Project Report on Stability-Optimized Explicit Runge-Kutta Methods

1. Introduction

This project focuses on solving systems of ordinary differential equations (ODEs) with mildly stiff terms. Commonly used explicit methods require a large number of timesteps to be taken to achieve the steady state solution, while implicit methods require a large amount of computational memory to store the solution at each step size. Stabilized explicit Runge-Kutta methods can be an efficient tool in solving problems because they have the stability domains extended along the negative real axis, having less severe step size restriction and costing less CPU memory.

Using low-storage Runge-Kutta methods (LSRK) is a way to stabilize the system by choosing significantly lower order p than s stages, and then using the remaining method degrees of freedom to maximize the linear stability region. Just as all other explicit methods, the LSRK schemes are only conditionally stable with an upper limit for the timestep. Since the limit is determined by the spectrum of the system of ODEs and the characteristic stability region, a specific scheme can be generated with tailored stability region.

In this project, three new generated fourth-order LSRK methods from Niegemann [1] were tested on three mildly stiff problems to demonstrate their solver efficiency.

2. Numerical methods

A Williamson scheme, or so-called “2*N*” schemes, where *N* is the number of equations being integrated times the number of grid points, have been chosen as an efficient ERK method to reduce memory. When solving the system of *N* coupled ODEs,

$$\frac{\partial}{\partial t}\mathbf{y}(t) = \mathbf{F}(t, \mathbf{y})$$

with an *s*-stage ERK method, a classical implementation requires the storage of the original *y*-vector, an intermediate *y*-vector, and all *s* function evaluations, leading to a total of $(s+1)N$ memory. Williamson assumes that it is only necessary to concerned with the memory requirement of a *y*-vector, what is in effect a *dy*-vector, and the memory of requirement of *F* is inconsequential. This scheme only requires $2N$ memory due to its independency of the number of stages *s,*

$$\begin{aligned}\mathbf{K}_1 &= \mathbf{y}_n \\ \left.\begin{aligned}\mathbf{K}_2 &= A_i\mathbf{K}_2 + \Delta t\mathbf{F}(t_n + c_i\Delta t, \mathbf{K}_1) \\ \mathbf{K}_1 &= \mathbf{K}_1 + B_i\mathbf{K}_2\end{aligned}\right\} & \forall i = 1 \dots s \\ \mathbf{y}_{n+1} &= \mathbf{K}_1\end{aligned}$$

In this work, since the spectrum of the ODE system is defined by the physical model and by the spatial discretization, which is limited to modify, the stability regions were be enlarged and tailored to generate the new fourth-order LSRK with 12, 13, and 14 stages, denoted as LSRK(s,p). For a given scheme to be of a certain order $p = 4$, the coefficients have to fulfill a number of $v_4 = 8$ order conditions, which are

$$\sum_{i=1}^{s} b_i = 1, \sum_{i=1}^{s} b_i c_i = \frac{1}{2}, \sum_{i,j=1}^{s} b_i a_{ij} c_j = \frac{1}{6}, \sum_{i,j,k=1}^{s} b_i a_{ij} a_{jk} c_k = \frac{1}{24}$$
$$\sum_{i=1}^{s} b_i c_i^2 = \frac{1}{3}, \sum_{i,j=1}^{s} b_i a_{ij} c_j^2 = \frac{1}{12}, \sum_{i,j=1}^{s} b_i c_i a_{ij} c_j = \frac{1}{8}, \sum_{i=1}^{s} b_i c_i^3 = \frac{1}{4}$$

Besides the criterion on order p, one more criterion is the stability such that for all complex eigenvalues $\lambda \in \mathbb{C}$ of the Jacobi matrix $J(t, \mathbf{y}) = \partial \mathbf{F} / \partial \mathbf{y}$,

$$|G(\lambda \Delta t)| \leqslant 1.$$

For ERK schemes, $G(z)$ takes the form of

$$G(z) = 1 + \gamma_1 z + \gamma_2 z^2 + \cdots + \gamma_s z^s,$$

where $z = \lambda \Delta t$ represents the rescaled complex eigenvalues and the coefficients γ are found as,

$$\gamma_i = \begin{cases} \displaystyle\sum_{j=1}^{s} b_j & \text{for } i = 1 \\ \mathbf{b}^T \mathcal{A}^{i-2} \mathbf{c} & \text{for } i > 1 \end{cases}$$

In order to generate new LSRK schemes, a set of optimized γ – values for a given number of stages were first determined from the desired shape of the stability domain, and then the LSRK scheme were found by fulfilling all required order conditions and the constraints shown above based on the γ – values.

The following tables show the Butcher coefficients of the new generated LSRK schemes,

Table 1

LSRK(12,4) coefficients (inviscid target stability region, cf. Fig. 3(a)).

i	γ_i	A_i	B_i	c_i
1	1	0.0000000000000000	0.0650008435125904	0.0000000000000000
2	1/2	−0.0923311242368072	0.0161459902249842	0.0650008435125904
3	1/6	−0.9441056581158819	0.5758627178358159	0.0796560563081853
4	1/24	−4.3271273247576394	0.1649758848361671	0.1620416710085376
5	$7.7793114345018587 \times 10^{-3}$	−2.1557771329026072	0.3934619494248182	0.2248877362907778
6	$1.2973631162180358 \times 10^{-3}$	−0.9770727190189062	0.0443509641602719	0.2952293985641261
7	$1.4820214027731423 \times 10^{-4}$	−0.7581835342571139	0.2074504268408778	0.3318332506149405
8	$1.8551101042762935 \times 10^{-5}$	−1.7977525470825499	0.6914247433015102	0.4094724050198658
9	$1.2351886928579280 \times 10^{-6}$	−2.6915667972700770	0.3766646883450449	0.6356954475753369
10	$1.2377768810554030 \times 10^{-7}$	−4.6466798960268143	0.0757190350155483	0.6806551557645497
11	$3.7434529900414887 \times 10^{-9}$	−0.1539613783825189	0.2027862031054088	0.7143773712418350
12	$3.1278890521988389 \times 10^{-10}$	−0.5943293901830616	0.2167029365631842	0.9032588871651854

Table 2

LSRK(13,4) coefficients (elliptical target stability region, cf. Fig. 3(b)).

i	γ_i	A_i	B_i	c_i
1	1	0.0000000000000000	0.0271990297818803	0.0000000000000000
2	1/2	−0.6160178650170565	0.1772488819905108	0.0271990297818803
3	1/6	−0.4449487060774118	0.0378528418949694	0.0952594339119365
4	1/24	−1.0952033345276178	0.6086431830142991	0.1266450286591127
5	$8.1116406653683835 \times 10^{-3}$	−1.2256030785959187	0.2154313974316100	0.1825883045699772
6	$1.2566761910282494 \times 10^{-3}$	−0.2740182222332805	0.2066152563885843	0.3737511439063931
7	$1.5605379767258244 \times 10^{-4}$	−0.0411952089052647	0.0415864076069797	0.5301279418422206
8	$1.5517942735576833 \times 10^{-5}$	−0.1797084899153560	0.0219891884310925	0.5704177433952291
9	$1.2224029698949826 \times 10^{-6}$	−1.1771530652064288	0.9893081222650993	0.5885784947099155
10	$7.4494312546583213 \times 10^{-8}$	−0.4078831463120878	0.0063199019859826	0.6160769826246714
11	$3.3568607387350691 \times 10^{-9}$	−0.8295636426191777	0.3749640721105318	0.6223252334314046
12	$1.0176127485607402 \times 10^{-10}$	−4.7895970584252288	1.6080235151003195	0.6897593128753419
13	$1.6382192183434098 \times 10^{-12}$	−0.6606671432964504	0.0961209123818189	0.9126827615920843

Table 3

LSRK(14,4) coefficients (circular target stability region, cf. Fig. 3(c)).

i	γ_i	A_i	B_i	c_i
1	1	0.0000000000000000	0.0367762454319673	0.0000000000000000
2	1/2	−0.7188012108672410	0.3136296607553959	0.0367762454319673
3	1/6	−0.7785331173421570	0.1531848691869027	0.1249685262725025
4	1/24	−0.0053282796654044	0.0030097086818182	0.2446177702277698
5	$8.0971474827892589 \times 10^{-3}$	−0.8552979934029281	0.3326293790646110	0.2476149531070420
6	$1.2380169165300218 \times 10^{-3}$	−3.9564138245774565	0.2440251405350864	0.2969311120382472
7	$1.4920544370587013 \times 10^{-4}$	−1.5780575380587385	0.3718879239592277	0.3978149645802642
8	$1.4105197862197588 \times 10^{-5}$	−2.0837094552574054	0.6204126221582444	0.5270854589440328
9	$1.0338060754675449 \times 10^{-6}$	−0.7483334182761610	0.1524043173028741	0.6981269994175695
10	$5.7551620074656494 \times 10^{-8}$	−0.7032861106563359	0.0760894927419266	0.8190890835352128
11	$2.3518316167532871 \times 10^{-9}$	0.0013917096117681	0.0077604214040978	0.8527059887098624
12	$6.6527970264862166 \times 10^{-11}$	−0.0932075369637460	0.0024647284755382	0.8604711817462826
13	$1.1639946786449694 \times 10^{-12}$	−0.9514200470875948	0.0780348340049386	0.8627060376969976
14	$9.4910013085549050 \times 10^{-15}$	−7.1151571693922548	5.5059777270269628	0.8734213127600976

3. Test problems

Three test problems were used to determine the accuracy and efficiency of the new LSRK schemes, and the forward Euler method and fourth-order explicit Runge-Kutta method (ERK4) were also included in the test for comparison. The first one is the basic test included in Niegemann [1],

$$\frac{\partial}{\partial t}u(t) = \frac{1}{u} - v\frac{\exp(t^2)}{t^2} - t$$
$$\frac{\partial}{\partial t}v(t) = \frac{1}{v} - \exp(t^2) - 2t\exp(-t^2)$$

with initial conditions $u(1) = 1, v(1) = \exp(-1)$, and the analytic solution is given by

$$u(t) = \frac{1}{t}, v(t) = \exp(-t^2).$$

In this project, the same initial conditions $t_0 = 1$ to $t_{\text{end}} = 1.4$ were used for various step sizes to reproduce similar results in Niegemann.

The second problem is an artificial test problem included in Cash [2],

$$\frac{dy}{dx} = -\lambda y + (\lambda - 1)e^{-x}, y(0) = 1,$$

which has the solution

$$y(x) = e^{-x}.$$

As mentioned in Cash [2], the problem becomes increasing stiff as λ increases, and can be regarded as being stiff when $\lambda = 100$. In this project, λ is ranged between 100 to 400 with a range of step sizes to check the performance of three new LSRK methods.

The third problem is a 1D Brusselator, and details can be found on Wikipedia [3]. The Brusselator is a theoretical model for a type of autocatalytic reaction. It is characterized by the reactions,

$$\begin{aligned} &A \to X \\ &2X + Y \to 3X \\ &B + X \to Y + D \\ &X \to E \end{aligned}.$$

When A and B are in vast excess, they can be modeled as constant concentration, and the rate equations become

$$\begin{aligned} &\frac{d}{dt}\{X\} = \{A\} + \{X\}^2\{Y\} - \{B\}\{X\} - \{X\} \\ &\frac{d}{dt}\{Y\} = \{B\}\{X\} - \{X\}^2\{Y\} \end{aligned}.$$

The rate constants of each term have all set to 1, same for the initial concentrations of two species. This system was computed in the unstable regime with $\Delta\text{t} = 0.1, A = 1, B = 3$. Both solution and phase space were plotted and compared with plots on Wikipedia.

4. Numerical results

Problem 1: Basic Test from Niegemann [1]

Table 4 Convergence Result of Basic Test

h	LSRK(12,4)	LSRK(13,4)	LSRK(14,4)	ERK4	Forward Euler
0.01	3.9681722903979852	3.8021248943913855	4.010526992795054	4.318577344451212	1.0121290353096029
0.001	3.789067925196068	4.051726003884108	4.145528696025226	4.038107009969611	1.0012038430271317
0.0001	0.5943611987234058	-0.5794454957535904	-0.8660257838590671	1.1940870521163023	1.0001203235908978

As shown in Table 4, all of three new methods achieved the theoretical convergence rate, and they all converged faster than the same order explicit Runge-Kutta method.

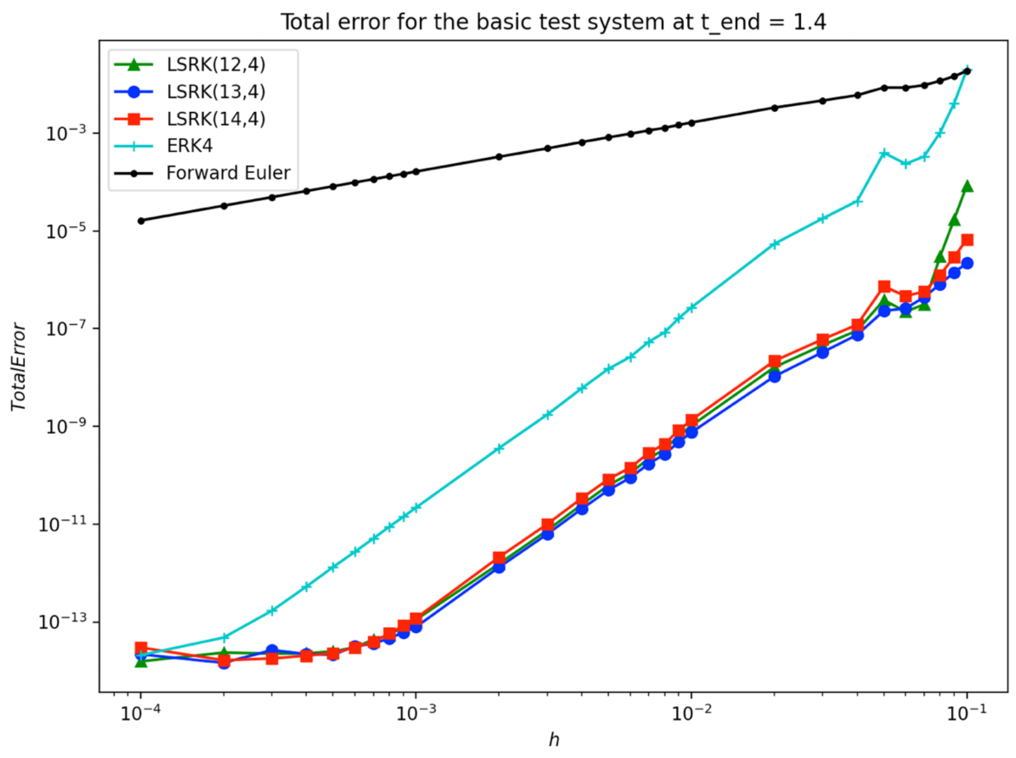


Figure 1 Total Error of Basic Test

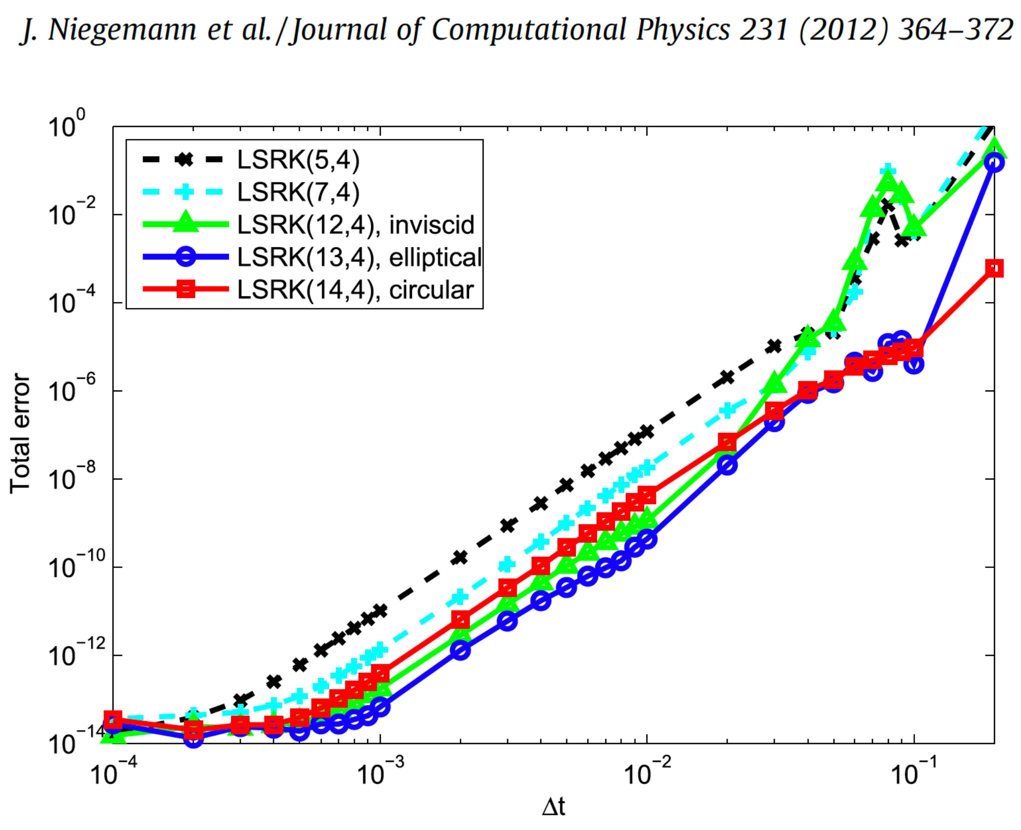
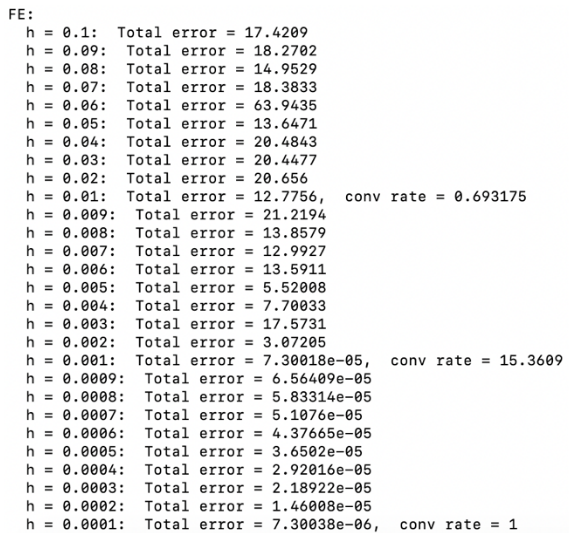


Figure 2 Total Error in Niegemann

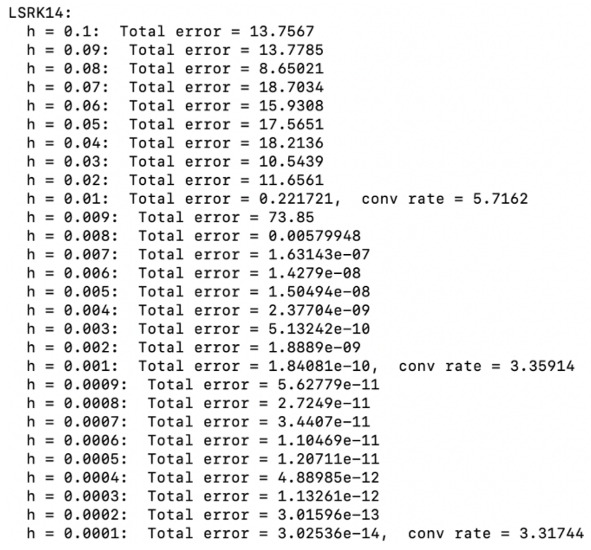
As shown above, Figure 1 is the plot of the basic test in this project, while Figure 2 is a screenshot of the result in Niegemann. By comparing three LSRK methods in both figures, the test in this project turned out running correctly, and they were all much more accurate than ERK and forward Euler method (FE).

One thing to point out is that there is another purpose of using FE to check the stiffness of the ODE system in this project. By taking the initial condition $t_{\text{end}} = 2.0$, the whole ODE problem became moderately stiff as the total error was 7.30018×10^5, ranging from 10^{-4} to 10^{-7} when it made sense at $\text{h} = 0.001$. Comparing with the best performance among the three new LSRK methods, LSRK(14,4) had a stable step size at $\text{h} = 0.007$.



```
FE:
  h = 0.1:  Total error = 17.4209
  h = 0.09:  Total error = 18.2702
  h = 0.08:  Total error = 14.9529
  h = 0.07:  Total error = 18.3833
  h = 0.06:  Total error = 63.9435
  h = 0.05:  Total error = 13.6471
  h = 0.04:  Total error = 20.4843
  h = 0.03:  Total error = 20.4477
  h = 0.02:  Total error = 20.656
  h = 0.01:  Total error = 12.7756,  conv rate = 0.693175
  h = 0.009:  Total error = 21.2194
  h = 0.008:  Total error = 13.8579
  h = 0.007:  Total error = 12.9927
  h = 0.006:  Total error = 13.5911
  h = 0.005:  Total error = 5.52008
  h = 0.004:  Total error = 7.70033
  h = 0.003:  Total error = 17.5731
  h = 0.002:  Total error = 3.07205
  h = 0.001:  Total error = 7.30018e-05,  conv rate = 15.3609
  h = 0.0009:  Total error = 6.56409e-05
  h = 0.0008:  Total error = 5.83314e-05
  h = 0.0007:  Total error = 5.1076e-05
  h = 0.0006:  Total error = 4.37665e-05
  h = 0.0005:  Total error = 3.6502e-05
  h = 0.0004:  Total error = 2.92016e-05
  h = 0.0003:  Total error = 2.18922e-05
  h = 0.0002:  Total error = 1.46008e-05
  h = 0.0001:  Total error = 7.30038e-06,  conv rate = 1
```

Figure 3 Total Error of FE when t_{end} = 2.0



```
LSRK14:
  h = 0.1:  Total error = 13.7567
  h = 0.09:  Total error = 13.7785
  h = 0.08:  Total error = 8.65021
  h = 0.07:  Total error = 18.7034
  h = 0.06:  Total error = 15.9308
  h = 0.05:  Total error = 17.5651
  h = 0.04:  Total error = 18.2136
  h = 0.03:  Total error = 10.5439
  h = 0.02:  Total error = 11.6561
  h = 0.01:  Total error = 0.221721,  conv rate = 5.7162
  h = 0.009:  Total error = 73.85
  h = 0.008:  Total error = 0.00579948
  h = 0.007:  Total error = 1.63143e-07
  h = 0.006:  Total error = 1.4279e-08
  h = 0.005:  Total error = 1.50494e-08
  h = 0.004:  Total error = 2.37704e-09
  h = 0.003:  Total error = 5.13242e-10
  h = 0.002:  Total error = 1.8889e-09
  h = 0.001:  Total error = 1.84081e-10,  conv rate = 3.35914
  h = 0.0009:  Total error = 5.62779e-11
  h = 0.0008:  Total error = 2.7249e-11
  h = 0.0007:  Total error = 3.4407e-11
  h = 0.0006:  Total error = 1.10469e-11
  h = 0.0005:  Total error = 1.20711e-11
  h = 0.0004:  Total error = 4.88985e-12
  h = 0.0003:  Total error = 1.13261e-12
  h = 0.0002:  Total error = 3.01596e-13
  h = 0.0001:  Total error = 3.02536e-14,  conv rate = 3.31744
```

Figure 4 Total Error of LSRK(14,4) when t_{end} = 2.0

Problem 2: Artificial Test from Cash [2]

As shown in Figure 5, as the number of stages increased, the method became more accurate and achieved a stable solution more easily. All three new LSRK methods performed better results than ERK4. As the λ increased, the problem became stiffer because a suitable step size for FE became smaller. When λ reached 400, LSRK(14,4) could still solve the problem well with a step size of 0.04, while both ERK4 and FE could only solve with a tiny step size around and under 0.005.

Table 5 Artificial Test Errors

Artificial Test Errors for lambda = 100					
h	LSRK(12,4)	LSRK(13,4)	LSRK(14,4)	ERK4	Forward Euler
0.1	1.1369300892276457e+301	9.813919378709457e-06	8.974541061468333e-06	1.6357159446697235e+292	3.445626160726274e+187
0.05	7.866904259053747e+237	2.6882752747758865e-06	1.5208407861400097e-06	1.041810830030867e+292	1.6557140887677843e+237
0.04	0.00028881079956472044	1.037338840936819e-06	7.184384178682279e-07	3.0175574499651486e+293	7.247100199278933e+234
0.03	9.540779832284407e-07	1.0892980156773646e-07	9.530087585751801e-08	3.57407306018657e+79	1.860170208400902e-05
0.02	1.4289452321403218e-07	4.5663977765730834e-08	5.176290446273768e-08	1.8831438933220035e-05	0.0001003299865974865
0.01	1.980305808224614e-09	1.9468561762714387e-09	3.144823901291005e-09	6.199530937500697e-07	1.8517059617029208e-05
0.005	3.717470775654874e-11	9.202483219894475e-11	1.8790857758688162e-10	3.07169527036244e-08	9.274187264374234e-06
0.001	2.1649348980190553e-15	1.1052270210143433e-13	2.8582691768974655e-13	4.055444868811264e-11	1.857348427392047e-06

Artificial Test Errors for lambda = 200					
h	LSRK(12,4)	LSRK(13,4)	LSRK(14,4)	ERK4	Forward Euler
0.1	1.8178183790343777e+304	6.222257098916985e+276	1.8258155786922653e+145	3.0968018870025794e+295	1.3689108806151259e+252
0.05	2.83886173587543e+300	2.454133943174952e-06	2.106991449091211e-06	4.1307148260064707e+291	2.8146742848111263e+301
0.04	1.54725417014981e+268	3.7424586654122294e-06	1.3341151902102055e-06	1.9706548189878066e+302	6.034989646610619e+291
0.03	5.5391093917763844e+299	1.9846678073198731e-07	9.98906616866968e-08	1.7900881210359808e+288	5.382766380033422e+295
0.02	7.354126707803665e-05	2.504966770122685e-07	1.7543152464050138e-07	2.5562187552220423e+275	6.71506680314718e+305
0.01	3.4682089755655454e-08	1.1013201128839967e-08	1.2731165177637394e-08	4.652654500447717e-06	2.5041457865895852e-05
0.005	4.752352400494431e-10	4.653216700845064e-10	7.752246622416692e-10	1.5413089810012437e-07	4.613850103696304e-06
0.001	4.2021941482062175e-14	4.48474590797332e-13	1.1440293157249926e-12	1.6975115757489334e-10	9.240098533536134e-07

Artificial Test Errors for lambda = 400					
h	LSRK(12,4)	LSRK(13,4)	LSRK(14,4)	ERK4	Forward Euler
0.1	2.3000255904644235e+285	1.715650725382779e+247	2.6061042647560053e+271	9.390021236352648e+294	2.431439170005829e+298
0.05	7.966987493041975e+241	1.0256841825782972e+236	1.1269563644147728e+294	7.812862375901839e+294	4.904519094760155e+302
0.04	2.361151738210404e+250	5.925681696025282e+253	2.1159139619864042e-06	3.9409751353298515e+245	5.212895958368881e+289
0.03	3.534045086832192e+246	8.540263373951352e+286	7.027608983634082e-08	1.2889084277284926e+275	2.0673262370434614e+274
0.02	1.027267713600833e+213	9.058981115805942e-07	3.2580963199579926e-07	4.948013651801913e+301	1.515094849115953e+291
0.01	1.6477747410315047e-05	6.15336533393851e-08	4.3344089373231753e-08	6.422242017091046e+274	2.3404430282012292e+281
0.005	8.542675566935998e-09	2.7032329708021052e-09	3.1568742064891353e-09	1.1563612684062363e-06	6.255195284473227e-06
0.001	5.822564652646633e-13	2.0213275497837913e-12	4.661993013854726e-12	7.456105732828178e-10	4.6084759980358214e-07

Problem 3: 1D Brusselator [3]

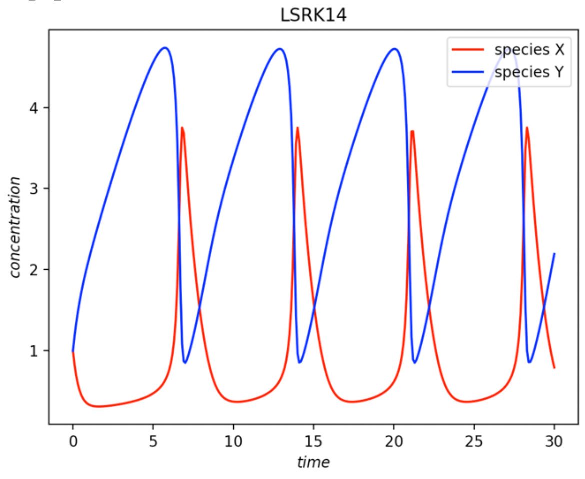


Figure 5 Solution for 1D Brusselator using LSRK(14,4)

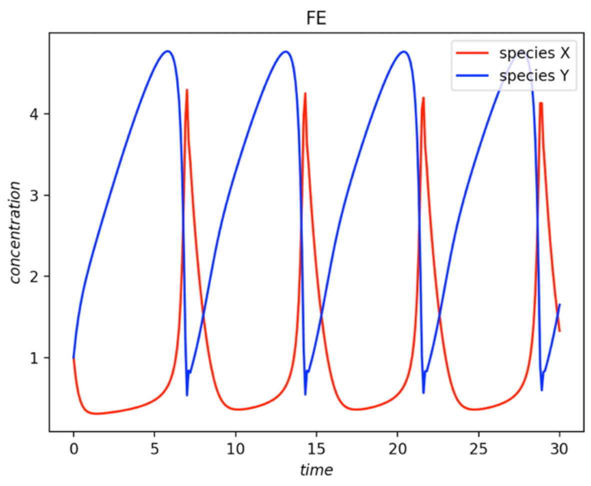


Figure 6 Solution for 1D Brusselator using Forward Euler

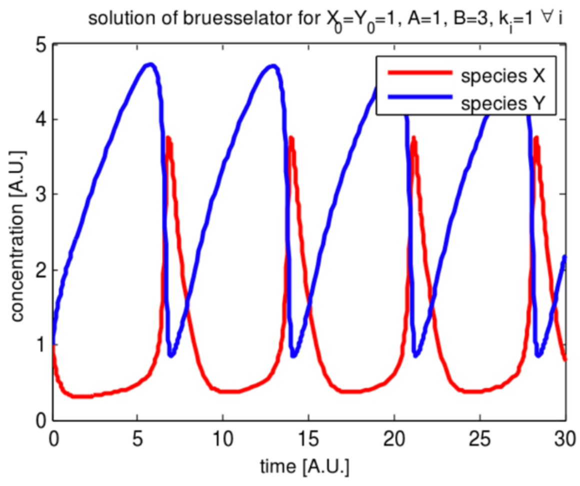


Figure 7 Brusselator in Wikipedia

As shown in Figure 5, using LSRK(14,4) method generated almost the same plot using "ode45" function in MATLAB as shown in Figure 7. The solution of using FE had a little bit of oscillations. This can be seen more clearly in Figure 8.

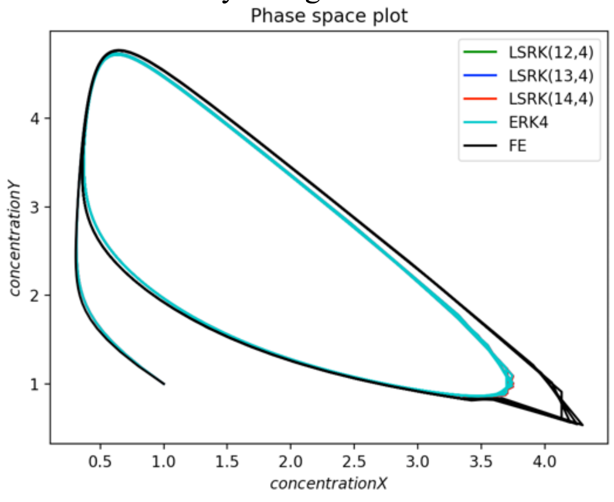


Figure 8 Phase Space of 1D Brusselator

5. Conclusion and outlook

All three new generated LSRK methods perform better dealing with mildly to moderately stiff terms than fourth order ERK. However, due to the limited stability region, they still cannot solve highly stiff problems. Instead, a partitioned implicit-explicit orthogonal Runge-Kutta method can be used to solve highly stiff diffusion-advection-reaction, employing an implicit treatment of the very stiff reaction terms, but an explicit treatment of the advection and diffusion terms.

In fact, many real-life numerical solutions in fields, such as electrodynamics, acoustics or hydrodynamics, are based on the method-of-lines, which is replacing spatial derivatives by finite difference approximations and solving a coupled ODE problem in time. This kind of problems usually occupy enormous amounts of computational memory to solve, and for complex physical systems, the number of coupled ODEs can easily reach $N \approx 10^8$ or higher. Therefore, using both implicit and stabilized explicit methods to treat different terms of a physical problem seems to be a trend.

Reference

[1] Niegemann, Diehl and Busch, "Efficient low-storage Runge-Kutta schemes with optimized stability regions," *Journal of Computational Physics*, 231:364-372, 2012.
[2] Cash, J. R., & Semnani, S, "A modified Adams method for nonstiff and mildly stiff initial value problems," *ACM Transactions on Mathematical Software*, 19(1), 63–80, 1993.
[3] "Brusselator," *Wikipedia*, Wikimedia Foundation, 5 Aug. 2020, https://en.wikipedia.org/wiki/Brusselator
[4] Moler, Cleve, "Stiff Differential Equations," *MathWorks*, 2003.