Leveraging Mixed Precision in Exponential Time Integration Methods

Cody J. Balos*, Steven Roberts†, and David J. Gardner‡
Center for Applied Scientific Computing,
Lawrence Livermore National Laboratory,
7000 East Ave, Livermore, CA
Email: *balos1@llnl.gov, †roberts115@llnl.gov, ‡gardner48@llnl.gov

***Abstract*—The machine learning explosion has created a prominent trend in modern computer hardware towards low precision floating-point operations. In response, there have been growing efforts to use low and mixed precision in general scientific computing. One important area that has received limited exploration is time integration methods, which are used for solving differential equations that are ubiquitous in science and engineering applications. In this work, we develop two new approaches for leveraging mixed precision in exponential time integration methods. The first approach is based on a reformulation of the exponential Rosenbrock–Euler method allowing for low precision computations in matrix exponentials independent of the particular algorithm for matrix exponentiation. The second approach is based on an inexact and incomplete Arnoldi procedure in Krylov approximation methods for computing matrix exponentials and is agnostic to the chosen integration method. We show that both approaches improve accuracy compared to using purely low precision and offer better efficiency than using only double precision when solving an advection-diffusion-reaction partial differential equation.**

***Index Terms*—differential equations, mixed precision, high-performance computing**

I. Introduction

In this paper we present two complementary concepts that enable accurate mixed precision computation in exponential time integators. Exponential time integrators are a class of numerical methods for solving ordinary differential equation (ODE) initial value problems of the form

$$u'(t) = f(u(t)), \quad u(t_0) = u_0, \quad t \in [t_0, t_f], \tag{1}$$

with $u(t) \in \mathbb{R}^N$. ODEs are ubiquitous across scientific domains and may arise directly from modeling some process or from discretizing a partial differential equation (PDE). Exponential integrators are particularly well-suited to stiff problems due to their exact treatment of linear terms. Alternative methods for stiff ODEs e.g., BDF or implicit Runge–Kutta methods, typically require an effective and efficient preconditioner which can be difficult to construct [30]. Exponential time integrators have been shown to be effective for many problems where practical preconditioners have not been developed [15], [22], [24], [27], [28], [40].

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 23-FS-013. LLNL-PROC-851497.

Recent trends in computer hardware towards low precision floating point operations have been spurred largely by artificial intelligence and machine learning applications. Reuther et al. provides a comprehensive survey of current AI accelerators and their properties [36]. The typical properties of this hardware indicate that leveraging low precision is necessary to achieve the full potential of much of this hardware. This has resulted in significant interest in mixed precision computation. The goal of incorporating mixed precision is to utilize the efficiency of low precision computation while maintaining an overall accuracy consistent with high precision computation. What constitutes low and high precision depends on the context, but it is common to consider double precision as high precision and low precision as anything less than double. Mixed precision has been particularly popular in the numerical linear algebra [1], [2], [5], [6], [19], [20], [26], [32] and deep learning literature [10], [17], [31], [33], [41]. However, incorporating mixed precision into numerical time integration methods has been studied much less [8], [11], [18]. To better utilize current and emerging hardware capabilities, further research on incorporating mixed precision into numerical time integration methods is needed. To this end, we present two approaches for leveraging mixed precision computations in exponential integrators:

1) a reformulation of the exponential Rosenbrock–Euler method with order of accuracy $\mathcal{O}\big(h^2 + \epsilon h\big)$ instead of $\mathcal{O}\big(h^2 + \epsilon\big)$ where ϵ is the floating point precision,
2) and the incorporation of low precision matrix-vector products in the Krylov approximation of matrix-exponential and vector products.

These two approaches have different requirements and characteristics which may dictate which is most suitable for a particular application. They can also be combined to create a practical and robust mixed precision exponential time-integrator.

The rest of this paper is organized as follows. In section II we present the reformulated exponential Rosenbrock–Euler method. This is followed by section III where we present the mixed precision Krylov approximation algorithm. In section IV we demonstrate both approaches in solving an advection-diffusion-reaction PDE. Finally, in section V we provide key conclusions, impacts, and directions for future work.

II. Reformulation of Exponential Euler

The exponential Rosenbrock–Euler method [35] applied to (1) is given by

$$u_{n+1} = u_n + h_n \varphi_1(h_n J_n) f(u_n) \tag{2}$$

where h_n is the timestep, $J_n = f'(u_n)$ is the Jacobian matrix, and u_n is the numerical approximation to $u(t_n)$. The function $\varphi_1(z) = (\exp(z) - 1)/z$ is just one member of the sequence of functions

$$\varphi_0(z) = \exp(z), \qquad \varphi_{k+1}(z) = \frac{\varphi_k(z) - \varphi_k(0)}{z}, \tag{3}$$

which are ubiquitous in the exponential integrator literature [23]. Although we have used the conventional scalar form, φ-functions can be lifted to square matrix-valued inputs by analyticity.

It is well-known that the exponential Rosenbrock–Euler method is second order accurate both in the classical sense [35] and for stiff, semilinear problems [24]. These results are based on the assumption that φ_1 is computed exactly; however, this is rarely the case in practice. Typically, it is computed to a specified tolerance and contains errors from floating point arithmetic. As demonstrated in experiments later in this section, performing the linear algebra associated with φ-functions on low precision hardware can severely limit the accuracy of an exponential integrator.

Using (3), we can equivalently express the exponential Rosenbrock–Euler scheme (2) as

$$\begin{aligned} u_{n+1} &= u_n + h_n f(u_n) + h_n(\varphi_1(h_n J_n) - I) f(u_n) \\ &= u_n + h_n f(u_n) + h_n^2 \varphi_2(h_n J_n) J_n f(u_n). \end{aligned} \tag{4}$$

In (4), the φ-function is scaled by h_n^2 as opposed to h_n in (2). Consequently, in a computer implementation, we may expect improved resilience to φ-function errors as $h_n \to 0$. However, this asymptotic analysis breaks down when J_n is disproportionally large and h_n is not sufficiently small. In this stiff regime, the term $h_n^2 \varphi_2(h_n J_n) J_n f(u_n)$ is susceptible to overflows as well as cancellation errors with $h_n f(u_n)$.

Therefore, we propose the following reformulated exponential Rosenbrock–Euler scheme which uses a parameter, γ_n, to vary between the forms of (2) and (4),

$$\widetilde{u}_{n+1} = \widetilde{u}_n + h_n \gamma_n f(\widetilde{u}_n) + \mathrm{fl}(h_n \psi(h_n \widetilde{J}_n, \gamma_n) f(\widetilde{u}_n)). \tag{5}$$

The function $\mathrm{fl}(x)$ represents the evaluation of x to a tolerance ϵ and is assumed to satisfy the error model $\mathrm{fl}(x) = (I + \delta)x$ with $\|\delta\|_2 \le \epsilon$. We use $\widetilde{u}_n$ and $\widetilde{J}_n = f'(\widetilde{u}_n)$ to denote numerical solutions computed with this error and to distinguish from u_n in (2) and (4) which assumes exact φ-functions. Finally, we introduce

$$\psi(z, \gamma) = \varphi_1(z) - \gamma \tag{6a}$$

$$= (1 - \gamma)\varphi_1(z) + \gamma \varphi_2(z) z. \tag{6b}$$

While the form (6a) is useful for analysis, (6b) is preferable for implementation as it is less susceptible to subtractive cancellation.

A. Error Analysis

In order to inform the selection of the yet unspecified parameter γ_n in (5), we first study the effect of γ_n on the numerical error. The local truncation error committed after one step is

$$e_1 = \widetilde{u}_1 - u(t_1).$$

This satisfies

$$\begin{aligned} \|e_1\|_2 &= \|\widetilde{u}_1 - u(t_1)\|_2 \\ &\le \|u_1 - u(t_1)\|_2 + \|\widetilde{u}_1 - u_1\|_2 \\ &\le C h_0^3 + \left\| \delta h_0 \psi(h_0 \widetilde{J}_0, \gamma_0) f(\widetilde{u}_0) \right\|_2 \\ &\le C h_0^3 + h_0 \epsilon \left\| \varphi_1(h_0 \widetilde{J}_0) f(\widetilde{u}_0) - \gamma_0 f(\widetilde{u}_0) \right\|_2, \end{aligned} \tag{7}$$

where we have used the triangle inequality and the second order convergence property of the exponential Rosenbrock–Euler method.

This suggests solving the optimization problem

$$\begin{aligned} \gamma_n &= \operatorname*{argmin}_{\gamma} \left\| \varphi_1(h_n \widetilde{J}_n) f(\widetilde{u}_n) - \gamma f(\widetilde{u}_n) \right\|_2^2 \\ &= \frac{f(\widetilde{u}_n)^T \varphi_1(h_n \widetilde{J}_n) f(\widetilde{u}_n)}{\|f(\widetilde{u}_n)\|_2^2} \end{aligned} \tag{8}$$

to select γ_n at each step to minimize the effect of the low precision arithmetic. As φ_1 is already required to compute ψ in (5), the additional cost of computing γ_n is negligible for many algorithms used to compute linear combinations of φ-functions. Alternatively, one can use the bound

$$\gamma_n \le \mu_2(\varphi_1(h_n \widetilde{J}_n)) \le \varphi_1(h_n \mu_2(\widetilde{J}_n)) \tag{9}$$

to choose γ_n. If $\mu_2(\widetilde{J}_n)$, the logarithmic 2-norm [39] of the Jacobian, can be readily estimated, (9) only requires inexpensive scalar arithmetic.

In the stiff regime where $\widetilde{J}_n \to -\infty$, the reformulated method (5) approaches the form of (2) because $\gamma_n \to 0$. Conversely, in the asymptotic regime where $h_n \to 0$, (5) approaches the form of (4). A Taylor expansion of (8) reveals $\gamma_n = 1 - \mathcal{O}(h_n)$. Thus, the local truncation error is $\|e_1\|_2 = \mathcal{O}\big(h_0^3 + \epsilon h_0^2\big)$ as opposed to $\mathcal{O}\big(h_0^3 + \epsilon h_0\big)$ for a standard implementation where $\gamma_n = 0$.

B. Convergence Experiment

In order to verify the improved accuracy of (5), we compare its convergence to a standard implementation of the exponential Rosenbrock–Euler method on an advection–diffusion–reaction PDE from [9, Section 5.1],

$$\begin{aligned} \frac{\partial u}{\partial t} &= \varepsilon \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \alpha \left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \right) \\ &\quad + \rho u \left(u - \frac{1}{2} \right) (1 - u), \\ u(0, x, y) &= 0.3 + 256(x(1-x)y(1-y))^2. \end{aligned} \tag{10}$$

The timespan is $[0, 0.3]$ and the spatial domain, $x, y \in [0, 1]$, is discretized by second order finite differences with $\Delta x = \Delta y = 0.05$. The remaining parameters are $\varepsilon = 0.05$, $\alpha = -1$,

and $\rho = 1$. Error in the numerical solution is measured as $\|u_n - u_{\text{ref}}\|_2$, where u_{ref} is a reference solution computed with an absolute and relative tolerance of 10^{-13}.

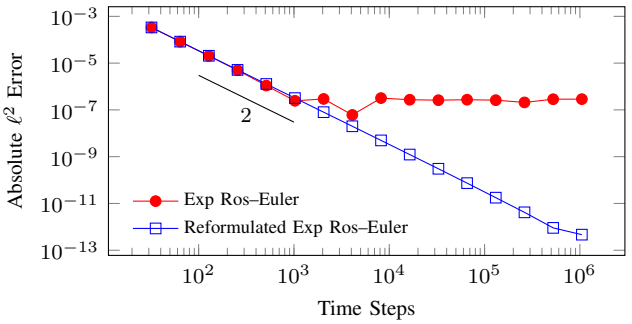

Fig. 1. The reformulated exponential Rosenbrock–Euler method (5) maintains second order convergence despite using single precision for φ-functions, while the standard form stagnates at the accuracy of the φ-function.

Our first experiment uses single precision for the terms in the fl function of (5) including the Jacobian evaluation and φ-functions computed with the KIOPS [16] algorithm. The remaining operations, including evaluating f, are performed in double precision. Figure 1 shows that the accuracy of the standard exponential Rosenbrock–Euler implementation is limited by the accuracy of the φ-functions as it cannot achieve an error below 10^{-7}. The reformulated version (5), however, is able to achieve errors six orders of magnitude smaller without suffering order reduction.

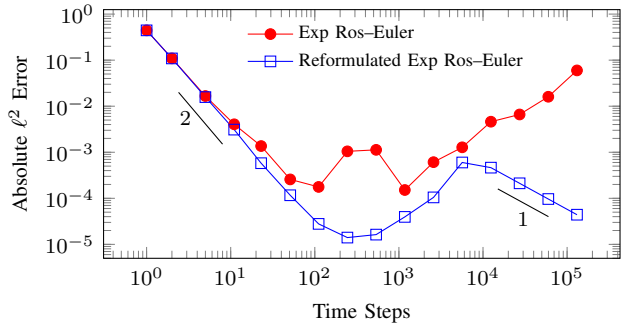

Fig. 2. With half precision φ-functions, the reformulated exponential Rosenbrock–Euler method (5) achieves a minimum error approximately ten times smaller than that of the standard form.

When using half precision instead of single precision for the Jacobian evaluation and φ-functions (Figure 2), the improvement in accuracy with the reformulated method (5) is more modest: an order of magnitude. After a momentary degradation in accuracy between 200 and 6000 time steps, the ϵh_0^2 term of the local error becomes dominant and we see further asymptotic improvements. In this test, we compute φ-functions using the algorithm from [3] as it was more robust to half precision than KIOPS.

III. Computing φ-function products with a mixed precision Krylov method

Historically, exponential methods were bound by the cost and difficulty of computing matrix exponentials in the φ-functions. However, over the last several decades a rich literature has developed around the use of Krylov approximations for the action of φ-functions on a vector [4], [16], [34], [37]. These Krylov-based approaches have made exponential integrators more practical to use and are based on the approximation,

$$\begin{aligned} \exp(\tau A)v &\approx \beta V_m \exp(\tau H_m) e_1, \\ m \ll n, \quad \beta = \|v\|_2, \quad e_1 &= (1, 0, \ldots, 0)^T, \end{aligned} \tag{11}$$

where $V_m \in \mathbb{R}^{n\times m}$ is the orthonormal basis of the Krylov subspace $\mathcal{K}_m(A, v)$ and $H_m \in \mathbb{R}^{m\times m}$ is the Hessenberg matrix generated by the Arnoldi process. In early approaches for computing (11), such as the methods in Expokit [37], the computational cost is dominated by the full orthogonlization method (FOM) [42]. State-of-the-art implementations, such as KIOPS [16], utilize an incomplete orthogonalization process (IOP). Using IOP is not only faster, but it shifts the majority of the computing effort to matrix-vector products [42] which map well to low precision computing units on modern hardware

A. Introducing low precision into IOP Arnoldi

It is therefore natural to consider introducing low precision computations into IOP Arnoldi via the matrix-vector products since they map well to many low precision hardware units. A naive approach is simply to perform all of the matrix-vector products in low precision. We demonstrate the problems with this approach by modifying the IOP Arnoldi procedure (Algorithm 1) in KIOPS so that the matrix-vector products are computed in low precision and the result is stored in double-precision (we use the `chop` method from [21] to simulate this) in two different experiments with IEEE single, NVIDIA TensorFloat-32 (TF32), IEEE half, and bfloat16 floating-point formats. The test setups we use are essentially the same as the ones utilized by Al-Mohy and Higham [4, Experiment 5 and 7] which are based on experiments conducted by Niesen and Wright [34, Experiment 1] and Sidje [37, Section 6.2].

Experiment 1: Compute $u = \exp(tA)b_0$ using KIOPS with Algorithm 1 for three matrices. The first two matrices are from the Harwell-Boeing collection [14] and are available in the SuiteSparse Sparse Matrix Collection [12]. With the `orani678` sparse matrix (order $n = 2529$ with $nnz = 90158$ nonzero elements) we use $t = 10$, $b_0 = [1, \ldots, 1]^T$, and set the KIOPS tolerance to $tol = \sqrt{\epsilon_d}$ where ϵ_d is machine precision for the double precision format. For the `bcspwr10` sparse matrix (order $n = 5300$ with $nnz = 21842$) we use $t = 10$, $b_0 = [1, 0, \ldots, 0, 1]^T$, and $tol = 10^{-5}$. The third test uses a Poisson matrix of order $n = 9801$, $t = 1$, and $tol = 10^{-12}$. The matrix and the b vector are generated with the MATLAB code

```
A = 2500 * gallery('poisson', 99);
g = (-0.98 : 0.02 : 0.98)';
```

```
[R1, R2] = meshgrid(g, g);
r1 = R1(:); r2 = R2(:);
b = (1 - r1.^2) .* (1 - r2.^2) .* exp(r1);
```

Experiment 2: Compute $u = \varphi_0(tA)b_0 + t\varphi_1(tA)b_1 + \cdots + t^4\varphi_4(tA)b_4$ via the modified KIOPS method with the `orani678`, `bcspwr10`, and Poisson matrices and $b_i = [1, \ldots, 1]^T$ using the same values for t and tol as in Experiment 1.

Algorithm 1 Naive low precision IOP Arnoldi with the low precision computation (line 4) boxed.

1: **Input:** $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times p}$, $V \in \mathbb{R}^{(N+p)\times(m_{\max}+1)}$, j, m
2: **while** $j < m$ **do**
3: $j = j + 1$
4: $V(1:N, j+1) =$ $\boxed{A \cdot V(1:N, j) + B \cdot V(N+1:N+p, j)}$
5: $V(N+1:N+p-1, j+1) = V(N+2:N+p, j)$
6: $V(N+p, j+1) = 0$
7: **for** $i = \max(1, j-1)$ **to** j **do**
8: $H(i, j) = V(:, i)^T \cdot V(:, j+1)$
9: $V(:, j+1) = V(:, j+1) - H(i, j) \cdot V(:, i)$
10: **end for**
11: $s = \|V(:, j+1)\|_2$
12: **if** $s \approx 0$ **then**
13: happy_breakdown = true
14: **break**
15: **end if**
16: $H(i+1, j) = s$
17: $V(:, j+1) = \frac{1}{s}V(:, j+1)$
18: **end while**
19: **return** V, H, j

Letting u_p be the solution generated using KIOPS with Algorithm 1 and precision p, we define the error as $err(u_p) = \|u_p - u_{\text{ref}}\|_\infty / \|u_{\text{ref}}\|_\infty$. The reference solution u_{ref} is generated with the standard KIOPS method in double precision with a tolerance of $tol = \epsilon_d$. Unless otherwise stated, results use the default KIOPS parameters. When using the naively modified IOP Arnoldi in KIOPS, we see that the error is far greater than the desired tolerance (Table I).

In an attempt to recover the lost accuracy from low precision matrix-vector products, we now reconsider replacing the exact (in finite arithmetic) matrix-vector products with the inexact matrix-vector product

$$\tilde{A}v = (A + E)v, \tag{12}$$

where E is some perturbation matrix. Substituting (12) into (11) and allowing the E to change with the Arndoli iterate yields the inexact Arnoldi approximation

$$\begin{aligned}(A + \mathcal{E}_m)V_m &= V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \\ \mathcal{E}_m &= \sum_{j=1}^{m} E_j v_j v_j^T.\end{aligned} \tag{13}$$

The theoretical underpinnings for this approach are developed in [38]. Furthermore, [7] provides bounds on the growth of $\|E_j\|_2$ as the iterations progress in various Krylov subspace methods including FOM Arnoldi. Dinh and Sidje extended the work to computing the matrix-exponential in [13]. However, the combination of IOP, inexact products, and matrix-exponential computations has, as far as we are aware, not been previously examined in the literature.

We numerically investigate the effectiveness of this intuitive approach by progressively introducing lower-precision matrix-vector products (i.e., allowing $\|E_j\|_2$ to grow) into the IOP Arnoldi algorithm within KIOPS as the Arnoldi iteration proceeds. We define two new parameters m_{chop_1} and m_{chop_2} that determine the Arnoldi iterates at which we switch from full double-precision matrix-vector products to single-precision and then from single to either TF32, half, or bfloat16 (Algorithm 2).

We repeat Experiments 1 and 2 while first varying m_{chop_1} until the target error is below $\max(err(u_d), tol)$. This metric is employed because $err(u_d) > tol$ in Experiment 2 with the `bcspwr10` matrix, so there is no hope of doing better than $err(u_d)$ in this case. Then, with m_{chop_1} fixed to the value we just found, we vary m_{chop_2} until the tolerance is met. Utilizing this procedure with mixed precision IOP Arnoldi enables KIOPS to achieve a much lower error while leveraging a precision lower than double for 40% or more of the Arnoldi iterates (Table II). Furthermore, we are able to leverage lower than single-precision for 25% – 60% of iterates.

IV. Integrated Numerical Experiments

To evaluate the performance of the two approaches for leveraging low precision computation in exponential integrators we test three methods: standard exponential Rosenbrock–Euler, the reformulated exponential Rosenbrock–Euler scheme (5), and the stiffly-accurate fourth-order exprk4s6 [29]. All three methods are tested with standard KIOPS and KIOPS with the mixed precision IOP Arnoldi (Algorithm 2) for evaluating φ-function vector products. As before we use `chop` for simulating low-precision computations. With mixed precision IOP Arnoldi we use m_{chop_1} and m_{chop_2} to set the iteration for switching to single or half precision matrix-vector products, respectively. The process used to choose these values is similar to the process used in the experiments in Section III. We find a value for m_{chop_1} that produces the desired error, fix its value, then we find m_{chop_2} that similarly allows the desired error to be met. For the exprk4s6 method, this means we have to choose the values for each of the four calls per time step that it makes to KIOPS. The six possible combinations of schemes are used to solve the advection-diffusion-reaction problem (10) with the same parameters but a finer spatial discretization, $\Delta x = \Delta y = 0.0025$, leading to a stiffer problem.

A. Overall accuracy

Figure 3 shows the error $\|u_n - u_{\text{ref}}\|_\infty / \|u_{\text{ref}}\|_\infty$ versus the number of time steps to demonstrate the convergence of the different schemes. The reference solution is generated with

			double	single	TF32	half	bfloat16
	matrix	tol	error	error	error	error	error
Experiment 1	orani678	1.49e−8	6.88e−12	2.41e−5	1.32e−1	1.71e+0	3.91e+1
	bcspwr10	1.00e−5	4.84e−10	3.10e−2	4.03e+2	7.67e+3	3.39e+3
	Poisson	1.00e−12	8.38e−14	1.15e−9	1.76e−5	1.14e−5	7.70e−3
Experiment 2	orani678	1.49e−8	1.69e−13	2.91e−5	7.63e−1	1.09e+0	2.87e+0
	bcspwr10	1.00e−5	1.73e−5	2.37e−6	2.06e−2	2.06e−2	1.27e−1
	Poisson	1.00e−12	2.06e−14	1.26e−9	4.60e−3	6.40e−3	1.85e−1

TABLE I

RESULTS FROM EXPERIMENTS WITH THE NAIVE LOW PRECISION IOP ARNOLDI (ALGORITHM 1) IN KIOPS SHOW THAT THE METHOD IS UNRELIABLE. IN ALL CASES, THE RELATIVE ERROR IS LARGE WHEN USING LOW PRECISION WITH RESPECT TO BOTH THE TOLERANCE AND THE RELATIVE ERROR ACHIEVED WITH DOUBLE PRECISION.

			single, TF32			single, half			single, bfloat16		
	matrix	target error	$m_{\text{chop}_{1,2}}$	m	error	$m_{\text{chop}_{1,2}}$	m	error	$m_{\text{chop}_{1,2}}$	m	error
Experiment 1	orani678	1.49e−8	30, 39	51	4.01e−9	30, 39	51	4.01e−9	30, 40	51	5.41e−9
	bcspwr10	1.00e−5	36, 54	86	6.82e−6	36, 54	86	6.82e−6	36, 54	87	6.64e−6
	Poisson	1.00e−12	15, 70	128	6.49e−13	15, 70	128	6.68e−13	15, 90	128	9.64e−13
Experiment 2	orani678	1.49e−8	28, 37	51	2.85e−9	28, 37	51	2.85e−9	28, 39	51	2.61e−9
	bcspwr10	1.73e−5	51, 63	106	1.73e−5	51, 63	106	1.73e−5	51, 66	106	1.73e−5
	Poisson	1.00e−12	28, 58	128	8.89e−13	28, 58	128	9.26e−13	28, 60	128	3.40e−13

TABLE II

PROGRESSIVELY INTRODUCING LOWER-PRECISION MATRIX-VECTOR PRODUCTS INTO THE IOP ARNOLDI PROCEDURE (ALGORITHM 2) WITHIN KIOPS ENABLES THE TARGET ERROR, $\max(err(u_d), tol)$, TO BE MET. m_{CHOP_1} AND m_{CHOP_2} ARE THE KRYLOV ITERATIONS FOR SWITCHING FROM SINGLE-PRECISION TO TF32, HALF, OR BFLOAT16. m IS THE NUMBER OF VECTORS IN $\mathcal{K}(A, v)$ FOR THE LAST ITERATION OF KIOPS AND IS GENERALLY A GOOD ESTIMATE FOR THE BASIS SIZE REQUIRED (128 IS THE DEFAULT MAXIMUM).

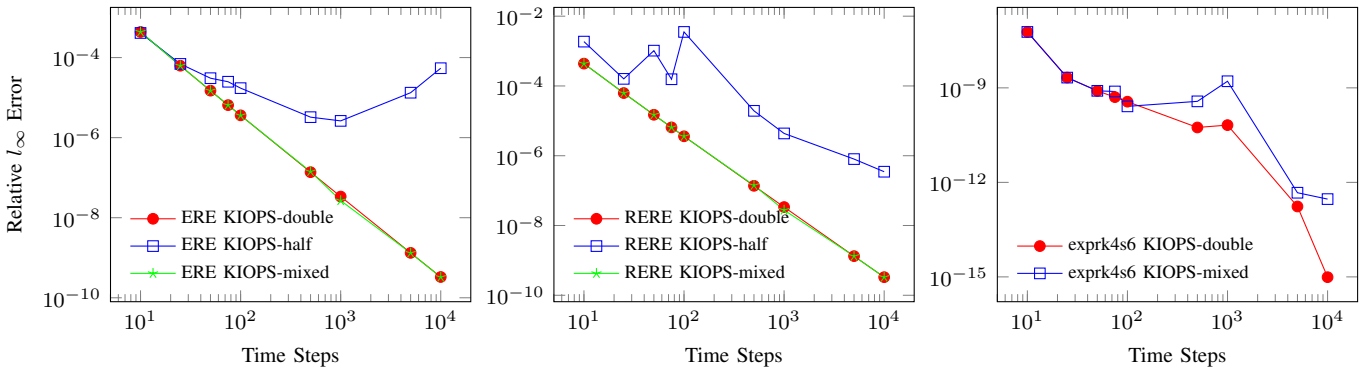

Fig. 3. The reformulated exponential Rosenbrock–Euler (RERE) method (5) achieves a much lower error than the standard exponential Rosenbrock–Euler (ERE) when using KIOPS with only half precision matrix-vector products via Algorithm 1. Using mixed precision matrix-vector products in KIOPS via Algorithm 2 significantly improves the error for all methods. exprk4s6 is unable to converge at all with Algorithm 1 and only half precision matrix-vector products while using Algorithm 2 enables it to run and obtain a reasonably accurate solution.

the exprk4s6 method with 10^5 time steps. Once again we see that the reformulated exponential Rosenbrock–Euler method (5) consistently achieves lower error and maintains second order convergence longer than the standard Rosenbrock–Euler method. The use of KIOPS with mixed precision IOP Arnoldi greatly improves the accuracy for both methods, with the error nearly identical to what is achieved when using KIOPS with double precision. In the case of the higher-order exprk4s6, mixed precision IOP Arnoldi enables using low precision as running with only half precision does not converge.

B. Idealized computational efficiency

Figure 4 provides an estimate of the computational efficiency and shows the error versus the number of "effective" matrix-vector products, $mv_{\text{effective}}$, where

$$mv_{\text{effective}} = mv_{\text{double}} + \frac{mv_{\text{single}}}{a} + \frac{mv_{\text{half}}}{b}. \tag{14}$$

We use matrix-vector products as a proxy for the wall-clock time since they are typically the critical path through the integration [27]. Since the sparse matrix-vector multiply is typically a memory bound computation, we set a to be the ratio of double and single memory bandwidth and b to be the ratio of double and half memory bandwidth. For typical hardware, like the NVIDIA A100, this simply yields $a = 2$ and $b = 4$. This estimate may be conservative if using lower-precision moves the sparse matrix-vector multiply into a compute-bound regime (possible on some hardware, like the Cerebras Wafer Scale Engine [25]). The notable result is that the mixed precision

Algorithm 2 Mixed precision IOP Arnoldi with the first low precision computations (line 7) in the dashed box and lowest precision the in solid box (line 5).

1: **Input:** $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times p}$, $V \in \mathbb{R}^{(N+p)\times(m_{\max}+1)}$, j, m
2: **while** $j < m$ **do**
3: $j = j + 1$
4: **if** $j + 1 > m_{\text{chop}_2}$ **then**
5: $V(1:N, j+1) = \boxed{A \cdot V(1:N, j) + B \cdot V(N+1:N+p, j)}$
6: **else if** $j + 1 > m_{\text{chop}_1}$ **then**
7: $V(1:N, j+1) = \boxed{A \cdot V(1:N, j) + B \cdot V(N+1:N+p, j)}$
8: **else**
9: $V(1:N, j+1) = A \cdot V(1:N, j) + B \cdot V(N+1:N+p, j)$
10: **end if**
11: $V(N+1:N+p-1, j+1) = V(N+2:N+p, j)$
12: $V(N+p, j+1) = 0$
13: **for** $i = \max(1, j-1)$ **to** j **do**
14: $H(i,j) = V(:,i)^T \cdot V(:, j+1)$
15: $V(:, j+1) = V(:, j+1) - H(i,j) \cdot V(:,i)$
16: **end for**
17: $s = \|V(:, j+1)\|_2$
18: **if** $s \approx 0$ **then**
19: happy_breakdown = true
20: **break**
21: **end if**
22: $H(i+1, j) = s$
23: $V(:, j+1) = \frac{1}{s} V(:, j+1)$
24: **end while**
25: **return** V, H, j

IOP Arnoldi makes all of the schemes more efficient in most regimes. The few exceptions are in the case of exprk4s6 when the error is around 10^{-10}. In this case, the extra Krylov iterations induced by the lower precision introduce too much overhead for the use of low precision to provide a benefit.

V. Conclusions

Modern computer hardware offers significantly increased low precision floating point performance in comparison to double precision. We have developed two approaches to leveraging low precision in exponential time integration methods.

With a minor modification to the exponential Rosenbrock–Euler method, our reformulated version (5) of the method attains improved resilience to inexact φ-functions. This enables utilizing cheaper, low precision arithmetic or looser tolerances in the most expensive part of the integrator. The reformulated exponential Rosenbrock–Euler method (5) is particularly effective at maintaining convergence when combining single precision φ-functions with double precision for the remaining computations. Half precision φ-functions present many challenges with avoiding underflow and overflow, particularly for stiff problems. Nevertheless, improved accuracy is still achievable with our reformulated scheme (5). While we focused on the exponential Rosenbrock–Euler method, the reformulation idea could be generalized to other exponential methods and will be the subject of future investigations.

Our mixed precision IOP Arnoldi algorithm incorporated into KIOPS, or similar Krylov approximation methods, can readily be utilized within higher order methods as demonstrated in experiments with the advection-diffusion-reaction PDE. This algorithm enables exponential methods to compute the φ-function products while leveraging low precision for the matrix-vector products and still recovering the required approximation accuracy. The process of manually choosing $m_{\text{chop}_{1,2}}$ for fixed matrices as in Section III is much more difficult in the context of ODE or spatially discretized PDE systems like the the advection-diffusion-reaction problem. This is primarily due to the dynamical nature of the problem changing the optimal values. As such, to make the mixed precision IOP Arnoldi more practical, an adaptive approach to selecting the precision for the matrix-vector products is needed. This is another topic we will explore in the future.

VI. Acknowledgements

We would like to thank Valentin Dallerit for enlightening discussions and insight into the KIOPS algorithm and software implementation. We are also grateful for the thoughtful feedback and insight of Professor Daniel Reynolds at Southern Methodist University.

References

[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, et al., *A survey of numerical linear algebra methods utilizing mixed-precision arithmetic*, The International Journal of High Performance Computing Applications, 35 (2021), pp. 344–369.

[2] S. Abdulah, Q. Cao, Y. Pei, G. Bosilca, J. Dongarra, M. G. Genton, D. E. Keyes, H. Ltaief, and Y. Sun, *Accelerating geostatistical modeling and prediction with mixed-precision computations: A high-productivity approach with PaRSEC*, IEEE Transactions on Parallel and Distributed Systems, 33 (2021), pp. 964–976.

[3] A. H. Al-Mohy and N. J. Higham, *Computing the action of the matrix exponential, with an application to exponential integrators*, SIAM journal on scientific computing, 33 (2011), pp. 488–511.

[4] A. H. Al-Mohy and N. J. Higham, *Computing the action of the matrix exponential, with an application to exponential integrators*, SIAM Journal on Scientific Computing, 33 (2011), pp. 488–511.

[5] H. Anzt, E. G. Boman, M. Gates, S. Kruger, S. Li, J. Loe, D. Osei-Kuffuor, S. Tomov, Y. M. Tsai, and U. M. Yang, *Towards use of mixed precision in ECP math libraries*, tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2021.

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

[7] A. Bouras and V. Frayss, *Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 660–678.

[8] B. Burnett, S. Gottlieb, Z. J. Grant, and A. Heryudono, *Performance evaluation of mixed-precision Runge-Kutta methods*, in 2021 IEEE High Performance Extreme Computing Conference (HPEC), IEEE, 2021, pp. 1–6.

[9] M. Caliari and A. Ostermann, *Implementation of exponential Rosenbrock-type integrators*, Applied Numerical Mathematics, 59 (2009), pp. 568–581.

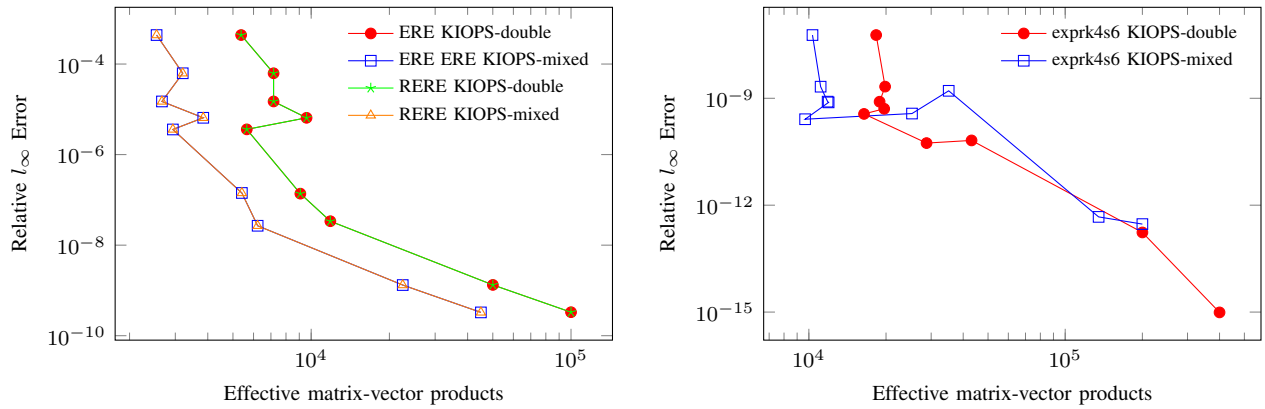

Fig. 4. In the case of the reformulated exponential Rosenbrock–Euler (RERE) method (5) and standard exponential Rosenbrock–Euler (ERE) method, using the mixed precision IOP Arnoldi (Algorithm 2) in KIOPS reduced the number of effective matrix-vector products compared to the double precision version. With exprk4s6, the mixed precision KIOPS was more efficient until the error reached 10^{-10}.

[10] W. Chen, P. Wang, and J. Cheng, *Towards mixed-precision quantization of neural networks via constrained optimization*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 5350–5359.

[11] M. Croci and G. Rosilho de Souza, *Mixed-precision explicit stabilized Runge–Kutta methods for single- and multi-scale differential equations*, Journal of Computational Physics, 464 (2022), p. 111349.

[12] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software (TOMS), 38 (2011), pp. 1–25.

[13] K. N. Dinh and R. B. Sidje, *Analysis of inexact Krylov subspace methods for approximating the matrix exponential*, Mathematics and Computers in Simulation, 138 (2017), pp. 1–13.

[14] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Transactions on Mathematical Software (TOMS), 15 (1989), pp. 1–14.

[15] L. Einkemmer, M. Tokman, and J. Loffeld, *On the performance of exponential integrators for problems in magnetohydrodynamics*, Journal of Computational Physics, 330 (2017), pp. 550–565.

[16] S. Gaudreault, G. Rainwater, and M. Tokman, *KIOPS: A fast adaptive Krylov subspace solver for exponential integrators*, Journal of Computational Physics, 372 (2018), pp. 236–255.

[17] C. Gong, Z. Jiang, D. Wang, Y. Lin, Q. Liu, and D. Z. Pan, *Mixed precision neural architecture search for energy efficient deep learning*, in 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), IEEE, 2019, pp. 1–7.

[18] Z. J. Grant, *Perturbed Runge–Kutta methods for mixed precision applications*, Journal of Scientific Computing, 92 (2022), pp. 1–20.

[19] A. Haidar, H. Bayraktar, S. Tomov, J. Dongarra, and N. J. Higham, *Mixed-precision iterative refinement using tensor cores on gpus to accelerate solution of linear systems*, Proceedings of the Royal Society A, 476 (2020), p. 20200110.

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

[21] N. J. Higham and S. Pranesh, *Simulating low precision floating-point arithmetic*, SIAM Journal on Scientific Computing, 41 (2019), pp. C585–C602.

[22] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM Journal on Scientific Computing, 19 (1998), pp. 1552–1574.

[23] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), p. 209–286.

[24] M. Hochbruck, A. Ostermann, and J. Schweitzer, *Exponential Rosenbrock-type methods*, SIAM Journal on Numerical Analysis, 47 (2009), pp. 786–803.

[25] M. Jacquelin, M. Araya–Polo, and J. Meng, *Scalable distributed high-order stencil computations*, in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, 2022, pp. 1–13.

[26] X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, et al., *Design, implementation and testing of extended and mixed precision BLAS*, ACM Transactions on Mathematical Software (TOMS), 28 (2002), pp. 152–205.

[27] J. Loffeld and M. Tokman, *Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs*, Journal of Computational and Applied Mathematics, 241 (2013), pp. 45–67.

[28] J. Loffeld and M. Tokman, *Implementation of parallel adaptive-Krylov exponential solvers for stiff problems*, SIAM Journal on Scientific Computing, 36 (2014), pp. C591–C616.

[29] V. T. Luan, *Efficient exponential Runge–Kutta methods of high order: Construction and implementation*, BIT Numerical Mathematics, 61 (2021), pp. 535–560.

[30] V. T. Luan, M. Tokman, and G. Rainwater, *Preconditioned implicit-exponential integrators (IMEXP) for stiff PDEs*, Journal of Computational Physics, 335 (2017), pp. 846–864.

[31] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al., *Mixed precision training*, in International Conference on Learning Representations, 2018.

[32] D. Mukunoki and T. Ogita, *Performance and energy consumption of accurate and mixed-precision linear algebra kernels on GPUs*, Journal of Computational and Applied Mathematics, 372 (2020), p. 112701.

[33] S. Nandakumar, M. Le Gallo, C. Piveteau, V. Joshi, G. Mariani, I. Boybat, G. Karunaratne, R. Khaddam-Aljameh, U. Egger, A. Petropoulos, et al., *Mixed-precision deep learning based on computational memory*, Frontiers in neuroscience, 14 (2020), p. 406.

[34] J. Niesen and W. M. Wright, *Algorithm 919: A Krylov subspace algorithm for evaluating the ϕ-functions appearing in exponential integrators*, ACM Transactions on Mathematical Software (TOMS), 38 (2012), pp. 1–19.

[35] D. A. Pope, *An exponential method of numerical integration of ordinary differential equations*, Commun. ACM, 6 (1963), p. 491–493.

[36] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, *AI Accelerator Survey and Trends*, 2021 IEEE High Performance Extreme Computing Conference (HPEC), 00 (2021), pp. 1–9.

[37] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Transactions on Mathematical Software (TOMS), 24 (1998), pp. 130–156.

[38] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM Journal on Scientific Computing, 25 (2003), pp. 454–477.

[39] G. Söderlind, *The logarithmic norm. History and modern theory*, BIT Numerical Mathematics, 46 (2006), pp. 631–652.

[40] M. Tokman, *Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods*, Journal of Computational Physics, 213 (2006), pp. 748–776.

[41] S. UHLICH, L. MAUCH, F. CARDINAUX, K. YOSHIYAMA, J. A. GARCIA, S. TIEDEMANN, T. KEMP, AND A. NAKAMURA, *Mixed precision DNNs: All you need is a good parametrization*, in International Conference on Learning Representations, 2020.

[42] H. D. VO AND R. B. SIDJE, *Approximating the large sparse matrix exponential using incomplete orthogonalization and Krylov subspaces of variable dimension*, Numerical Linear Algebra with Applications, 24 (2017).