Mixed-Precision in adaptive Runge-Kutta method for large ODE systems

Mouhamad Al-Sayed, Samuel Bernard, Arsène Marzorati, Jonathan Rouzaud-Cornabas

Mixed-Precision in adaptive Runge-Kutta method for large ODE systems*

Mouhamad Al-Sayed Ali† Samuel Bernard‡ Arsène Marzorati§
Jonathan Rouzaud-Cornabas¶

May 22, 2026

Abstract

Mixed-precision methods combine low and high precision arithmetics to exploit low precision computational speed and high precision accuracy. Large ODE systems that contain many heterogeneous interactions lead to a high computational cost that could be tackled with mixed-precision solvers. We tested mixed-precision versions of the Bogacki-Shampine 3(2) Runge-Kutta pair over three benchmark systems: coupled linear oscillators, the Kuramoto model and a circadian clock model. Our study is performed in a way that can be adapted to any finite-precision format, software architecture and numerical scheme. We found that mixed-precision solvers can preserve most of the high-precision solver accuracy under a wide range of solver tolerances. Moreover, mixed-precision solver accuracy improves with system size, reaching levels equivalent to high-precision solvers in small system size. We also observed that mixed-precision arithmetic does not impact the number of evaluation in a way that balances the benefit of fast operations in low precision. Taken together, these results show that mixed-precision methods can offer significant computational speed-up at little or no loss of accuracy in large coupled ODE systems.

1 Introduction

Numerical methods for dynamical systems such as ordinary differential equations (ODEs) are widely used in many scientific fields such as biology [34], physics or engineering [24, 42] and for solving discretized partial differential equations [5]. This is why this kind of problem has been studied for a long time and numerous numerical tools for solving it have been developed [11, 21]. However, the applicability of these systems is often limited by their complexity, *i.e.* their size and connectivity. For example, in computational biology with bacteria ecosystems [35], in meteorology [30, 37] or more generally in geophysics models [1], large systems have to be solved, requiring large amount of data to be stored and manipulated.

Computational needs of large ODE systems have led to the development and specification of optimized and well-tuned solvers [12, 26, 28]. Dimension reduction techniques have been developed, such as reduced order modeling [2, 16, 32] which involves transforming the original large system into a much smaller one, or mean-field theory [3, 7, 8] which averages the effective interactions. However, while these methods are effective, they require additional assumptions, for example homogeneity, sparsity or locality in interactions. These hypotheses may be restrictive in some real cases, such as in the study of biological networks [38]. Our study addresses a more general case including fully pairwise interactions. To avoid unnecessary, constraints we explore the possibility of using mixed-precision methods for solving large coupled ODE systems. Mixed-precision methods consist in using several arithmetic precision formats within a single algorithm in order to increase computation speed while reducing the space required to store the model. Indeed, there are many formats for encoding numbers on computers and performing numerical computations. The most widespread formats for real numbers are the floating-point standards of the IEEE 754 [44] with half (16 bits), single (32 bits) and double (64 bits) precision. These finite representations influence arithmetic, accuracy and computing speed. Accuracy depends on the *machine epsilon* which is the smallest value summable to 1 and speed (storage, memory displacement and arithmetic operations) is directly impacted by the number of bits used for the encoding.

*This manuscript has been submitted to the SIAM Journal on Scientific Computing and is currently under revision.

†URMAR Univ. Rennes, Rennes, 35000, France. (mouhamad.alsayedali@univ-rennes.fr).

‡MUSICS Team, Inria Lyon, Villeurbanne, France and CNRS, Université Claude Bernard Lyon 1, ICJ UMR5208, Inria. (samuel.bernard@inria.fr).

§BioTIC and MUSICS Teams, Inria Lyon, Villeurbanne, France and INSA de Lyon, Université Claude Bernard Lyon 1, ICJ UMR5208, Inria. (arsene.marzorati@inria.fr).

¶BioTIC Team, Inria Lyon, Villeurbanne, France and INSA Lyon, Inria, CITI, UR3720, 69621 Villeurbanne, France(jonathan.rouzaud-cornabas@inria.fr).

Recent studies have demonstrated the advantages of mixed-precision methods across various scientific fields. These approaches combine different numerical precision formats to optimize both accuracy and efficiency [31]: higher-precision formats are selectively applied to the critical components of algorithms to preserve or enhance accuracy [13], while lower-precision formats help reduce computational and storage costs due to their smaller bit sizes [22].

Mixed-precision approaches have been particularly successful in linear algebra, where algorithms such as GMRES and LU factorization have been effectively re-implemented [23]. They are also widely used in deep learning, enhancing both training [33] and inference phases through quantization of weights and activations [39]. Beyond these areas, mixed-precision computing has found applications in computational fluid dynamics [9], climate forecasting [40], and other scientific disciplines [27].

Of the many numerical schemes available [21], few studies have proposed to adapt the algorithms with mixed-precision. In [10, 20], *Grant and al.* develop a complete framework for implicit additive Runge-Kutta (RK) methods. While *Croci and al.* make a complete study of the accuracy for mixed-precision versions of stabilized explicit Runge-Kutta-Chebyshev methods in [15].

In this work, we study the impact of mixed-precision on the accuracy when solving nonlinear ODE systems. Contrary to previous work, we choose another approach working with an adaptive explicit scheme. The stability criterion for explicit schemes is closely linked to the size of the discretization time step. This parameter can be set before computing the solution, or it can be adjusted at runtime to control the stability of the numerical scheme and limit the number of operations. The Bogacki-Shampine 3(2) pair [6] was designed to optimize the intersection of the stability regions of its two low order RK schemes and limit the computational cost by using embedded schemes and the First Same As Last (*FSAL*) property. For its simplicity, efficiency and economical aspects, the study is conducted on this numerical scheme.

We first present the general form of ODEs system which could take advantage of mixed-precision methods. Next, three benchmarks satisfying this framework are introduced: (1) linear oscillators with a linear coupling term, (2) the Kuramoto system [29], a standard representation of synchronization with a non-linear interaction term, and (3) a circadian clock model (Section 2.1). The latter combines a version of the Goodwin model [19] describing a negative feedback oscillators in cellular system, such as circadian rhythms, and a version of the FitzHugh-Nagumo [18, 36] model which was originally developed to describe the electro-physiology of neurons. We use it here to describe the balance between different protein complexes driving the sharp transition to mitosis during the cell cycle. We then briefly recall the structure of the Bogacki-Shampine 3(2) pair and detail the introduction of mixed-precision into the scheme (Section 2.2). We set how low precision is distributed inside the system evaluation and among the stages of the numerical scheme. We implement in Matlab four versions of the numerical scheme, one in single, one in double precision and two in mixed-precision (Section 2.3). And finally we define the metrics used to exploit the results of the different solvers on the benchmarks (Section 2.4). In particular, we introduce a performance proxy as run-times on Matlab are not representative of actual performance. Indeed, computations in Matlab are enforced to satisfy the required arithmetic precision, but this entails a loss of global performance (Section 3.1). We organise the results into four main arguments. First, we focus on how to include mixed-precision into the numerical scheme. The distribution and number of operations performed in low precision leverage performance. They necessarily modify the potential acceleration of a faster computation, but they also affect the number of steps computed by the solver (Section 2.5). Second, we look at accuracy with respect to mixed-precision and system sizes. Mixed-precision solvers appear to be more robust to tolerance constraints than the single precision one and, in some cases, can reach an accuracy similar to that of the double precision solver (Section 3.2). In addition, all solvers benefit from the increase in system size. Finally, we show that the error control fails for the single precision solver at smaller tolerances, whereas the mixed-precision solvers better match the expected theoretical behaviour even at strict tolerances (Section 3.4).

2 Methods

In this section, we present the general framework and three ODE systems which are used as benchmarks. Then, we introduce the numerical scheme applied on the systems, and we explain how we insert the mixed-precision in its calculations. We end this section with a presentation of the metrics and the performance proxy.

2.1 ODE system: General model and benchmarks

General model We consider a system describing a population of N heterogeneous agents, with $N \gg 1$, described by the variable $X \in \mathbb{R}^{dN}$. Each agent is identified with the variable $\mathbf{X}_i$ in $\mathbb{R}^d$, considering the following extraction $\mathbf{X}_i = \left(X_{(i-1)d+1}, ..., X_{id}\right)$. The dynamical dependence is split into two parts, an agent-centered term $F_i : \mathbb{R}\times\mathbb{R}^d \to \mathbb{R}^d$ and a term $G_{ij} : \mathbb{R}\times\mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}^d$ accounting for complex pairwise interactions,

which are weighted by a vector $M_{ij} \in \mathbb{R}^d$. The state of each agent follows the differential equation:

$$\dot{\mathbf{X}}_i(t) = F_i(t, \mathbf{X}_i) + \sum_{j=1}^{N} M_{ij}(t, \mathbf{X}_i) \odot G_{ij}(t, \mathbf{X}_i, \mathbf{X}_j),\ i \in 1, ..., N, \tag{1}$$

where $\odot$ is the Hadamard product (or element-wise product). There is no additional assumption on the form of G_{ij} except that all the N agents can fully interact heterogeneously, leading to an evaluation of N^2 terms in the right-hand side.

We build the agent-centered term $F : \mathbb{R} \times \mathbb{R}^{dN} \to \mathbb{R}^{dN}$ for the whole population by extending the functions F_i blockwise. In the same way, the interaction terms G_{ij} and M_{ij} extend to a population interaction function $G : \mathbb{R} \times \mathbb{R}^{dN} \to \mathbb{R}^{dN \times N}$ and an array of vectors of weights $M \in \mathbb{R}^{dN \times N}$. We denote as a convolution product the weighting of the interactions ($[M * G] : \mathbb{R} \times \mathbb{R}^{dN} \to \mathbb{R}^{dN}$). The system of equations describing the general framework for the whole population is

$$\dot{X}(t) = F(t, X) + [M * G](t, X). \tag{2}$$

Now, we present three benchmarks conforming to Equation (2). For each benchmark, we explicitly define the functions F_i and G_{ij} and the weight vector M_{ij}. All the parameter values are listed in Tables 4 to 6 of Section A.

Coupled linear oscillators (Benchmark 1) Benchmark 1 is a system of harmonic oscillators coupling by a linear term.

$$\begin{aligned} F_i^{LCO} &: \mathbb{R}^2 \to \mathbb{R}^2 & G_{ij}^{LCO} &: \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2 \\ \mathbf{X}_i &\mapsto \begin{pmatrix} \mathbf{X}_{i,2} \\ -\mathbf{X}_{i,1} \end{pmatrix} & (\mathbf{X}_i, \mathbf{X}_j) &\mapsto \begin{pmatrix} \mathbf{X}_{j,1} - \mathbf{X}_{i,1} \\ 0 \end{pmatrix} \end{aligned} \tag{3}$$

$$M_{ij} = (\frac{1}{N}, 0)^T$$

System Equation (3) posses an analytic solution.

Kuramoto model (Benchmark 2) Benchmark 2 is the classic Kuramoto model [29].

$$\begin{aligned} F_i^{Kur} &: \mathbb{R} \to \mathbb{R} & G_{ij}^{Kur} &: \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\ \mathbf{X}_i &\mapsto \omega_i & (\mathbf{X}_i, \mathbf{X}_j) &\mapsto K \sin(\mathbf{X}_j - \mathbf{X}_i) \end{aligned} \tag{4}$$

$$M_{ij} = \frac{1}{N}$$

Parameters are: ω_i, the natural frequency of the ith oscillator, and K, the coupling constant. The ω_i are generated with a centered normal distribution with standard deviation σ. σ and K are changed for each test respectively between $[0, 1]$ and $[0, 3\sigma]$ (Section A.2).

Circadian clock/cell cycle model (Benchmark 3) Benchmark 3 combines two nonlinear oscillators coupled through a nonlinear term.

$$\begin{aligned} F_i^{CC} &: \mathbb{R}^4 \to \mathbb{R}^4 \\ \mathbf{X}_i &\mapsto \left(\frac{k_0 \theta^h}{\theta^h + \mathbf{X}_{i,2}^h} \left(a\mathbf{X}_{i,1}^2 + 1 \right) - k_1 \mathbf{X}_{i,1},\ k_2(\mathbf{X}_{i,1} - \mathbf{X}_{i,2}), \right. \\ &\qquad \left. \mathbf{X}_{i,3}\left(1 - \frac{\mathbf{X}_{i,3}^2}{3}\right) - \mathbf{X}_{i,4} + \mathrm{I}_0\left(1 - \frac{k_3^2}{k_3^2 + \mathbf{X}_{i,1}^2}\right),\ \epsilon\big(\mathbf{X}_{i,3} + b - c\mathbf{X}_{i,4}\big) \right)^{\mathrm{T}} \\ G_{ij}^{CC} &: \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}^4 \\ (\mathbf{X}_i, \mathbf{X}_j) &\mapsto \left(\tfrac{k_0 \theta^h a}{\theta^h + \mathbf{X}_{i,2}^h} K \arctan(\mathbf{X}_{j,1} - \mathbf{X}_{i,1}),\ 0,\ 0,\ 0 \right)^{\mathrm{T}} \end{aligned} \tag{5}$$

$$M_{ij} = (\frac{1}{N}, 0, 0, 0)^T$$

Parameter values are fixed for $k_0 = 2$, $k_2 = 0.144832$, $k_3 = 2$, $a = 2$, $b = 0.7$, $c = 0.8$, $h = 4$, $\epsilon = 0.228249$. Parameter $k1$ takes random value following a normal distribution with mean $\bar{k}_1 = 0.339278$ and standard deviation $\sigma_1 = 0.090909$. The parameter $\theta^h = \frac{k_1}{k_0 - k_1}$. The coupling coefficient K and the stimulus I_0 are changed for each test (Section A.3).

2.2 Numerical scheme calculations and mixed-precision

The Bogacki-Shampine 3(2) pair We chose an adaptive scheme with embedded Runge-Kutta methods [6]. It combines a third order method, with three stages, and a second order one with four stages Table 1. At the *n-th* step of the scheme we denote by X^n the third order solution and $\tilde{X}^n$ the second order one. To get the solutions at the next step, the following four stages (K_ℓ) are computed:

$$K_\ell = F\big(t + c_\ell h, X^n + h\sum_{m=1}^{\ell-1} a_{\ell m} K_m\big) + [M * G]\big(t + c_\ell h, X^n + h\sum_{m=1}^{\ell-1} a_{\ell m} K_m\big),\ \ell = 1, 2, 3, 4. \tag{6}$$

The solutions at the next step $t + h$ are given by:

$$X^{n+1} = X^n + h\sum_{\ell=1}^{3} b_\ell K_\ell.$$

$$\tilde{X}^{n+1} = X^n + h\sum_{\ell=1}^{4} \tilde{b}_\ell K_\ell.$$

c_ℓ	$a_{\ell 1}$	$a_{\ell 2}$	$a_{\ell 3}$	
0				
$1/2$	$1/2$			
$3/4$	0	$3/4$		
1	$2/9$	$1/3$	$4/9$	
b_m	$2/9$	$1/3$	$4/9$	0
$\tilde{b}_m$	$7/24$	$1/4$	$1/3$	$1/8$

Table 1: *Butcher table for the Bogacki-Shampine 3(2) pair [6].*

The fourth stage, used only in the second order solution, corresponds to the evaluation of the first stage in the next step. Thus, if the step is accepted, the first stage is already computed (First Same As Last method). This method minimizes the computational cost by two aspects: the embedded stages of the schemes and the FSAL property. The latter applies only when a step is accepted, otherwise the first stage of the next step (stage K_4) must be recomputed while the current step is rejected.

Local relative error estimation The specificity of adaptive solvers is the modification of the step size (h) to control the local error, *i.e.* the error introduced in a single step. The step is accepted only if the approximation of the local error meets a tolerance criterion. This approximation is computed as the difference between the two numerical solutions X^{n+1} and $\tilde{X}^{n+1}$. The tolerance criteria are defined with the absolute (Ab) and relative (Rel) tolerances.

The relative local error approximation is done with a normalization vector of weights W which corresponds to the norm of the higher order solution vector and enables to switch from relative to absolute tolerance criterion if this norm of the solution is too small. Then, using the stages (see Equation (6)), we compute the difference vector V_{BS} between the two solutions of the embedded schemes. This difference is weighted with W. Finally, the local relative error, E^n_{BS}, is approximated with the maximum of the absolute value of V_{BS} elements:

$$\begin{aligned} W_k &= \max(|X^n_k|, |X^{n+1}_k|, \tfrac{Ab}{Rel}), && k \in \{1, ..., dN\}. \\ V_{BS,k} &= \frac{|X^{n+1}_k - \tilde{X}^{n+1}_k|}{W_k}, && k \in \{1, ..., dN\}. \\ E^n_{BS} &= \max_k(V_{BS,k}). \end{aligned} \tag{7}$$

We chose the metrics for Equation (7) based on published specifications [41], but others choices are possible [21, Section II.4].

The step is accepted if and only if the local relative error (E^n_{BS}) is smaller than the relative tolerance. The error approximation is also used within the correction of the step size. We recall that a numerical scheme of order p has a local truncation error of magnitude $O(h^{p+1})$. Here, if a step is accepted, the solution of the third order ($p = 3$), X^{n+1} is taken as the solution of the numerical scheme.

Failure conditions We add the following failure conditions. Numerical simulation stop if any of those conditions is satisfied:

- Maximal solving time: runtime over 1.5 hour.
- Maximal number of iterations: failed and successful steps $> I_{max} = 10^5$.
- Maximal number of failed steps: number of failed steps $> 0.85\ I_{max} = 8.5 \times 10^4$.
- Slow solver: runtime over 45 min and less than 10% of the simulation completed.
- Step size too small: To prevent the solver from reaching the magnitude limit of the finite-precision format we add the following condition $h > h_{min} = 100\epsilon_m$ where ϵ_m is the machine epsilon (the smallest positive value summable to 1).

2.3 Mixed-precision for stage computations

We introduce the mixed-precision into the stage computations (Table 2). For this study we only mix single (S) and double (D) arithmetic precision formats from the IEEE 754 standard [44]. We make the following choices:

- All the operations coming from the linear combinations of the numerical stages are performed in high precision (D).
- For all mixed-precision solvers, the solution is stored in high precision (D).
- For a given stage we can choose different precision for the evaluation of G_{ij}, F_i and the average of the interactions (G_{ij}), *i.e.* three arithmetic precision can be chosen for one stage.
- In the case of accumulation in high precision (D) of low precision (S) terms, low precision terms are cast into high precision before accumulation.
- We used safeguard casts to ensure the adequate precision of the computations. Consequently, the real performance does not represent the theoretical one.

We use two versions of mixed-precision distribution inside the numerical scheme (denoted Mixed1 and Mixed2). In addition to Mixed1 and Mixed2, we also compute solutions with mono-precision solvers: single precision (Single), and double precision (Double).

	Single			Mixed1			Mixed2			Double		
Stage	F	$\sum$	G	F	$\sum$	G	F	$\sum$	G	F	$\sum$	G
k_2	S	S	S	S	S	S	D	D	S	D	D	D
k_3	S	S	S	S	S	S	D	D	S	D	D	D
k_4	S	S	S	D	D	S	D	D	S	D	D	D

Table 2: *Mixed-precision distribution among the Bogacki-Shampine 3(2) pair. F corresponds to the precision used for the evaluation of F_i, $\sum$ for the accumulation of the interaction terms and G for the interaction G_{ij}. S stands for single precision and D for double precision.*

2.4 Error metric

We present three metrics used for assessing the accuracy of the four solvers. These metrics are defined for tests that are terminated by all solvers, *i.e.* the solution covers the whole integration interval. If one solver did not finish a test due to any reason among the stop criteria (see paragraph "Failure conditions" in Section 2.2), then this test is excluded from analysis for all solvers.

Normalized final error A reference solution, X_{ref}, is computed in double precision with the Matlab solver `ode45` which uses the 5(4) Dormand-Prince pair [41] with a 10^{-9} relative tolerance. The normalized final error is

$$||X_{ref}(t_f) - X(t_f)||_X = \frac{||X_{ref}(t_f) - X(t_f)||_2}{\sqrt{N}}, \tag{8}$$

where t_f is the final time in the test and N the number of agents (see Equation (2)). The normalization by the size of the system enables to facilitate the comparison of the error on each element with respect to the theoretical error and for different sizes.

Real local error For Benchmark 1 (Equation (3)), it is possible to compute an analytic solution and evaluate the real error. The real local error is the difference between analytic solution and the numerical solution after one time step.

$$E^n_{analytic} = \max_k \Big(\frac{|X^{n+1}_k - X_{k,ex}(X^n, h_n)|}{max\big(|X_{k,ex}(X^n, h_n)|, \delta\big)} \Big), \quad \text{with } h_n = t_{n+1} - t_n,\ k \in \{1, ..., dN\}, \tag{9}$$

where X_{ex} is the analytic solution at time h_n with X_n as initial condition and computed in high precision. As the initial condition for the analytic solution is always updated to the last step solution, the error accumulated in previous steps have no impact on the calculation. We took a safeguard $\delta = \frac{Abs}{Rel}$, ratio of tolerances in order to stick to the local error approximation computed by the solver, see Equation (7).

2.5 Performance metric

The real performance of a mixed-precision method is based on the expected computational speed-up per computation unit (function evaluation, or arithmetic operations), times the number of units to be computed. The speed-up associated with low-precision includes memory movement and arithmetic operations which are the two main sources of numerical performance bottleneck, and is implementation and architecture-dependent. When using adaptive schemes, the number of units to compute depends on the solution accuracy through the number of steps, and ultimately on arithmetic precision. To reflect the constraints we introduce a performance metric that depends on an expected speed-up coefficient r, and the total number of numerical steps used to solve the problem.

Let t_h and t_l be the times, in high and in low precision, covering the costs for one floating-point operation as addition or multiplication, including the memory movement of the data, reading, operation itself and results writing. We define the time ratio with respect to the highest precision (t_h), used as reference:

$$r := \frac{t_l}{t_h}. \tag{10}$$

We assume that there is an expected speed-up for floating-point operation when using low arithmetic precision instead of high one [4]. This speed-up may come from several hardware aspects. Smaller formats are faster to compute, to access and move in memory. In addition, optimizations such as vectorization or SIMD (Single Instruction, Multiple Data) are more efficient due to the smaller memory size of lower precision formats. This leads to:

$$0 < r < 1.$$

We denote the total computing times T_h and T_m, in high precision and in mixed-precision respectively. We also defined their ratio as follows:

$$\Gamma := \frac{T_m}{T_h}. \tag{11}$$

As a measure of performance we look at the number of floating-point operations. We introduce Θ, the total number of operations realised in one solver step. It is composed of the evaluation of the ODE system function, Θ_{ev} and the numerical scheme combinations Θ_{sch}.

$$\begin{aligned}
\Theta_{ev} &= \theta_F(d)N + \theta_G(d)N^2 + \theta_w(d)N^2, \\
\Theta_{sch} &= \theta_{sch}(s)\ dN, \\
\Theta &= s\ \Theta_{ev} + \Theta_{sch} = sN^2(\theta_G(d) + \theta_w(d)) + (s\theta_F(d) + d\theta_{sch}(s))N,
\end{aligned}$$

where

- $\theta_F(d)$ and $\theta_G(d)$ are the numbers of floating-point operations required for evaluating respectively the functions F_i and G_{ij} (Section Section 2.1). We emphasise that the number of operations is impacted here by the dimension of the agent d and not of the size of the population N.
- $\theta_w(d)$ is the number of floating-point operations for the weighting of the interactions (Hadamard product). The factor N^2 accounts for the summation of all the interactions for every agent.
- s is the number of stages required by the numerical scheme and $\theta_{sch}(s)$ the number of linear combinations between the stages inside the numerical scheme (for a scalar problem).
- It can be seen that for large N, $\Theta \sim sN^2$. We assume that the majority of the computational cost lies in the stage evaluation, especially in the interaction terms.

For the whole computation this has to be multiplied by the number of steps required. We introduce the mixed-precision influence with 3 parameters: ϱ, β and γ. The first one is the fraction of operations performed in lower precision. The second, β, is the ratio between the number of steps (successful and failed are included) computed in high precision and in mixed-precision, so it takes into account impacts on the global integration trajectories indicated by the successful step number and also the efficiency of the adaptive process through the failed step number[1]. The last one, γ is the ratio of computing times for one step between mixed-precision solver and high precision solver which include the memory movements and the mathematical operations. We can express the ratio Γ between the computing times in mixed-precision and high precision as the ratio between γ and β:

$$
\begin{aligned}
T_h &= N_{step,h}\Theta\, t_h, \\
T_m &= N_{step,m}\Theta(\varrho t_l + (1-\varrho)t_h), \\
\beta &= \frac{N_{step,h}}{N_{step,m}}, \qquad (12)\\
\gamma &= \frac{\varrho t_l + (1-\varrho)t_h}{t_h} = \varrho r + (1-\varrho), \qquad (13)\\
\Gamma &= \frac{T_m}{T_h} = \frac{\gamma}{\beta}. \qquad (14)
\end{aligned}
$$

A ratio $\Gamma < 1$ means that the mixed-precision solver is faster than the high precision solver.

2.6 Descriptive statistics

For each representation of the results, we select only tests that have been finished by all the solvers (Single, Mixed1, Mixed2, Double and the reference solver). We use some statistical quantities:

- The median as a reference value with 5-th and the 95-th percentiles as error bars for representing the evolution of the normalized final error against other variables (size and relative tolerance).
- The first, the second and the third quartiles for the box plots with the 1-st and the 99-th percentiles as whiskers.
- For figures with local error, we have values for each time step for every selected test. We take the averaged value of the local errors ($E^n_{analytic}$ and Err^n_{BS}). In that way, we have one value for each variable by test. In the figures we plot the median, with 5-th and 95-th percentiles as error bars for the averaged variables (approximated and real local error).

3 Results and discussion

All the code is available in a public repository on Inria's GitLab[2].

3.1 Application of the performance metric

First, we look at the distribution of β (Equation (12)) among the finished tests (see Figure 4 Section B, for the number of failed tests), *i.e.* the ratio of number of steps between Double and the mixed-precision solvers.

	Benchmark 1	Benchmark 2	Benchmark 3
Single	(0.98, 1.0, 1.0, 1.0, 1.026)	(0.427, 0.997, 1.0, 1.0, 1.153)	(0.979, 1.0, 1.0, 1.0, 1.017)
Mixed1	(0.987, 1.0, 1.0, 1.0, 1.015)	(0.482, 0.997, 1.0, 1.0, 1.118)	(0.986, 1.0, 1.0, 1.0, 1.017)
Mixed2	(0.989, 1.0, 1.0, 1.0, 1.016)	(0.507, 0.998, 1.0, 1.0, 1.103)	(0.996, 1.0, 1.0, 1.0, 1.003)

Table 3: *Distribution of β parameter for each solver by benchmark. Each vector is composed as (min, 1st quartile, median, 3rd quartile, max) of β values for one solver under one benchmark. The statistics are done over the tests that have been finished by all the solvers.*

Table 3 shows indicative values (minimum, 1st quartile, median, 3rd quartile and maximum) of the β parameter. The first remark is that for all benchmarks, all solvers perform as many steps as the Double solver

[1] If β is far from 1, one will not directly see if it is due to a very different integration trajectory or to a larger number of failed steps.

[2] Link to Inria's Gitlab repository: https://gitlab.inria.fr/amarzora/mixed-precision-adaptive-solver.

in the majority of the tests. In a very few tests and only for Benchmark 2 the ratio is far from 1. Indeed, for all the benchmarks the 1st and third quartiles are very close or equal to 1.

As the number of steps for an adaptive scheme is not fixed before the solving, we want to estimate if the difference between a mixed-precision and double precision solver in terms of supplementary steps is not large enough, $\beta \ll 1$, to balance the computing speed-up of low precision ($\gamma < 1$). In this situation, the ratio of the total computing time between mixed and high precision, Γ, is maintained under 1, *i.e.* a better performance for the mixed-precision solver. That is expressed by the following relation between the parameters:

$$\Gamma = \frac{\gamma}{\beta} < 1 \Leftrightarrow \gamma < \beta.$$

A better performance is observed for all tests only if the lowest β is higher than γ. If the ratio r between the computing time for floating-point operations in low and high precision (Equation (10)) was fixed and not impacted by any internal supplementary cost such as casting of variable, we would be able to estimate the value of γ (Equation (13)). That is why we consider a "naive" approach whereby that single precision operation is twice as fast as a double operation. For each mixed-precision solver, we estimate the fraction of low precision operations, ϱ_k. We choose to perform all the operations of the scheme, Θ_{sch}, in high precision. Considering that systems are large ($N \gg 1$), we compute a limit value ϱ_k as follows (for each version, refer to Table 2):

$$\text{Version 1 (Mixed1): } \varrho_1 = \frac{2\Theta_{ev} + N^2\theta_G(d)}{\Theta}, \qquad \lim_{N\to+\infty} \varrho_1 = 1 - \frac{\theta_w(d)}{3(\theta_G(d) + \theta_w(d))}.$$

$$\text{Version 2 (Mixed2): } \varrho_2 = \frac{3N^2\theta_G(d)}{\Theta}, \qquad \lim_{N\to+\infty} \varrho_2 = 1 - \frac{\theta_w(d)}{\theta_G(d) + \theta_w(d)}.$$

Finally, we consider a limit case $\theta_w(d) \simeq \theta_G(d)$. The complexity of G_{ij} increases the value of $\theta_G(d)$ whereas the weighting of the interactions increases only with the dimension of the agent. Normally, in models of interest we would have $\theta_G(d) \leq \theta_w(d)$, leading to a higher ϱ_k. Yet, with this consideration we obtain the following values of ϱ_k and γ_k respectively for Mixed1 and Mixed2. The trivial case for Single is also indicated:

$$\begin{aligned}
\text{Version 0 (Single):} \quad & \varrho_0 = 1 \implies \gamma_0 = r = 0.5, \\
\text{Version 1 (Mixed1):} \quad & \varrho_1 = \tfrac{5}{6} \implies \gamma_1 = \tfrac{7}{12} \simeq 0.583, \\
\text{Version 2 (Mixed2):} \quad & \varrho_2 = \tfrac{1}{2} \implies \gamma_2 = \tfrac{3}{4} = 0.75.
\end{aligned}$$

γ_2 is higher than $\gamma_1 \simeq 0.583$ due to fewer operations performed in low precision.

For Benchmark 1 and Benchmark 3, the minimal values for β are very close to 1, around 0.98, for all solvers. That means in the worst case, our mixed-precision solvers perform at most 2% more steps than the Double. But as we stated before, this will not be enough to balance the gain γ_k that we estimate reachable with low precision in computational time through the r parameter.

For Benchmark 2, the minimal values are very low, and under the critical value of γ_k for all solvers. 0.43 for Single, 0.48 for Mixed1, and 0.51 for Mixed2. To keep $\Gamma \leq 1$ in these cases, the parameter r should be smaller than 0.5 (below 0.42 for Single). That is more restrictive but not impossible as the single precision takes up half as much memory space as double precision [4]. In addition, for the last two benchmarks, the G_{ij} functions are not linear. Thus, the complexity of G_{ij} functions leads to a larger θ_G, increasing the ratio of low arithmetic operations ϱ_k and lowering the time ratio for one step γ_k.

However, one should pay attention to the benchmark and the numerical scheme which are used. Indeed, our approach can be adapted to any numerical scheme, but the results will depend on the solution path taken by the solver. This can be impacted by the non-linearity and the complexity of the benchmarks but also by the choice of the numerical scheme. This is caught by β parameter, which in our case seems clearly negligible.

Another aspect is the dependence of the hardware. That is included in r and γ parameters. The r parameter includes computational performances such as vectorization, parallelization or memory access which benefit for lower formats, but r could be also negatively impacted by the cost of variable casting. We did not push the study on this aspect due to the difficulty to access how floating-point operations are performed.

As we indicated in the mixed-precision implementation, the solution is stored in high precision (D), and all the computations related to the linear combination of the scheme just as the approximation of the local error are performed in this precision. We anticipate that for large sizes, the dominant cost should arise from interaction terms, that is why we inserted low precision in their evaluation. Yet, a possible modification would be to insert low precision in the other parts of the solver in order to modify the ρ parameter and allow greater exploitation of the computational benefits captured by the r parameter.

3.2 Accuracy vs tolerance, for a fixed system size

In this section, we compare the evolution of the normalized final error (Equation (8)) with respect to the relative tolerance imposed on the solver, at fixed system size. As the tolerance is reduced, *i.e.* becomes stricter, the

normalized final error is expected to decrease. It is also expected that for very tight tolerances, the machine epsilon of the single precision format (Section 2.2) limits the error reduction for Single. In this situation, we give a closer look at Mixed1 and Mixed2 errors, and study how the presence of low precision computation limits their accuracy.

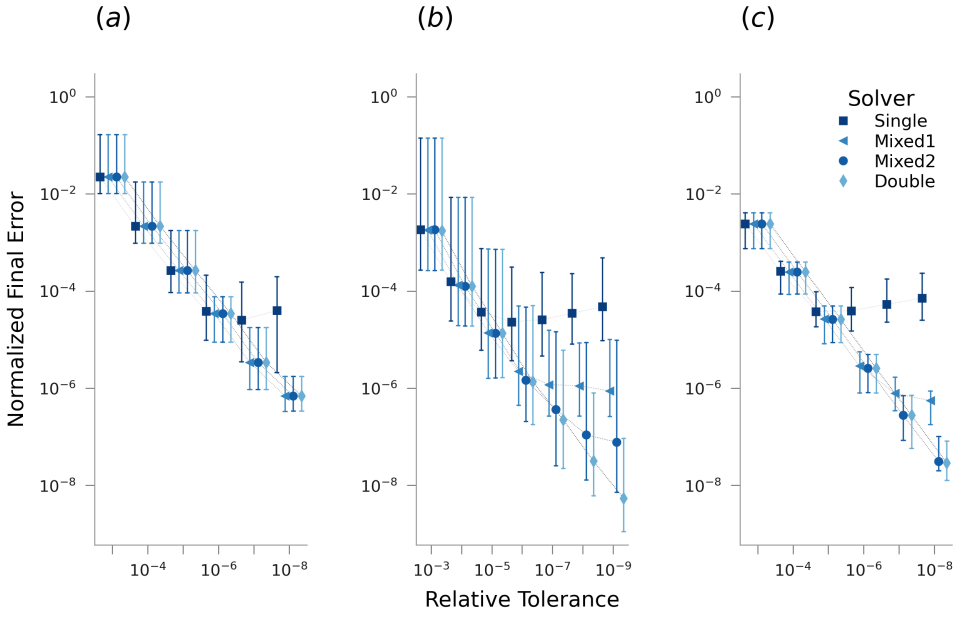

Figure 1: *Normalized final error, see Equation* (8)*, with respect to Matlab's* `ode45` *solution. 1000 agents for Benchmark 1 (a), to 2000 for Benchmark 2 (b) and to 700 for Benchmark 3 (c). The error bars are the 5-th and 95-th percentiles. For tolerances of* (10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9})*, there are respectively in (a) (113, 97, 78, 59, 33, 19, 0), in (b) (498, 423, 359, 270, 189, 122, 53) and (c) (289, 235, 199, 134, 96, 42, 0) performed tests. Each solver is identified by its symbol: Single square, Mixed1 triangle, Mixed2 circle and Double diamond. As the tolerance values tested are discrete, a small shift is added between the solvers to distinguish their symbol and error bars.*

For the 3 benchmarks and all solvers, the normalized final error decreases as the tolerance is reduced but stays at large values (10^{-3}, 10^{-4}, 10^{-5}). As expected, the normalized final error of Single (Figure 1, *squares*) plateaus for relative tolerances below 10^{-5} in all benchmarks. Indeed, the normalized final error is between 10^{-4} and 10^{-5} for the three benchmarks. Even a slight rise is visible while the relative tolerance continues to be reduced. The accuracy is limited by the machine epsilon in single precision (10^{-8}), Single solution does not satisfy the tolerance criterion in the sense that there is no difference in the error for stricter tolerances.

In contrast, errors steadily decrease in the three other solvers, albeit with noticeable differences. In the case of Benchmark 1 (Figure 1, *a*), the averaged normalized final errors coincide for all three solvers. The mixed-precision solvers have an accuracy equivalent to Double. However, even if Mixed1 (Figure 1, *triangles*) has a better accuracy than Single, its error is also limited at stricter tolerances, below 10^{-7}. The normalized final error plateaus around 10^{-6} for Benchmark 2 and Benchmark 3 (Figure 1, *b and c*), whereas Mixed2 and Double (Figure 1, *circles and diamonds*) have a normalized final error that reaches 10^{-8} and does not plateau. Yet, on Benchmark 2, Mixed2 seems to plateau from tolerances 10^{-8} and smaller, its normalized final error is better than Mixed1 error, but it is approximately 10 times larger than Double error.

These results highlight the limitations of low precision operations in terms of accuracy, especially when dealing with stringent tolerance requirements. While single precision algorithms offer computational efficiency, their limited potential for high accuracy under tight tolerances must be carefully considered in applications where accuracy is a strong criterion. Mixed-precision methods have the possibility to use the computational efficiency of low arithmetic precision, while pushing back the accuracy limit even for stringent tolerances.

3.3 System size vs accuracy, for fixed relative tolerance

In this section, we test how the system size affect the accuracy of the solution. We look at the evolution of the normalized final error (Equation (8)) with respect to the size of the system. The relative tolerance is fixed. Because we want to use mixed-precision as a lever for modeling large-scale systems, we have to check that when the scale is increased, the low precision does not become prohibitive.

The median (Figure 2, *black horizontal lines*) and the first and 99th percentiles of the normalized final error are either unchanged or tend to decrease as the system size increases. For example, with Benchmark 1,

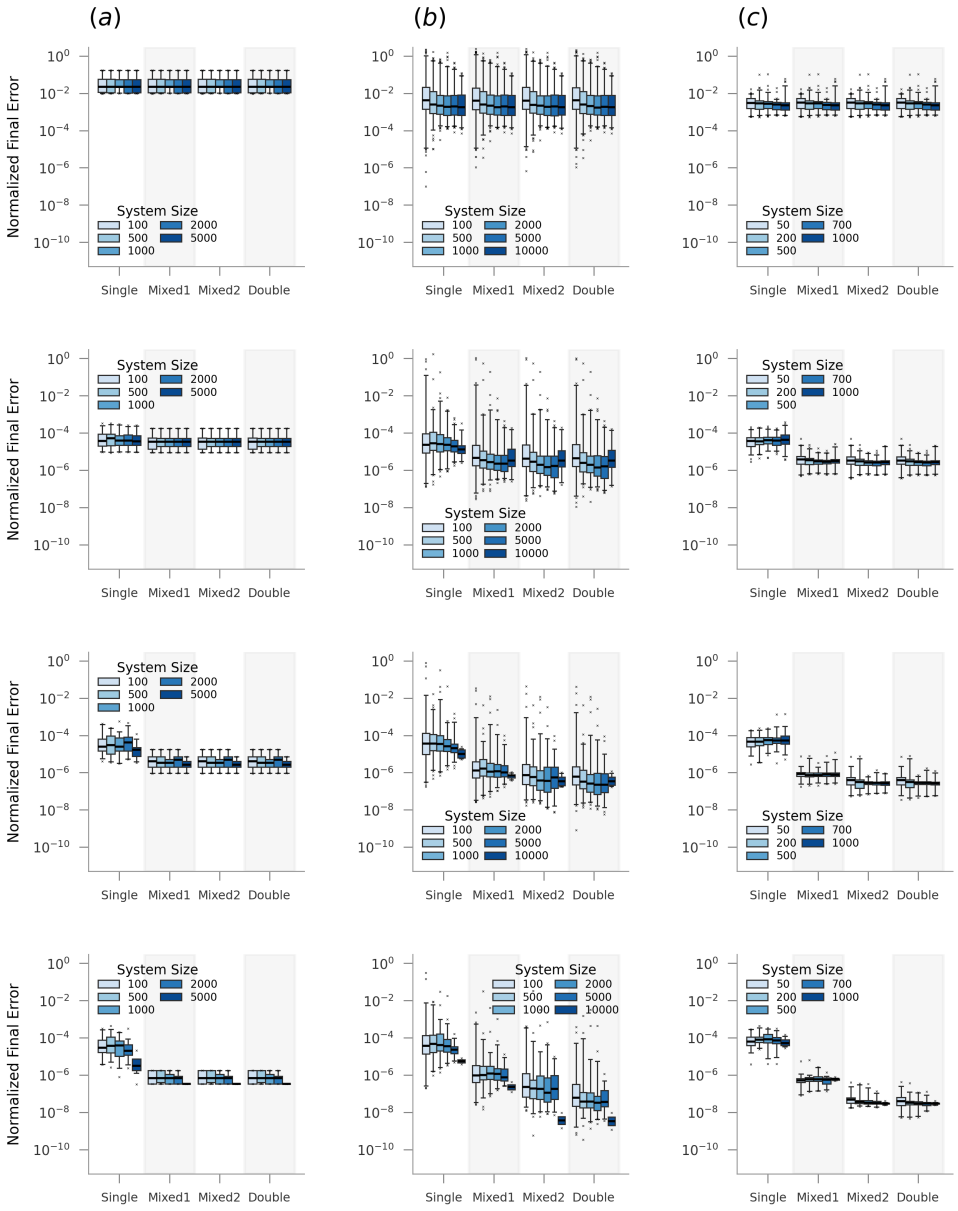

Figure 2: *Distribution of normalized final error, see Equation* (8)*, for all the tests completed by all the solvers. One box plot represents a distribution, with limits at 1-st and 99-th percentiles, for all the tests of one benchmark, indicated by the column (a) Benchmark 1, Equation* (3)*, (b) Benchmark 2, Equation* (4) *and (c) Benchmark 3, Equation* (5)*. With a unique system size, indicated by the intensity of blue, pale blue for small sizes (100 for Benchmark 1 and Benchmark 2 and 50 for Benchmark 3) to dark blue for the largest ones (5,000 for Benchmark 1, 10,000 for Benchmark 2, 1,000 for Benchmark 3). At a specific relative tolerance that can be found with the row, from top to bottom* (10^{-3}, 10^{-6}, 10^{-7}, 10^{-8})*. Run by one solver, indicated on the abscissa of the boxes group, from left to right (Single, Mixed1, Mixed2, Double).*

(Figure 2, *column a*), there is no real change as the size increases. In addition, the mixed-precision solvers have a normalized final error as accurate as the Double at every tolerance. The effect of the size is more pronounced in the case of Benchmark 2 (*column b*), at every tolerance the whiskers and the median are lowered as the size increases. For Benchmark 3 (*column c*) the decrease of the error with the increasing size is present but less pronounced.

Furthermore, for all benchmarks, Mixed2 has results closed to Double ones at every tolerance. The error obtained on larger systems by Mixed2 ends up being of the same order as the error of Double for lower sizes at same tolerance. For example, for Benchmark 2 tests solved with a 10^{-6} tolerance (*second row, column b*), the median of the normalized final error for systems of 5,000 oscillators computed with Mixed2 is lower than the Double one for tests with sizes of 500 or 1,000 oscillators at the same tolerance. Similarly, for Benchmark 3 tests solved with a 10^{-6} tolerance (*second row and column c*), the normalized final error for systems of 1,000 agents solved by Mixed1 and Mixed2 are equivalent or better than the Double normalized final error for systems of 100 agents.

The previous observation is model-dependent in the sense that is more pronounced for Benchmark 2, (Figure 2, *column b*) than for Benchmark 1, (*column a*) and Benchmark 3 (*column c*). However, either the errors are equivalent between mixed-precision solvers and high precision solver, or, with the error improvement on large systems, the mixed-precision solvers have an error equivalent to the high precision solver error for same constraint but on smaller systems. Even if there is a small loss in accuracy, as expected with lower precision, it is limited with mixed-precision solvers and tends to be reduced as the system size increases. In addition, for large systems the potential speed-up is more important, as underlined in Section 3.1, because the fraction of low precision operations increases as the size increases.

3.4 Adaptive scheme: a procedure limited by the arithmetic precision

In this section, we look at the efficiency of the error approximation in the Bogacki-Shampine 3(2) pair with respect to the arithmetic precision. For Benchmark 1, the existence of an analytic solution allows us to compare the approximate local error (E_{BS}, Equation (7)) between two numerical solutions computed with the scheme, and the real local error ($E_{analytic}$, Equation (9)) between the numerical solution and the analytic one.

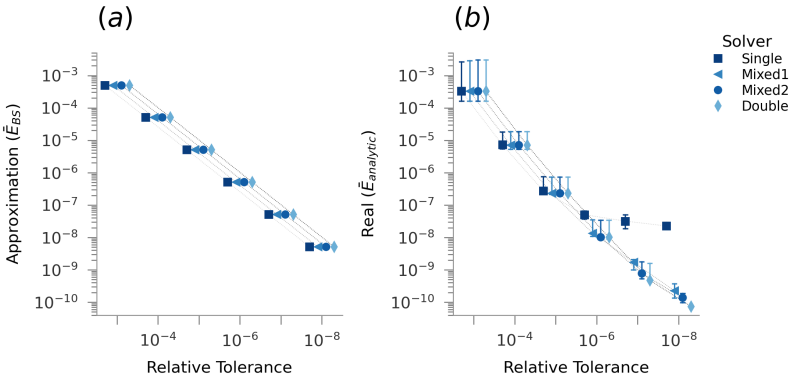

Figure 3: *(a) Averaged local error (one value per test) approximated as the difference between the 2 RK schemes (E_{BS}, in Equation (7)). (b) Averaged real local error (one value per test). In each plot the median which is computed over all the tests performed for a fixed tolerance is given with the error bars corresponding to the 5th and 95th percentiles. As the tolerance values tested are discrete, a shift was added to distinguish the points and the error bars. In (a) and (b) square is for Single, triangle Mixed1, circle for Mixed2, and diamond for Double.*

As expected, the approximated local error is reduced as the tolerance tightens (Figure 3, *a*) and its value is always lower than the relative tolerance, meaning that for all the solvers, the tolerance criteria are satisfied. However, regarding the local error computed against the analytic solution (Figure 3, *b*), the tolerance criterion is not satisfied for Single (Figure 3, *b, squares*). The averaged local error plateaus around 10^{-7} for tolerances lower than 10^{-6}. Indeed, the machine epsilon of single precision (10^{-8}) limits the real deviation between the analytic solution and the numerical one. On the contrary, mixed-precision solvers real errors (Figure 3, *b, triangles and circles*) are reduced as the tolerance is tightened. The real and the approximated errors evolve similarly to Double errors (Figure 3, *b, diamonds*), even at stricter tolerance.

The specificity of adaptive schemes lies in the step size selection procedure. The solver computes an approximated local error in order to adapt the step size to the tolerance. In (Figure 3, *a*), the procedure works for all solvers with respect to the approximation. However, the low precision impacts the real accuracy of the solution. In (Figure 3, *b*), the real local error at strict tolerance is higher than the approximation of Single and

the tolerance criterion is not satisfied. However, the mixed-precision solvers are not impacted in that way for the tested tolerances. Their accuracy is close to the high-precision solver.

4 Conclusion

In this work, we have proposed a mixed-precision version of an adaptive RK method and studied its performance through a proxy independent of the architecture and arithmetic format used. Our approach can be generalised to any similar method such as embedded Fehlberg methods [17], and extended to any type of numerical format (*e.g.* bfloat16 or fp8). We theoretically expect a performance gain that essentially depends on the ratio of operations performed in low arithmetic precision and the difference of computational cost between high and low arithmetic precision formats.

Contrary to a full low precision solver, we showed that mixed-precision solvers are an interesting balance between computational performance and accuracy. Global performance depends on the complexity of the systems induced by the agent interactions, and by the order of the scheme and the tolerance, which affect the number of time steps. For adaptive scheme, the number of steps cannot be set before solving, but we showed that the difference regarding the number of steps, between a standard solver in double precision and others in mixed-precision is not high enough to balance the speed-up of low precision (Table 3).

As the tolerance tightens, numerical convergence of RK methods implies that infinite precision solution should converge to the true solution. Yet, the low precision impacts the accuracy through round-off errors, for example Single accuracy is limited by its machine epsilon for larger tolerances (Figure 1). The real advantage is the gain of accuracy with the size. Not only mixed-precision solvers achieve finer errors at strict tolerances, but their solution accuracy is improved as the size increases (Figure 2). For large systems, the accuracy of mixed-precision solvers is equivalent to the Double error for smaller systems.

It is also important to emphasise the role of arithmetic precision in the adaptive procedure. Error control works well for coarse tolerances, but for more restrictive ones, the single precision solver is unable to achieve the required error magnitude. Indeed, the local error with respect to the analytic solution is higher than the fixed tolerance, *i.e.* the step should be rejected, but the error approximation computed by the solver is below the tolerance (Figure 3). The mixed-precision solvers are more robust and the error procedure is efficient even at strict tolerance.

Two parameters can be adjusted to get a better performance: the ratio of low precision operations over the number of performed floating-point operations within the numerical scheme, and the performance of the architecture through the r parameter that can be modified with other floating-point formats or the use of specific hardware architectures with process units (CPU, GPU) designed for mixed-precision [14]. Here, we consider only two types of precision, high (double precision) and low (single precision), which are the most widespread and available. Nonetheless, our metric can be easily extended to any floating-point format and potentially other type of formats such as fixed-point [25]. In addition, the performance proxy can be easily adapted for several native performances. It could be interesting to first investigate the runtime performance and extend this work to other numerical methods and other finite-precision formats. Indeed, computer representation of numbers requires a balance between available range, achievable precision and performance in terms of storage and computation speed. Although the floating-point formats defined by the IEEE 754 standard are the most widely used [44], others are designed to emphasise one or more of the above criterion. For example, *Google Brain bfloat16* has the same number of bits as the IEEE 754 half precision (16-bit format) but it has a range equivalent to that of the standard 32-bit single precision format [43].

References

[1] Ackmann, J., Dueben, P. D., Palmer, T., and Smolarkiewicz, P. K. Mixed-precision for linear solvers in global geophysical flows. *Journal of Advances in Modeling Earth Systems 14*, 9 (2022), e2022MS003148. Publisher: Wiley Online Library.

[2] Antoulas, A. Approximation of large-scale dynamical systems: An overview. *IFAC Proceedings Volumes 37*, 11 (2004), 19–28.

[3] Aoki, H., Tsuji, N., Eckstein, M., Kollar, M., Oka, T., and Werner, P. Nonequilibrium dynamical mean-field theory and its applications. *Reviews of Modern Physics 86*, 2 (2014), 779–837.

[4] Baboulin, M., Buttari, A., Dongarra, J., Kurzak, J., Langou, J., Langou, J., Luszczek, P., and Tomov, S. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications 180*, 12 (2009), 2526–2533.

[5] Baker, J., and Christofides, P. D. Finite-dimensional approximation and control of non-linear parabolic PDE systems. *International Journal of Control 73*, 5 (2000), 439–456.

[6] Bogacki, P., and Shampine, L. F. A 3 (2) pair of Runge-Kutta formulas. *Applied Mathematics Letters 2*, 4 (1989), 321–325. Publisher: Elsevier.

[7] Bordenave, C., McDonald, D., and Proutiere, A. A particle system in interaction with a rapidly varying environment: Mean field limits and applications. *arXiv preprint math/0701363* (2007).

[8] Bortolussi, L., and Gast, N. Mean-field limits beyond ordinary differential equations. *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems: 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2016, Bertinoro, Italy, June 20-24, 2016, Advanced Lectures 16* (2016), 61–82.

[9] Brogi, F., Bnà, S., Boga, G., Amati, G., Ongaro, T. E., and Cerminara, M. On floating point precision in computational fluid dynamics using openfoam. *Future Generation Computer Systems 152* (2024), 1–16.

[10] Burnett, B., Gottlieb, S., and Grant, Z. J. Stability analysis and performance evaluation of mixed-precision Runge-Kutta methods, Dec. 2022.

[11] Butcher, J. C. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[12] Cavaglieri, D., and Bewley, T. Low-storage implicit/explicit Runge-Kutta schemes for the simulation of stiff high-dimensional ODE systems. *Journal of Computational Physics 286* (2015), 172–193.

[13] Chiang, W.-F., Baranowski, M., Briggs, I., Solovyev, A., Gopalakrishnan, G., and Rakamarić, Z. Rigorous floating-point mixed-precision tuning. *ACM SIGPLAN Notices 52*, 1 (2017), 300–315.

[14] Choquette, J., and Gandhi, W. Nvidia A100 GPU: Performance & innovation for GPU computing. In *2020 IEEE Hot Chips 32 Symposium (HCS)* (2020), IEEE Computer Society, pp. 1–43.

[15] Croci, M., and de Souza, G. R. Mixed-precision explicit stabilized Runge-Kutta methods for single- and multi-scale differential equations. *Journal of Computational Physics 464* (Sept. 2022), 111349. arXiv:2109.12153 [cs, math].

[16] Dar, Z., Baiges, J., and Codina, R. Reduced order modeling. In *Machine Learning in Modeling and Simulation: Methods and Applications*. Springer, 2023, pp. 297–339.

[17] Fehlberg, E. *Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*, vol. 315. National aeronautics and space administration, 1969.

[18] FitzHugh, R. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal 1*, 6 (1961), 445–466.

[19] Goodwin, B. C. Oscillatory behavior in enzymatic control processes. *Advances in enzyme regulation 3* (1965), 425–437.

[20] Grant, Z. J. Perturbed Runge-Kutta methods for mixed precision applications, Dec. 2020. arXiv:2012.13055 [cs, math].

[21] Hairer, E., Norsett, S., and Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*, second revised edition ed., vol. 8 of *Springer Series in Computational Mathematics*. Springer, Jan. 1993.

[22] Hayford, J., Goldman-Wetzler, J., Wang, E., and Lu, L. Speeding up and reducing memory usage for scientific machine learning via mixed precision. *Computer Methods in Applied Mechanics and Engineering 428* (2024), 117093.

[23] Higham, N. J., and Mary, T. Mixed precision algorithms in numerical linear algebra. *Acta Numerica 31* (May 2022), 347–414.

[24] Hirsch, M. W., Smale, S., and Devaney, R. L. *Differential equations, dynamical systems, and an introduction to chaos*. Academic press, 2013.

[25] Hubrecht, T., Desrentes, O., and de Dinechin, F. Activations in low precision with high accuracy. *hal preprint hal-04776745* (2024).

[26] Ichimura, T., Fujita, K., Yamaguchi, T., Naruse, A., Wells, J. C., Schulthess, T. C., Straatsma, T. P., Zimmer, C. J., Martinasso, M., Nakajima, K., et al. A fast scalable implicit solver for nonlinear time-evolution earthquake city problem on low-ordered unstructured finite elements with artificial intelligence and transprecision computing. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis* (2018), IEEE, pp. 627–637.

[27] Kashi, A., Lu, H., Brewer, W., Rogers, D., Matheson, M., Shankar, M., and Wang, F. Mixed-precision numerics in scientific applications: survey and perspectives. *arXiv preprint arXiv:2412.19322* (2024).

[28] Kennedy, C. A., Carpenter, M. H., and Lewis, R. M. Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations. *Applied numerical mathematics 35*, 3 (2000), 177–219.

[29] Kuramoto, Y. *Chemical turbulence*. Springer, 1984.

[30] Kurth, T., Treichler, S., Romero, J., Mudigonda, M., Luehr, N., Phillips, E., Mahesh, A., Matheson, M., Deslippe, J., Fatica, M., et al. Exascale deep learning for climate analytics. In *SC18: International conference for high performance computing, networking, storage and analysis* (2018), IEEE, pp. 649–660.

[31] Le Grand, S., Götz, A. W., and Walker, R. C. Spfp: Speed without compromise—a mixed precision model for gpu accelerated molecular dynamics simulations. *Computer Physics Communications 184*, 2 (2013), 374–380.

[32] Lucia, D. J., Beran, P. S., and Silva, W. A. Reduced-order modeling: New approaches for computational physics. *Progress in aerospace sciences 40*, 1-2 (2004), 51–117.

[33] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017).

[34] Murray, J. D. *Mathematical Biology I: An introduction*. Springer, 2002.

[35] Nagarajan, K., Ni, C., and Lu, T. Agent-based modeling of microbial communities. *ACS synthetic biology 11*, 11 (2022), 3564–3574.

[36] Nagumo, J., Arimoto, S., and Yoshizawa, S. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE 50*, 10 (1962), 2061–2070.

[37] Palmer, T. N. More reliable forecasts with less precise computations: A fast-track route to cloud-resolved weather and climate simulators? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372*, 2018 (June 2014), 20130391.

[38] Paul, T. J., and Kollmannsberger, P. Biological network growth in complex environments: A computational framework. *PLOS Computational Biology 16*, 11 (2020), e1008003.

[39] Rakka, M., Fouda, M. E., Khargonekar, P., and Kurdahi, F. Mixed-precision neural networks: A survey. *arXiv preprint arXiv:2208.06064* (2022).

[40] Saffin, L., Hatfield, S., Düben, P., and Palmer, T. Reduced-precision parametrization: Lessons from an intermediate-complexity atmospheric model. *Quarterly Journal of the Royal Meteorological Society 146*, 729 (2020), 1590–1607.

[41] Shampine, L. F., and Reichelt, M. W. The Matlab ODE suite. *SIAM journal on scientific computing 18*, 1 (1997), 1–22. Publisher: SIAM.

[42] Strogatz, S. H. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[43] Wang, S., and Kanwar, P. BFloat16: the secret to high performance on Cloud TPUs. *Google Cloud Blog 4* (2019).

[44] Zuras, D., Cowlishaw, M., Aiken, A., Applegate, M., Bailey, D., Bass, S., Bhandarkar, D., Bhat, M., Bindel, D., Boldo, S., et al. IEEE standard for floating-point arithmetic. *IEEE Std 754*, 2008 (2008), 1–70.

A Parameters

For all benchmarks we have

Tolerances The absolute tolerance is lower or equal than the relative tolerance.

Explored parameters All the explored parameters are generated with the *sobolset* function. The value is then adapted to the parameter interval (see Tables 4 to 6).

A.1 Coupled linear oscillators (Benchmark 1)

Initial conditions All the initial conditions are generated in the same way. The seed is fixed by the number identifying the test. Then the function *rand* of Matlab (returns a random scalar drawn from the uniform distribution in the interval $(0, 1)$) is used to create a vector of initial conditions between $(0, 1)$ which is multiplied by the maximum value (here 2).

Parameters	Value(s)
Number of tests	2000
N (number of oscillators)	$\{100, 500, 1000, 2000, 5000\}$
Tolerances	$\{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$
T_f (Final time)	$\{10\pi,\ 20\pi,\ 50\pi\}$
Initial conditions	$[0, 1]$ (uniform distribution)

Table 4: *Explored parameters for Coupled linear oscillators.*

A.2 Kuramoto (Benchmark 2)

Initial conditions All the initial conditions are generated in the same way. The seed is fixed by the number identifying the test. Then the function *rand* of Matlab (returns a random scalar drawn from the uniform distribution in the interval $(0, 1)$) is used to create a vector of initial conditions between $(0, 1)$ which is multiplied by the maximum value (here 2π).

Natural frequencies The natural frequencies are generated with a seed fixed with the number of the test. The function *randn* (returns a random scalar drawn from the standard normal distribution) is used to create the vector which is multiplied by the parameter σ. In other words, we have $\omega \sim \mathcal{N}(0,\, \sigma^2)$.

Parameters	Value(s)
Number of tests	12000
N (Number of oscillators)	$\{100, 500, 1000, 2000, 5000, 10000\}$
Tolerances	$\{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$
T_f (Final time)	$\frac{4\pi}{\mathrm{med}(\lvert W\rvert)K+0.001}$
Initial conditions	$[0, 2\pi]$ (uniform distribution)
σ	$[0, 1]$
ω_i	$[-\sigma, \sigma]$
K(coupling coefficient)	$[0, 3\sigma]$

Table 5: *Explored parameters for Kuramoto.*

A.3 Circadian clock (Benchmark 3)

Initial conditions The seed is fixed by the number of the test. Then the four components are created in this way:

$$(\mathbf{X}_{i,1},\ \mathbf{X}_{i,2},\ \mathbf{X}_{i,3},\ \mathbf{X}_{i,4}) = (1, 1, -1.19, -0.62) + 0.2 \times (p_1, p_2, p_3, p_4).$$

Where p_k follows a uniform distribution between $[-0.5, 0.5]$.

Parameters	Value(s)
Number of tests	5000
N (Number of cells)	$\{50, 200, 500, 700, 1000\}$
Tolerances	$\{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$
T_f (Final time)	$[2, 3] \times T_{cycle} = [48, 72]$
K(coupling coefficient)	$\{0.001, 0.1, 1, 10\}$
I_0	$\{0.228249, 1.5, 10\}$

Table 6: *Explored parameters for Benchmark 3*

B Tests selections

First we present the success rate of the 4 solvers in Table 7. The rate is lower for Benchmark 2 but we performed more tests with higher sizes and more strict tolerances, see Table 5. We also note that the success rate is very close among the solvers. That is why we restrict the study of the results only on the tests that have been finished, *i.e.* the final time has been reached before any failure criterion, see **Failure conditions** in Section 2.2.

In Figure 4 the ratio between the number of tests finished by all the solvers compared to the number of launched tests with the same couple of system size (abscissa) and relative tolerance (ordinate) and this for each benchmark.

Solver	Benchmark 1	Benchmark 2	Benchmark 3
NoT	2,000	12,000	5,000
Single	99.0%	87.2%	99.3%
Mixed1	98.5%	87.2%	99.3%
Mixed2	98.6%	87.2%	99.3%
Double	98.8%	89.8%	99.2%
Ref	98.8%	89.8%	99.2%

Table 7: *Success rate for each solver by benchmark.*

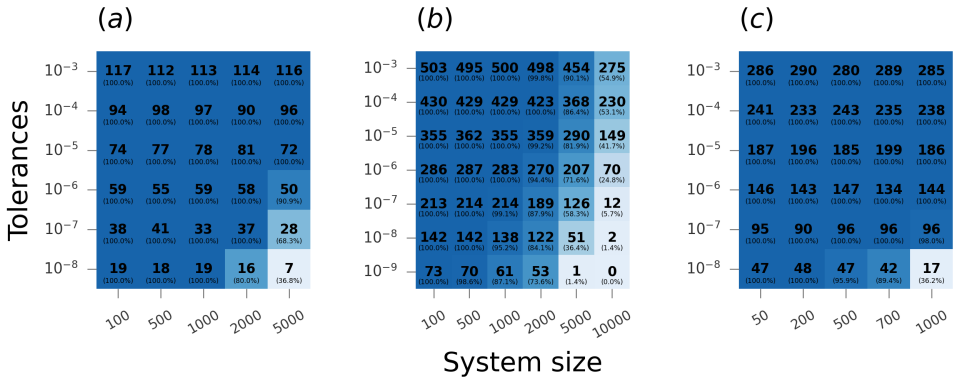

Figure 4: *Number of tests successfully solved by all solvers for each couple of system size (abscissa) and relative tolerance (ordinate). (a) Benchmark 1, (b) Benchmark 2 and (c) Benchmark 3.*