Leopold-Franzens-Universität

universität innsbruck

Master Thesis

Matrix-free Leja based exponential integrators in Python

Maximilian Samsinger

May 3, 2019

Supervised by Lukas Einkemmer and Alexander Ostermann

Leopold-Franzens-Universität Innsbruck

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Ich erkläre mich mit der Archivierung der vorliegenden Bachelorarbeit einverstanden.

Datum

Unterschrift

Matrix-free Leja based exponential integrators in Python

Abstract

1 Introduction

Consider the action of the matrix exponential function

$$e^A v, \quad A \in \mathbb{C}^{N\times N}, v \in \mathbb{C}^N.$$

It can be difficult or impossible to compute e^A in a first step and then the action $e^A v$ in a seperate step. This is especially true in applications where $N > 10000$ is not uncommon. Furthermore the matrix exponential of a sparse matrix is in general no longer sparse. Therefore it is more feasable to compute the action of the matrix exponential in a single step. This can be done by approximating the matrix exponential with a matrix polynomial p_n of degree n in A

$$e^A v \approx p_n(A)v.$$

This approach has many advantages. The cost of the computation of $p_n(A)v$ mainly depends on the calculation of $n \in \mathbb{N}$ matrix-vector multiplications with A. Not only can A be sparse, which significantly decreases the costs of the computation, the explicit knowledge of A itself is no longer required. A can be replaced by a linear function, which can be more convenient and saves memory.

2 Experiment 1

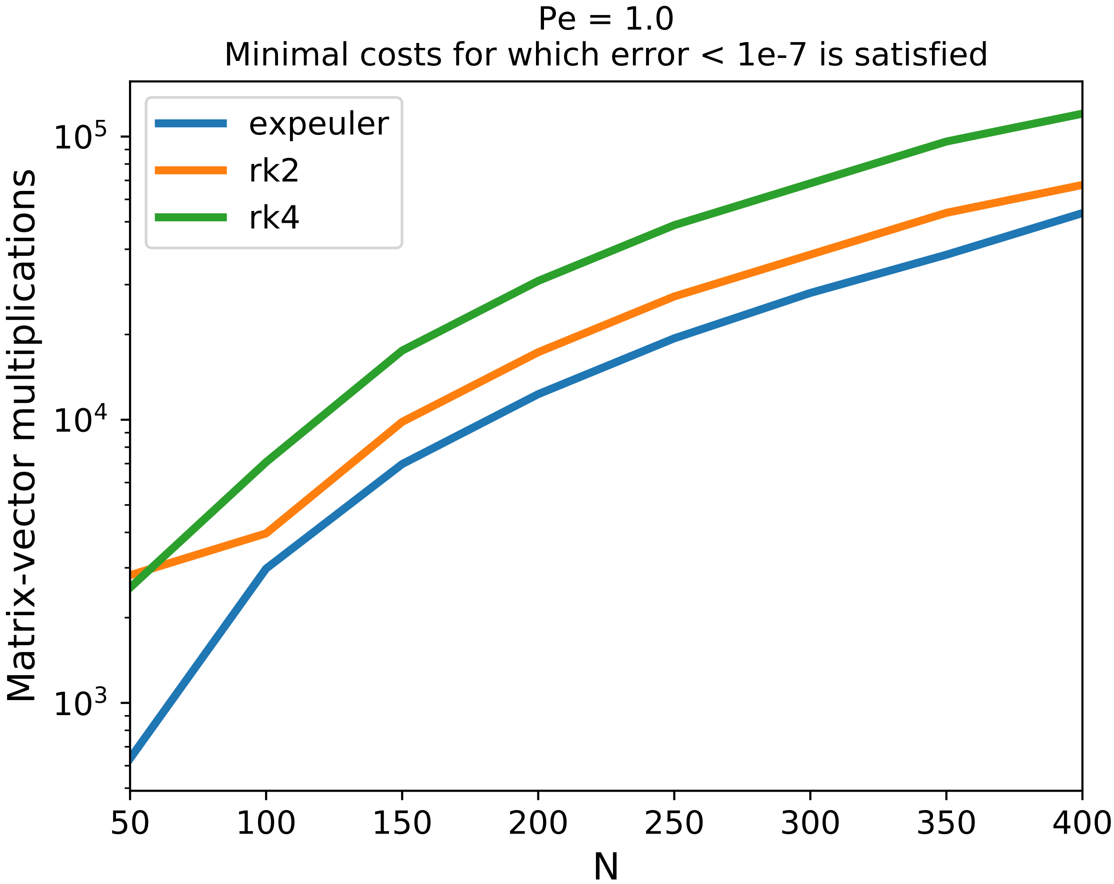
We discretize the one-dimensional advection-diffusion equation

$$\begin{aligned} \partial_t u &= a\partial_{xx}u + b\partial_x u & a, b \in [0,1] \\ u_0(t) &= e^{-80\cdot(t-0,45)^2} & t \in [0, 0.1] \end{aligned}$$

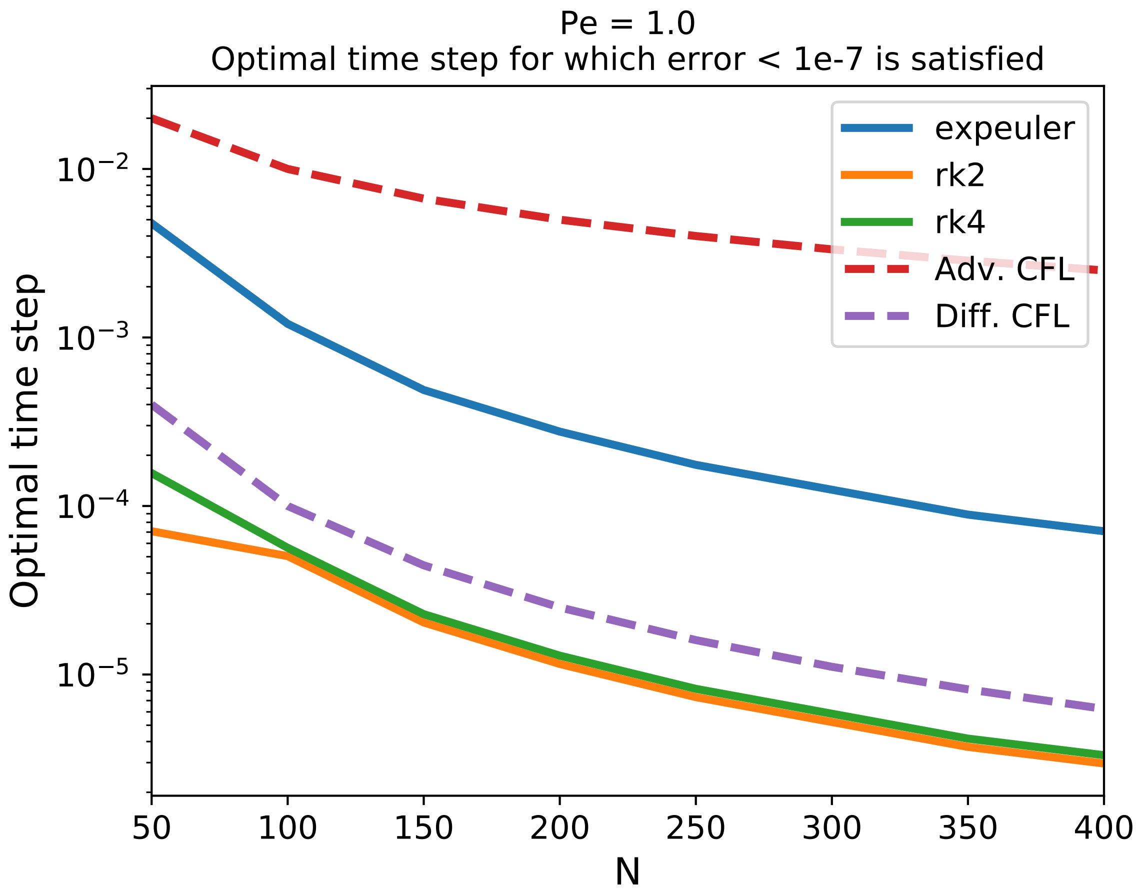
with homogeneous Dirichlet boundary conditions on the domain $\Omega = [0, 1]$.

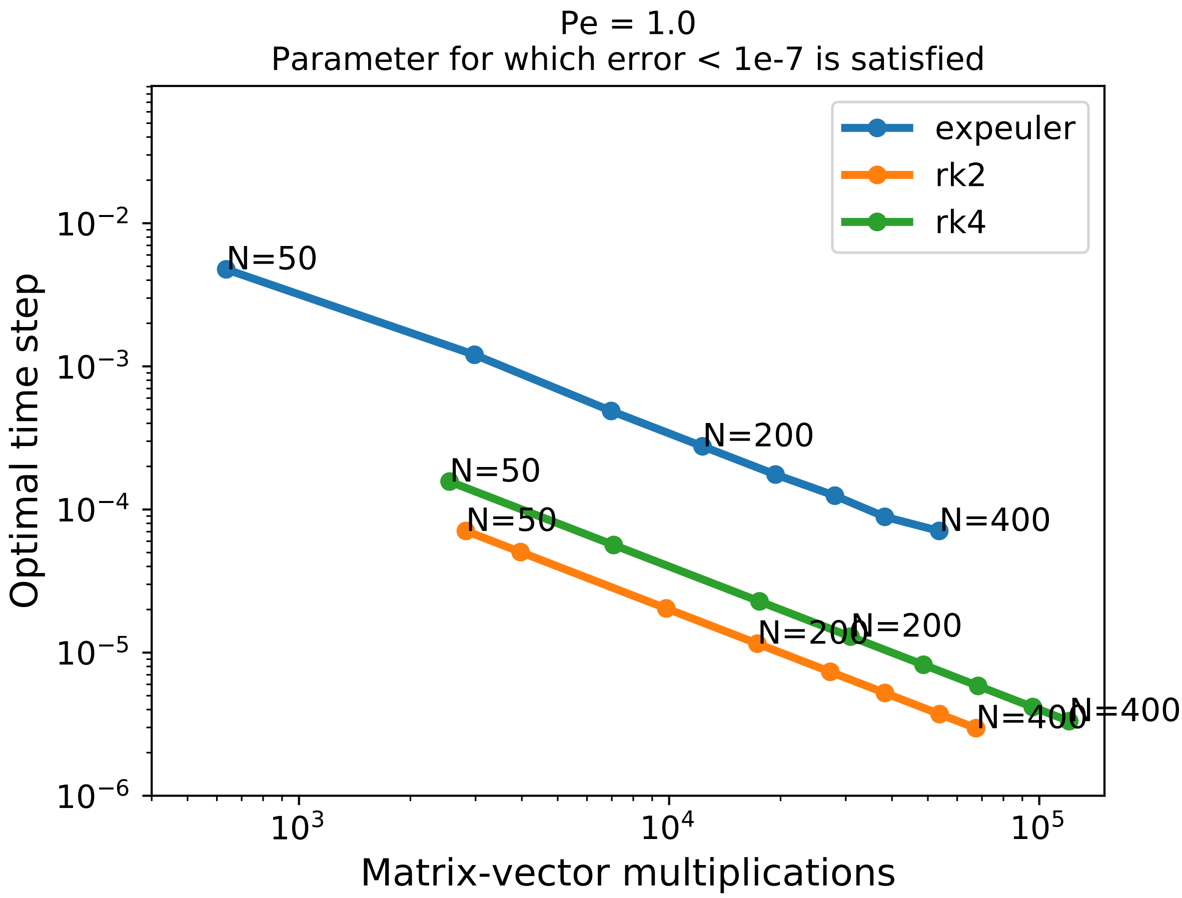
$$\mathrm{Pe} = \frac{\Delta x}{|c|}$$

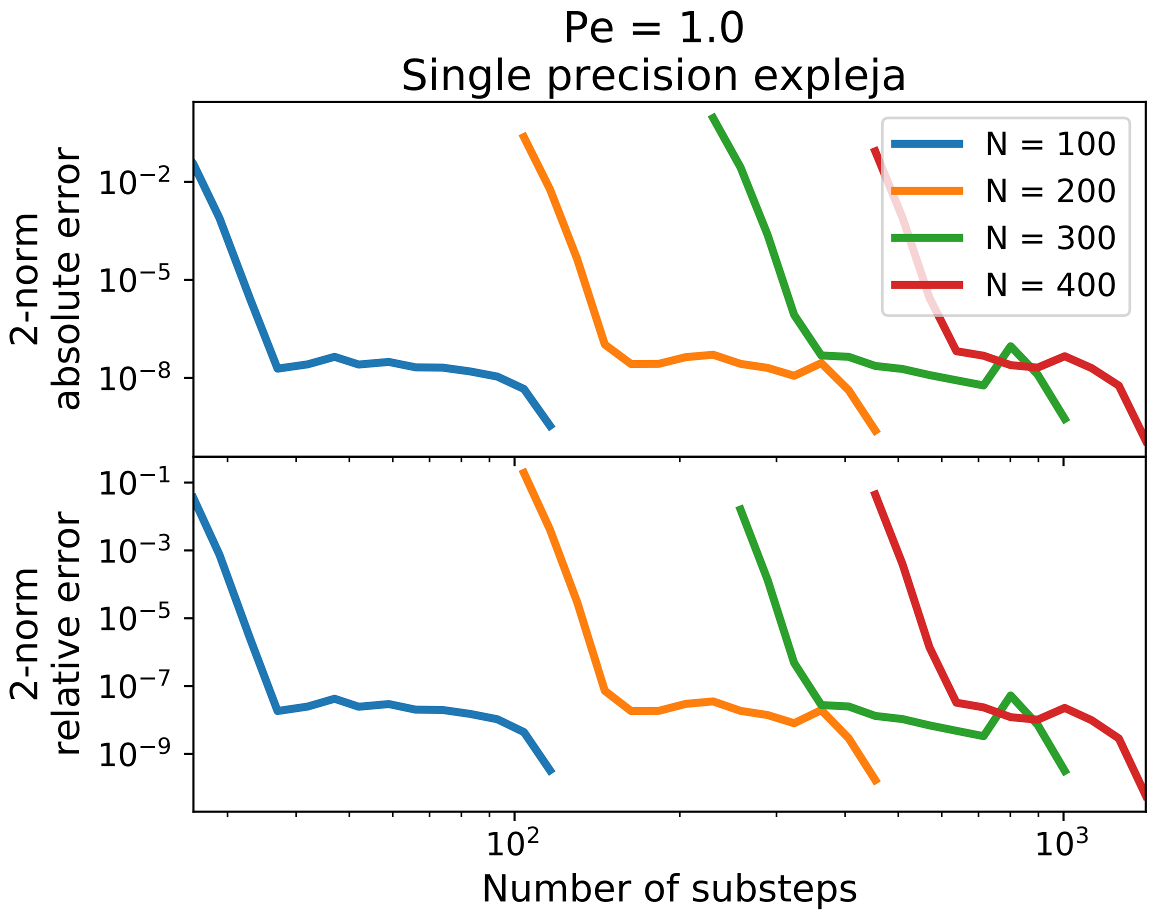
we denote the grid Péclet number. In the following experiments we will denote the advection-diffusion ratio, scaled by grid size $\Delta x = (N + 1)^{-1}$



Pe = 1.0
Minimal costs for which error < 1e-7 is satisfied
expeuler
rk2
rk4
Matrix-vector multiplications
10^5
10^4
10^3
50
100
150
200
250
300
350
400
N


Pe = 1.0
Optimal time step for which error < 1e-7 is satisfied
Optimal time step
N
expeuler
rk2
rk4
Adv. CFL
Diff. CFL
10^{-2}
10^{-3}
10^{-4}
10^{-5}
50
100
150
200
250
300
350
400


Pe = 1.0
Parameter for which error < 1e-7 is satisfied
expeuler
rk2
rk4
N=50
N=200
N=400
N=50
N=50
N=200
N=200
N=400
N=400
Optimal time step
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{3}
10^{4}
10^{5}
Matrix-vector multiplications



Question: Do we only care about the relative error? If yes, in which norm? Argument for 2

References

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

[2] A. Al-Mohy, N. Higham. Computing the action of the matrix exponential, with an application to exponential integrators, SIAM Journal on Scientific Computing 33 (2011), 488-511.

[3] L. Reichel. Newton interpolation at Leja points, BIT Numerical Mathematics 30 (1990), 332-346.

[4] M. Caliari, M. Vianello, L. Bergamaschi. Interpolating discrete advection-diffusion propagators at Leja sequences, Journal of Computational and Applied Mathematics 172 (2004), 79-99.

[5] M. Caliari, P. Kandolf, A. Ostermann, S. Rainer. The Leja method revisited: backward error analysis for the matrix exponential, SIAM Journal on Scientific Computation, Accepted for publication (2016). arXiv:1506.08665.

[6] Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org. Manual at https://docs.python.org/2/. [Online; accessed 2015-12-14]

[7] E. Jones, E. Oliphant, P. Peterson, SciPy: Open Source Scientific Tools for Python, Available at http://www.scipy.org/. [Online; accessed 2015-12-14]