2010.

Deaton, R., Garzon, M., Rose, J., Franceschetti, D., and Stevens, S.: DNA Computing: A Review, Fundamenta Informaticae, 35, 231–245, https://doi.org/10.3233/FI-1998-35123413, https://doi.org/10.3233/FI-1998-35123413, 1998.

Diot, S., Loubère, R., and Clain, S.: The MOOD method in the three-dimensional case: Very-High-Order Finite Volume Method for Hyperbolic Systems., 2012.

Dumbser, M., Balsara, D. S., Toro, E. F., and Munz, C.-D.: A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes, Journal of Computational Physics, 227, 8209–8253, https://doi.org/10.1016/j.jcp.2008.05.025, https://doi.org/10.1016/j.jcp.2008.05.025, 2008.

Fambri, F., Dumbser, M., Köppel, S., Rezzolla, L., and Zanotti, O.: ADER discontinuous Galerkin schemes for general-relativistic ideal magnetohydrodynamics, Monthly Notices of the Royal Astronomical Society, https://doi.org/10.1093/mnras/sty734, https://doi.org/10.1093/mnras/sty734, 2018.

Feilmeier, M.: Hybridrechnen, Springer, https://doi.org/10.1007/978-3-0348-5490-0, 1974.

Georgescu, I. M., Ashhab, S., and Nori, F.: Quantum simulation, Reviews of Modern Physics, 86, 153–185, https://doi.org/10.1103/revmodphys.86.153, http://dx.doi.org/10.1103/RevModPhys.86.153, 2014.

Gruber, T., Eitzinger, J., Hager, G., and Wellein, G.: LIKWID 5: Lightweight Performance Tools, https://doi.org/10.5281/zenodo.4275676, 2020.

Hager, G., Wellein, G., and Treibig, J.: LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments, in: 2012 41st International Conference on Parallel Processing Workshops, pp. 207–216, IEEE Computer Society, Los Alamitos, CA, USA, https://doi.org/10.1109/ICPPW.2010.38, https://doi.ieeecomputersociety.org/10.1109/ICPPW.2010.38, 2010.

Harten, A.: High resolution schemes for hyperbolic conservation laws, Journal of computational physics, 135, 260–278, 1997.

Hirsch, C.: Numerical computation of internal and external flows. Vol. 2-Computational Methods for Inviscid and Viscous Flows, Chichester, 1990.

Huang, Y., Guo, N., Seok, M., Tsividis, Y., Mandli, K., and Sethumadhavan, S.: Hybrid analog-digital solution of nonlinear partial differential equations, in: Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture,

ACM, https://doi.org/10.1145/3123939.3124550, https://doi.org/10.1145/3123939.3124550, 2017.

Karplus, W. and Russell, R.: Increasing Digital Computer Efficiency with the Aid of Error-Correcting Analog Subroutines, IEEE Transactions on Computers, C-20, 1971.

Kendon, V. M., Nemoto, K., and Munro, W. J.: Quantum analogue computing, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 368, 3609–3620, https://doi.org/10.1098/rsta.2010.0017, https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2010.0017, 2010.

Köppel, S.: Towards an exascale code for GRMHD on dynamical spacetimes, Journal of Physics: Conference Series, 1031, 012 017, https://doi.org/10.1088/1742-6596/1031/1/012017, https://doi.org/10.1088/1742-6596/1031/1/012017, 2018.

Michoski, C., Milosavljević, M., Oliver, T., and Hatch, D. R.: Solving differential equations using deep neural networks, Neurocomputing, 399, 193–212, https://doi.org/10.1016/j.neucom.2020.02.015, http://www.sciencedirect.com/science/article/pii/S0925231220301909, 2020.

Nomura, T. and Deiters, R.: Improving the analog simulation of partial differential equations by hybrid computation, Simulation, 1968.

Reihing, J.: A time-sharing analog computer, in: Proceedings of the western joint computer conference, 1959.

Rodgers, D. P.: Improvements in Multiprocessor System Design, SIGARCH Comput. Archit. News, 13, 225–231, https://doi.org/10.1145/327070.327215, https://doi.org/10.1145/327070.327215;http://volynskii1965, 1985.

Röhl, T., Eitzinger, J., Hager, G., and Wellein, G.: LIKWID Monitoring Stack: A flexible framework enabling job specific performance monitoring for the masses, CoRR, abs/1708.01476, http://arxiv.org/abs/1708.01476, 2017.

Sarpeshkar, R.: Analog Versus Digital: Extrapolating from Electronics to Neurobiology, Neural Computation, 10, 1601–1638, https://doi.org/10.1162/089976698300017052, https://doi.org/10.1162/089976698300017052, 1998.

Schenck, C. and Fox, D.: SPNets: Differentiable Fluid Dynamics for Deep Neural Networks, CoRR, abs/1806.06094, http://arxiv.org/abs/1806.06094, 2018.

Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S.: A Survey of Neuromorphic Computing and Neural Networks in Hardware, http://arxiv.org/pdf/1705.06963v1, 2019.

Shu, C.-W.: High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments, Journal of Computational Physics, 316, 598–613, https://doi.org/10.1016/j.jcp.2016.04.030, https://www.sciencedirect.com/science/article/pii/S0021999116300766, 2016.

Siegelmann, H. T.: Computation Beyond the Turing Limit, Science, 268, 545–548, https://doi.org/10.1126/science.268.5210.545, https://doi.org/10.1126/science.268.5210.545, 1995.

Sod, G.: Numerical Methods in Fluid Dynamics: Initial and Initial Boundary-Value Problems, Cambridge University Press, 1985.

Subramaniam, B., Saunders, W., Scogland, T., and Feng., W.-c.: Trends in Energy-Efficient Computing: A Perspective from the Green500, in: Proceedings of the International Green Computing Conference, 2013.

Subramaniam, B. et al.: Green 500 List, 2020, see http://www.green500.org, 2020.

Titarev, V. and Toro, E.: ADER schemes for three-dimensional non-linear hyperbolic systems, Journal of Computational Physics, 204, 715–736, https://doi.org/https://doi.org/10.1016/j.jcp.2004.10.028, https://www.sciencedirect.com/science/article/pii/S0021999104004358, 2005.

Titarev, V. A. and Toro, E. F.: Journal of Scientific Computing, 17, 609–618, https://doi.org/10.1023/a:1015126814947, https://doi.org/10.1023/a:1015126814947, 2002.

Toro, E. F.: Primitive, Conservative and Adaptive Schemes for Hyperbolic Conservation Laws, in: Numerical Methods for Wave Propagation, pp. 323–385, Springer Netherlands, https://doi.org/10.1007/978-94-015-9137-9_14, https://doi.org/10.1007/978-94-015-9137-9_14, 1998.

Ulmann, B.: Model-1 Analog Computer Handbook/User Manual, http://analogparadigm.com/downloads/handbook.pdf, 2019.

Ulmann, B.: Analog and Hybrid Computer Programming, De Gruyter, 2020.

Vichnevetsky, R.: A new stable computing method for the serial hybrid computer integration of partial differential equations, in: Spring Joint Computer Conference, 1968.

Vichnevetsky, R.: Hybrid methods for partial differential equations, Simulation, 1971.

Volynskii and Bukham: Analogues for the Solution of Boundary-Value Problems, Pergamon Press, library of Congress Catalog Card No 64-25643, 1965.

Wang, Y., Yu, B., Berto, F., Cai, W., and Bao, K.: Modern numerical methods and theirapplications in mechanical engineering, Advances in Mechanical Engineering, 11, https://doi.org/10.1177/1687814019887255, 2019.

Wilhelm, F., Steinwandt, R., Langenberg, B., Liebermann, P., Messinger, A., and Schuhmacher, P.: Status of quantum computer development, 2017.

Zhou, Y., Stoudenmire, E. M., and Waintal, X.: What Limits the Simulation of Quantum Computers?, Physical Review X, 10, https://doi.org/10.1103/physrevx.10.041038, http://dx.doi.org/10.1103/PhysRevX.10.041038, 2020.

Ziegler, M.: Novel hardware and concepts for unconventional computing, Sci Rep, 10, https://doi.org/10.1038/s41598-020-68834-1, 2020.