2019 18th European Control Conference (ECC)
Napoli, Italy, June 25-28, 2019

Numerical ODE solvers and integration methods in the computation of CPA Lyapunov functions

Sigurdur Freyr Hafstein*,1

Abstract— **Recently, several publications have been published, where continuous and piecewise-affine Lyapunov functions are constructed for nonlinear systems by numerically computing their values on a grid and then interpolating these values over the simplices of a simplicial complex. The value of such a Lyapunov function is computed at a grid point by numerically solving an initial-value problem and then integrating a positive definite function of the solution on a given time-interval. In this paper we systematically investigate how different initial-value solution methods compare in this application. Further, we propose a method to compute the integrals that is superior to former approaches.**

I. INTRODUCTION

We consider a general ordinary differential equation (ODE) of the form

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}), \text{ where } \mathbf{f}:\mathbb{R}^d \to \mathbb{R}^d \text{ and } \mathbf{f}(0)=0. \tag{1}$$

If $\mathbf{f}$ is locally Lipschitz the ODE (1) has a unique solution $t \mapsto \phi(t,\xi)$ for every initial-value $\xi \in \mathbb{R}^d$ and defines a dynamical system. The stability of the equilibrium at the origin, i.e. the constant solution $t \mapsto \phi(t,0)=0$, is usually studied within the framework of the Lyapunov stability theory. Its centerpiece, the Lyapunov function, is a scalar-valued function V from the state-space of the system, that has a minimum at the origin and is decreasing along all solution trajectories of the system in a neighbourhood of the equilibrium at the origin. If the Lyapunov function is C^1 the latter condition is equivalent to its orbital derivative being negative in a neighbourhood of the origin, i.e.

$$V'(\mathbf{x}) := \frac{d}{dt}\phi(t,\mathbf{x})\Big|_{t=0} = \nabla V(\mathbf{x})\bullet\mathbf{f}(\mathbf{x}) < 0, \ \forall \mathbf{x}\in\mathcal{N}\setminus\{0\},$$

where $\mathcal{N} \subset \mathbb{R}^d$ is an open neighbourhood of the origin. If V is merely locally Lipschitz the same proposition holds true with the orbital derivative defined as the Dini-derivative

$$V'(\mathbf{x}) := \limsup_{h\to 0+} \frac{V(\mathbf{x}+h\mathbf{f}(\mathbf{x}))-V(\mathbf{x})}{h}. \tag{2}$$

The condition that V has a minimum at the origin is usually stated

$$V(0)=0 \text{ and } V(\mathbf{x})>0 \text{ if } \mathbf{x}\neq 0.$$

The usefulness of the Lyapunov function comes from the fact that if one exists for the system (1), then the origin is asymptotically stable and if the connected component of the sublevel-set

$$\mathcal{L}_{V,m} := \{\mathbf{x}\in\mathbb{R}^d : V(\mathbf{x})\le m\}$$

containing the origin is a compact subset of the open neighbourhood $\mathcal{N}$, then it is a subset of the origin's basin of attraction. This holds for every $m>0$. We denote the intersection of this connected component and $\mathcal{N}$ by $\mathcal{L}^{cc}_{V,m}(0)$. Thus if there exists a Lyapunov function V for the system (1) and $\mathcal{L}^{cc}_{V,m}(0) \subset\subset \mathbb{R}^d$ (compact subset), then

$$\mathbf{x}\in\mathcal{L}^{cc}_{V,m}(0) \text{ implies } \lim_{t\to\infty}\phi(t,\mathbf{x})=0.$$

If local stability is a sufficiently strong concept for the problem at hand, one often resorts to linearizing the system (1) at the origin. It is well known that if the Jacobian $A := D\mathbf{f}(0)\in\mathbb{R}^{d\times d}$ is Hurwitz, i.e. all its eigenvalues have strictly negative real parts, then $V(\mathbf{x})=\mathbf{x}^TP\mathbf{x}$ is a Lyapunov function for the system, where $P\in\mathbb{R}^{d\times d}$ is the (unique) symmetric and positive definite solution to the so-called Lyapunov equation, a particular case of the Sylvester equation [35], $PA+PA^T=-I$, $I\in\mathbb{R}^{d\times d}$ being the identity matrix. Indeed $\|P^{-1}\|^{-1}\|\mathbf{x}\|^2 \le V(\mathbf{x}) \le \|P\|\|\mathbf{x}\|^2$ and $V'(\mathbf{x}) = -\|\mathbf{x}\|^2$, where $\|\bullet\|$ denotes the Euclidian norm.

If a reasonably sized lower bound on the basin of attraction for a nonlinear system is needed, one must resort to more advanced methods to compute a Lyapunov function for the system, that does take its nonlinearities into account.

II. LYAPUNOV FUNCTIONS FOR NONLINEAR SYSTEMS

For the reasons discussed in the last section there have been numerous methods proposed in the literature to generate Lyapunov functions for nonlinear systems. To name a few [30], [8], [9], [31], [1] parameterize Lyapunov functions that are the sums-of-squared (SOS) polynomials, see also the software [29]. For other polynomial methods cf. e.g. [32], [24], [34] and in particular [36], where the Zubov equation [39] is solved. Some other methods that rely on solving the Zubov equation are [7], [13]. In [23], [22], [26], [17] linear programming is used to parameterize Lyapunov functions, see also the software [4]. For an overview of numerical methods to compute Lyapunov functions see the recent review [15].

One approach is to approximate numerically formulas for Lyapunov functions [2], [18], [10], [11] from classical converse theorems [27], [38], [25] in the Lyapunov stability theory. These converse theorems assert the existence of Lyapunov functions for systems with asymptotically stable

*This research was supported by the Icelandic Research Fund (Rannís) in grants number 163074-052 and 152429-051, *Complete Lyapunov functions: Efficient numerical computation* and *Lyapunov Methods and Stochastic Stability* respectively.

1Sigurdur Freyr Hafstein is with the Science Institute, University of Iceland, Dunhagi 5, 107 Reykjavk, Iceland shafstein@hi.is

equilibria and give formulas, in terms of the systems's solution, for these Lyapunov functions. Because these formulas include the solutions to the systems, that are in general not obtainable for nonlinear systems, one resorts to approximate their values at a finite number of points. The Lyapunov function must be decreasing along solution trajectories in a whole neighbourhood of the equilibrium in question. If this cannot be asserted the constructed (Lyapunov) function is of little use, i.e. an approximation to a Lyapunov function is of little value. Therefore the computed values must be interpolated such that the resulting function is a true Lyapunov function in a whole area. This can be achieved by using the linear programming (LP) problem from [14], but instead of using LP to compute the values of the Lyapunov function at the vertices of a simplicial complex, one uses a formula from a converse theorem to fix its values at the vertices and then verifies if the linear constraints of the LP problem are fulfilled using these values. If the linear constraints are fulfilled for all vertices of a simplex, then the affine interpolation of these values over the simplex defines a function, whose orbital derivative is negative along all solution trajectories passing through this simplex. This was already shown in [2].

In this paper we will study different ODE solvers in this regard. The values $V(\mathbf{x})$ of a potential Lyapunov functions are computed on a grid using the formula

$$V(\mathbf{x}) = \int_0^T \alpha(\phi(\tau,\mathbf{x}))d\tau, \tag{3}$$

where $\alpha : \mathbb{R}^d \to \mathbb{R}$ is a positive definite function, i.e. $\alpha(0) = 0$ and $\alpha(\mathbf{x}) > 0$ for $\mathbf{x} \neq 0$, and $T > 0$ is a constant that is large enough. These grid points are also the vertices of the simplices of a triangulation used to create an LP problem as in [14]. Then these values are written into the variables of the LP problem and its linear constraints are verified. If the linear constraints are fulfilled for all vertices of a simplex in the triangulation, then we can be sure that the function obtained by interpolating the values at the grid points/vertices over the simplex is indeed decreasing along solution trajectories on the simplex. If one of the constraints is not fulfilled or if the ODE solver failed to deliver a value for one of the vertices, then we cannot guarantee that the function is decreasing along solution trajectories on that particular simplex. By counting the simplices where we can guarantee the decrease condition, we have a quantitative measure on how well the different ODE solution methods perform in this particular application.

Before we discuss the different ODE solution methods we compared, we propose a more exact numerical integration of the integral (3) than used in [3], [5], [19], [20].

III. The numerical integration

An approximation to the solution of system (1) with a particular initial-value $\phi(0,\xi) = \xi$ is computed at $N+1$ equally distributed time points on the time interval $[0,T]$:

$$\phi_i \approx \phi(t_i,\xi) \text{ at } t_i = \frac{iT}{N} \text{ for } i = 0:N, \text{ where} \tag{4}$$
$$N = 2^n m,\ m,n \in \mathbb{N} \text{ and } m \text{ is not divisible by 2.}$$

Here as elsewhere $0:N$ is an abbreviation for $0,1,\dots,N$. The integral in (3) for $V(\xi)$ is then approximated using a variant of the Romberg integration, cf. e.g. [33]. Set $\alpha_i := \alpha(\phi_i)$ for $i = 0:N$. First, we use the composite Trapezoidal rule to approximate $I_\xi = V(\xi)$ using $N, N/2, \dots, m$ intervals. For this define recursively $N_0 := N$ and $N_{k+1} := N_k/2$ for $k = 0:n-1$. Define $h_k := T/N_k$ for $k = 0:n$ and $B := (\alpha_0 + \alpha_N)/2$. We set

$$\text{Trap}_k = h_k\left(B + \sum_{j=1}^{N_k-1} \alpha_{j2^k}\right) \quad \text{for } k = 0:n. \tag{5}$$

It is well known that in the ideal case that $\alpha_i = \alpha(\phi(t_i,\xi))$ we have that $\text{Trap}_k = I_\xi + O(h_k^2)$ and by using extrapolation we get, using the tableau

$$R_{r,0} := \text{Trap}_r \quad \text{for } r = 0:n \tag{6}$$

and then for $s = 1:n$,

$$R_{r,s} = \frac{4^s R_{r,s-1} - R_{r+1,s-1}}{4^s - 1} \quad \text{for } 0 \le r \le n-s, \tag{7}$$

that $R_{0,n} = I_\xi + O(h^{2(n+1)})$ with $h = h_0 = T/N = T/(2^n m)$ as the length of the interval between two consecutive time-points the solution is computed at.

The advantage of using this formula is that Trap_k can be computed by setting $\text{Trap}_k = 0$ for $k = 0:n$, and then updating the values without the need to store old results of α_j, $j < i$. When ϕ_i is computed we add α_i to Trap_k, if and only if $i/2^k$ is an integer. This can easily be implemented in C/C++ as:

```
int r=i,k=0;
double alpha_i = alpha(phi_i);
Trap[k++] += value;
while(r%2 == 0){  // or !(r&1)
  r /= 2;         // or r >>= 1;
  Trap[k++] += alpha_i;
}

```

In [2], [5], [19], [20] the composite Simpson's rule was used to integrate the integral (3). This corresponds in our novel approach to using $R_{0,1}$ as an approximation of the integral.

IV. The ODE solution methods

For our comparison of numerical ODE solution methods we used the Euler method and Runge-Kutta methods of order 4 and 6, all of which are explicit one-step methods, the Adams-Bashforth explicit multi-step methods of order 4 and 5, and the Adams-Bashforth-Moulton predictor-corrector multi-step methods of order 4 and 5. The Euler method is only of first order, i.e. $\phi_i = \phi(t_i,\xi) + O(h^k)$ for $k = 1$ and with $0 \le t_i \le T$ and $h = t_{i+1} - t_i$. Note, however, that the order is quite misleading because the constant implicitly hidden in $O(h^k)$ can grow exponentially with T. The order of the other methods also refers to k in this formula, but they also have the same problem with the implicit constant. Note that we did not consider variable step-size methods. One of the reasons is that they would additionally complicate the numerical integration over the solutions.

For reference we write the formulas for the methods used. Here $t_i = ih$, and ϕ_i is the approximation to the solution $t \mapsto \phi(t,\xi)$ of $\mathbf{x}' = \mathbf{f}(\mathbf{x})$ at time t_i, in particular $\phi_0 = \xi$, and $\mathbf{f}_i = \mathbf{f}(\phi_i)$.

Euler method:

$$\phi_{i+1} = \phi_i + h\mathbf{f}_i$$

Runge-Kutta of order 4 (RK 4):

$$\begin{aligned}
\phi_{i+1} &= \phi_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)\\
\mathbf{k}_1 &= h\mathbf{f}_i\\
\mathbf{k}_2 &= h\mathbf{f}(\phi_i + \mathbf{k}_1/2)\\
\mathbf{k}_3 &= h\mathbf{f}(\phi_i + \mathbf{k}_2/2)\\
\mathbf{k}_4 &= h\mathbf{f}(\phi_i + \mathbf{k}_3)
\end{aligned}$$

Runge-Kutta of order 6 (RK 6):

$$\begin{aligned}
\phi_{i+1} &= \phi_i + \frac{h}{160}(23\mathbf{k}_1 + 58\mathbf{k}_3 + 58\mathbf{k}_4 - \mathbf{k}_5 - \mathbf{k}_6 + 23\mathbf{k}_7)\\
\mathbf{k}_1 &= h\mathbf{f}_i\\
\mathbf{k}_2 &= h\mathbf{f}(\phi_i + \mathbf{k}_1)\\
\mathbf{k}_3 &= h\mathbf{f}(\phi_i + 4/9\mathbf{k}_1 + 2/9\mathbf{k}_2)\\
\mathbf{k}_4 &= h\mathbf{f}(\phi_i + 11/36\mathbf{k}_1 + 1/9\mathbf{k}_2 - 1/12\mathbf{k}_3)\\
\mathbf{k}_5 &= h\mathbf{f}(\phi_i + 151/36\mathbf{k}_1 + 29/9\mathbf{k}_2 - 7/4\mathbf{k}_3 - 6\mathbf{k}_4)\\
\mathbf{k}_6 &= h\mathbf{f}(\phi_i - 112/9\mathbf{k}_1 - 116/9\mathbf{k}_2 + 32/3\mathbf{k}_3 + 18\mathbf{k}_4\\
&\qquad - 2\mathbf{k}_5)\\
\mathbf{k}_7 &= h\mathbf{f}(\phi_i - 5/4\mathbf{k}_1 - 29/23\mathbf{k}_2 + 397/276\mathbf{k}_3\\
&\qquad + 152/69\mathbf{k}_4 - 10/69\mathbf{k}_5 + 1/69\mathbf{k}_6)
\end{aligned}$$

Adams-Bashforth four-step method (AB 4); of order 4:

$$\phi_{i+1} = \phi_i + \frac{h}{24}(55\mathbf{f}_i - 59\mathbf{f}_{i-1} + 37\mathbf{f}_{i-2} - 9\mathbf{f}_{i-3})$$

Adams-Bashforth five-step method (AB 5); of order 5:

$$\begin{aligned}
\phi_{i+1} = \phi_i + \frac{h}{720}(&1901\mathbf{f}_i - 2774\mathbf{f}_{i-1} + 2616\mathbf{f}_{i-2}\\
&- 1274\mathbf{f}_{i-3} + 251\mathbf{f}_{i-4})
\end{aligned}$$

Adams-Bashforth-Moulton predictor-corrector method of order 4 (ABM 4); four-step predictor and three-step corrector:

$$\begin{aligned}
\phi_{i+1} &= \phi_i + \frac{h}{24}(9\widetilde{\mathbf{f}}_{i+1} + 19\mathbf{f}_i - 5\mathbf{f}_{i-1} + 5\mathbf{f}_{i-2})\\
\widetilde{\phi}_{i+1} &= \phi_i + \frac{h}{24}(55\mathbf{f}_i - 59\mathbf{f}_{i-1} + 37\mathbf{f}_{i-2} - 9\mathbf{f}_{i-3})\\
\widetilde{\mathbf{f}}_{i+1} &= \mathbf{f}(\widetilde{\phi}_{i+1})
\end{aligned}$$

Adams-Bashforth-Moulton predictor-corrector method of order 5 (ABM 5); five-step predictor and four-step corrector:

$$\begin{aligned}
\phi_{i+1} &= \phi_i + \frac{h}{720}(251\widetilde{\mathbf{f}}_{i+1} + 646\mathbf{f}_i - 264\mathbf{f}_{i-1} + 106\mathbf{f}_{i-2}\\
&\qquad - 19\mathbf{f}_{i-3})\\
\widetilde{\phi}_{i+1} &= \phi_i + \frac{h}{720}(1901\mathbf{f}_i - 2774\mathbf{f}_{i-1} + 2616\mathbf{f}_{i-2}\\
&\qquad - 1274\mathbf{f}_{i-3} + 251\mathbf{f}_{i-4})\\
\widetilde{\mathbf{f}}_{i+1} &= \mathbf{f}(\widetilde{\phi}_{i+1})
\end{aligned}$$

A few comments are in order. The Euler method is only included for reference; we expect it to be the fastest but also to deliver the worst results because it is only of order one. The standard Runge-Kutta method of order 4 (RK 4) is often considered to be the workhorse of ODE solvers; it is easy to implement and fairly accurate. Further, if one wants a higher-order Runge-Kutta method than four one needs at least one more $\mathbf{k}_i$ as an intermediate step than the order of the method, cf. [6], where we also got our Runge-Kutta method of order 6 from. Note that there are several Runge-Kutta Methods of any order. We expect the Adams-Bashforth multi-step methods to be faster than the Runge-Kutta methods, because no intermediate steps (the $\mathbf{k}_i$s) are needed. Indeed, one uses the previous values of ϕ_j, $j < i$, when computing ϕ_i, hence the name 'multi-step methods'. The initial steps were computed using the RK 6 method. The Adams-Bashforth-Moulton predictor-corrector methods use the Adams-Bashforth multi-step methods to predict the value at next step, but then try to improve it using a step of the implicit Adams-Moulton method of same order. All these methods are commonly discussed in introductory books on numerical solutions to ODE, cf. e.g. [33]. A more detailed discussion is given in, for example, [21].

V. The results

As our benchmark systems to compare the different ODE solution methods for our application we choose three of the systems investigated in [20], where they were studied using the Adams-Bashforth method of fourth-order (AB 4) to solve the initial-value problems and the composite Simpson's rule was used to compute the numerical integrals. The upper-bounds $B^{\nu}_{i,j}$ in the LP problem were fixed with the same values as there and we we refer the interested reader to that paper for more detailed information on the setup, the results, and their interpretation. Here we only compare the performance and the quality of the results of the different ODE solution methods. Two different functions $\alpha : \mathbb{R}^d \to \mathbb{R}$ are used in the integral (3), the canonical $\alpha(\mathbf{x}) = \|\mathbf{x}\|^2$, which delivers the formula

$$V(\mathbf{x}) = \int_0^T \|\phi(\tau,\mathbf{x})\|^2 d\tau \quad (1a, 2a, 3a) \tag{8}$$

for the Lyapunov function candidate and the results from this integral are labelled (1a) for Example 1, (2a) for Example 2, and (3a) for Example 3. We also use a more advanced α that was shown in [19], [20] to deliver superior results in

	Euler's Method for reference
Example	Results ★ Computation time
1a	1,214,782 of 8,000,000 Failed (15.184775%) ★ Computed in 55.1 seconds
1b	1,429,778 of 8,000,000 Failed (17.872225%) ★ Computed in 77.7 seconds
2a	1,991,636 of 8,000,000 Failed (24.89545%) ★ Computed in 29.7 seconds
2b	1,996,344 of 8,000,000 Failed (24.9543%) ★ Computed in 52.2 seconds
3a	7,770,876 of 24,000,000 Failed (32.37865%) ★ Computed in 31.8 seconds
3b	8,694,822 of 24,000,000 Failed (36.22843%) ★ Computed in 31.8 seconds
3b†	8,851,776 of 24,000,000 Failed (36.8824%) ★ Computed in 3.65 seconds

	RK 4	AB 4	ABM 4	AB 5	ABM 5	RK 6
Ex.	Failed ★ Time	Failed ★ Time	Failed ★ Time	Failed ★ Time	Failed ★ Time	Failed ★ Time
1a	-20.72% ★ +322%	-21.04% ★ +50%	-20.70% ★ +169%	-20.83% ★ +63%	-20.70% ★ +169%	-20.72% ★ +654%
1b	-53.80% ★ +229%	-53.89% ★ +38%	-53.80% ★ +119%	-53.84% ★ +40%	-53.80% ★ +122%	-53.80% ★ +455%
2a	-0.835% ★ +378%	-0.835% ★ +93%	-0.835% ★ +257%	-0.835% ★ +115%	-0.835% ★ +247%	-0.835% ★ +768%
2b	-0.962% ★ +218%	-0.962% ★ +51%	-0.962% ★ +139%	-0.962% ★ +64%	-0.962% ★ +137%	-0.962% ★ +435%
3a	+0.122% ★ +386%	+0.122% ★ +94%	+0.122% ★ +253%	+0.122% ★ +125%	+0.122% ★ +277%	+0.122% ★ +816%
3b	-0.480% ★ +377%	-0.481% ★ +51%	-0.480% ★ +139%	-0.480% ★ +64%	-0.480% ★ +137%	-0.480% ★ +829%
3b†	-0.826% ★ +336%	-2.029% ★ +116%	-0.703% ★ +262%	+8.211% ★ +141%	-0.904% ★ +262%	-0.830% ★ +708%

TABLE I

THE RESULTS OF OUR NUMERICAL EXPERIMENTS SUMMARIZED IN TWO TABLES. THE UPPER TABLE GIVES ABSOLUTE VALUES FOR THE EULER METHODS AND THE LOWER GIVES VALUES RELATIVE TO THE EULER METHOD FOR THE OTHER METHODS. FOR A DETAILED DISCUSSION OF THE RESULTS AND THE EXAMPLES SEE THE TEXT IN SECTION V.

the sense, that a much larger lower bound on the basin of attraction can be asserted. The formula is

$$V(\mathbf{x}) = \int_0^T \frac{\|\phi(\tau,\mathbf{x})\|^2}{\delta + \|\phi(\tau,\mathbf{x})\|^p} d\tau, \quad (1b, 2b, 3b, 3b^\dagger) \qquad (9)$$

where the parameters δ and p are specifically chosen for the system at hand. The corresponding results are labelled (1b) for Example 1, (2b) for Example 2, and (3b) and (3b†) for Example 3.

Example 1 is a planar system from [12],

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) \quad \text{with} \quad \mathbf{f}(x,y) = \begin{pmatrix} -x+y \\ 0.1x - 2y - x^2 - 0.1x^3 \end{pmatrix}. \quad \text{(Ex.1)}$$

We set $T = 20$ in the formulas (8) and (9) for a Lyapunov function, and for the latter we set $\delta = 0.6$ and $p = 1.2$. The grid used for the vertices of the simplices was in both cases 2001×2001 with 4,004,001 points/vertices and 8,000,000 simplices/triangles. The computation of the Lyapunov function using formula (8) was done on the rectangle $[-20,20]^2$ and the computation of the Lyapunov function using formula (9) was done on the rectangle $[-20,20] \times [-40,40]$. In both cases we used $N = 1{,}000$ in (4). Since $1{,}000 = 2^3 \cdot 125$ the numerical integration method is of order $2 \cdot (3+1) = 8$

Example 2 is a planar system from [37],

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) \;\text{ with }\; \mathbf{f}(x,y) = \begin{pmatrix} -0.84x - 1.44y - 0.3xy \\ 0.54x + 0.34y + 0.3xy \end{pmatrix}. \quad \text{(Ex.2)}$$

Again we set $T = 20$ in formulas (8) and (9) and for the latter we set $\delta = 0.3$ and $p = 1.4$. As in Example 1 the grid used for the vertices of the simplices was 2001×2001 with 4,004,001 points/vertices and 8,000,000 simplices/triangles. The computation of the Lyapunov function using both formula (8) and formula (9) was done on the rectangle $[-8,8] \times [-2,8]$ and again we used $N = 1{,}000$ in (4); thus the numerical integration is of order 8.

Example 3 is a three-dimensional system from [16], also studied in [28]:

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) \quad \text{with} \quad \mathbf{f}(x,y,z) = \begin{pmatrix} -x+y+z^2 \\ -y+xy \\ -z \end{pmatrix}. \qquad \text{(Ex.3)}$$

We set $T = 10$ in formulas (8) and (9) and for latter we set $\delta = 1$ and $p = 2$. The grid was in both cases $201 \times 201 \times 101$ with $4{,}080{,}501$ points and 24,000,000 simplices/tetrahedra. The computation of the Lyapunov function using both formula (8) and (9) was done on the cube $[-8,3] \times [-3,8] \times [-2,2]$. Here we used $N = 1{,}000$ in (3a) and (3b), resulting in a numerical integration method of order 8. In (3b†) we did an experiment with $N = 100$, which results in a numerical integration method of order 6, but much faster computations.

The results for these examples are presented in Table I. In the upper table we give absolute numbers for the computational times and results when using the Euler method for all the systems. For the interpretation of the results we exemplary discuss the first line: '1a' means that on the right are the results for Example 1 using formula (8) (1b would mean that formula (9) is used). '1,214,782 of 8,000,000 Failed' means that in $1{,}214{,}782$ of the $8{,}000{,}000$ triangles/simplices the method failed to guarantee that the orbital derivative of the corresponding Lyapunov function candidate has a negative orbital derivative on the triangle/simplex. The reason for this can be threefold. First, it might be that the orbital derivative is actually negative on the simplex, if we were able to compute the values of (8) or (9) exactly, but that our method fails to delivers values that are good enough. Second, it might be that the orbital derivative is not negative on the simplex, even when using exact values. Third, it might be that the Lyapunov function candidate using formulas (8) or (9) is not defined on the simplex, because of finite-time blowup of solutions. Anyways, we would like this number to be as low as possible and this number is the 'quality' measure we have on the different ODE solution methods for our application. 'Computed in 55.1 seconds' is self explanatory. The computer used had an i7-7700K CPU (4 cores@4.2GHz) and the methods were coded in C++.

In the lower table we give results for the other numerical ODE solution methods relative to the Euler method. Again let us explain the table by discussing its first line: '1a' has the same meaning as in the table above. '-20.72%' (Failed) means that in $1,214,782 \cdot (1-0.2072) = 963,079$ simplices the RK 4 method failed to guarantee the negativity of the orbital derivative, i.e. in 20.72% fewer simplices than the Euler method, which is a much improved result. '+322%' (Time) means that the RK 4 methods needed 322% more time than the Euler method, i.e. $55.1 \cdot (1+3.22) = 233$ seconds, to deliver the results.

Additional to the results in the table we also run Example 1b with a computer with a faster CPU (i9-7900X, 10 cores@3.3GHz), to see how well the method parallelizes. It needed 272 seconds vs. the 432 seconds the i7-7700K needed, i.e. about 37% less time, which is not too far off from a simply estimated maximum improvement of $4 \times 4.2\,\text{GHz}/(10 \cdot 3.3\,\text{GHz}) \approx 51\%$. Thus, as expected, this method to compute CPA Lyapunov functions parallelizes quite well with the number of cores.

VI. Discussion

From the results in Table I we conclude that the Adams-Bashforth method of fourth order (AB 4) is the most appropriate method to evaluate the Lyapunov function candidate at the vertices of the triangulation. It need about 50-100% more time than the Euler method, but delivers considerably better results. Using a higher-order Adams-Bashforth method (AB 5) adds very little to the quality of the results at the cost of ca. 10% added computational time.

Somewhat surprisingly the addition of an Adams-Moulton correction step does not seem to have any positive effect on the quality of the results and that, not so surprisingly, at the considerable cost of ca. 75% added computational time. The Runge-Kutta methods deliver just as good results as the Adams-Bashforth methods, but are computationally much more expensive because of their intermediate steps.

Note that this conclusion is drawn from our somewhat limited computational experience. A more detailed study might consider other parameters, for example, the minimum size of time-steps for the ODE solution methods to deliver qualitatively good results, more ODE solution methods, higher-dimensional systems, etc.

VII. Conclusions

We compared several ordinary differential equation solution methods for the numerical computation of Lyapunov function candidates for nonlinear systems. The Lyapunov function candidates are computed by approximating their values at the vertices of the simplices a triangulation and then interpolation these values over the simplices. The quality of the results can be quantified as the proportion of the simplices, of which we can guarantee that the Lyapunov function candidate is decreasing along solution trajectories of the system. Our results indicate that the Adams-Bashforth multi-step method of fourth order (AB 4) is the most effective method in terms of the quality of the results and the computational time. Further, we presented a Romberg integration scheme for our application, which allows for the numerical integration over solution trajectories with higher accuracy at practically none added computational cost in comparison to earlier approaches.

References

[1] J. Anderson and A. Papachristodoulou. Advances in computational Lyapunov analysis using sum-of-squares programming. *Discrete Contin. Dyn. Syst. Ser. B*, 20(8):2361–2381, 2015.

[2] J. Björnsson, P. Giesl, S. Hafstein, C. Kellett, and H. Li. Computation of continuous and piecewise affine Lyapunov functions by numerical approximations of the Massera construction. In *Proceedings of the CDC, 53rd IEEE Conference on Decision and Control*, Los Angeles (CA), USA, 2014.

[3] J. Björnsson, P. Giesl, S. Hafstein, C. Kellett, and H. Li. Computation of Lyapunov functions for systems with multiple attractors. *Discrete Contin. Dyn. Syst. Ser. A*, 35(9):4019–4039, 2015.

[4] J. Björnsson, Gudmundsson, and S. Hafstein. Class library in C++ to compute Lyapunov functions for nonlinear systems. In *Proceedings of MICNON, 1st Conference on Modelling, Identification and Control of Nonlinear Systems*, number 0155, pages 788–793, 2015.

[5] J. Björnsson and S. Hafstein. Efficient Lyapunov function computation for systems with multiple exponentially stable equilibria. *Procedia Computer Science*, 108:655–664, 2017. Proceedings of the International Conference on Computational Science (ICCS), Zurich, Switzerland, 2017.

[6] J. Butcher. On Runge-Kutta processes of high order. *J. Austral. Math. Soc.*, 4(2):179–194, 1964.

[7] F. Camilli, L. Grüne, and F. Wirth. A generalization of Zubov's method to perturbed systems. *SIAM J. Control Optim.*, 40(2):496–515, 2001.

[8] G. Chesi. LMI techniques for optimization over polynomials in control: a survey. *IEEE Trans. Automat. Control*, 55(11):2500–2510, 2010.

[9] G. Chesi. *Domain of Attraction: Analysis and Control via SOS Programming*. Springer, 2011.

[10] A. Doban. *Stability domains computation and stabilization of nonlinear systems: implications for biological systems*. PhD thesis: Eindhoven University of Technology, 2016.

[11] A. Doban and M. Lazar. Computation of Lyapunov functions for nonlinear differential equations via a Yoshizawa-type construction. *IFAC-PapersOnLine*, 49(18):29 – 34, 2016.

[12] R. Genesio, M. Tartaglia, and A. Vicino. On the estimation of asymptotic stability regions: State of the art and new proposals. *IEEE Trans. Automat. Control*, 30(8):747–755, 1985.

[13] P. Giesl. *Construction of Global Lyapunov Functions Using Radial Basis Functions*. Lecture Notes in Math. 1904, Springer, 2007.

[14] P. Giesl and S. Hafstein. Revised CPA method to compute Lyapunov functions for nonlinear systems. *J. Math. Anal. Appl.*, 410:292–306, 2014.

[15] P. Giesl and S. Hafstein. Review of computational methods for Lyapunov functions. *Discrete Contin. Dyn. Syst. Ser. B*, 20(8):2291–2331, 2015.

[16] O. Hachicho and B. Tibken. Estimating domains of attraction of a class of nonlinear dynamical systems with LMI methods based on the theory of moments,. In *Proceedings of the 41th IEEE Conference on Decision and Control (CDC)*, pages 3150–3155, Los Angeles (CA), USA, 2002.

[17] S. Hafstein. *An algorithm for constructing Lyapunov functions*, volume 8 of *Monograph*. Electron. J. Diff. Eqns., 2007.

[18] S. Hafstein, C. Kellett, and H. Li. Computing continuous and piecewise affine Lyapunov functions for nonlinear systems. *Journal of Computational Dynamics*, 2(2):227 – 246, 2015.

[19] S. Hafstein and A. Valfells. Study of dynamical systems by fast numerical computation of Lyapunov functions. In *Proceedings of the 14th International Conference on Dynamical Systems: Theory and Applications (DSTA)*, volume Mathematical and Numerical Aspects of Dynamical System Analysis, pages 220–240, 2017.

[20] S. Hafstein and A. Valfells. Efficient computation of Lyapunov functions for nonlinear systems by integrating numerical solutions. *submitted*, 2018.

[21] E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 3rd edition, 2008.

[22] T. Johansen. Computation of Lyapunov functions for smooth, nonlinear systems using convex optimization. *Automatica*, 36:1617–1626, 2000.

[23] P. Julian, J. Guivant, and A. Desages. A parametrization of piecewise linear Lyapunov functions via linear programming. *Int. J. Control*, 72(7-8):702–715, 1999.

[24] R. Kamyar and M. Peet. Polynomial optimization with applications to stability analysis and control – an alternative to sum of squares. *Discrete Contin. Dyn. Syst. Ser. B*, 20(8):2383–2417, 2015.

[25] C. Kellett. Converse Theorems in Lyapunov's Second Method. *Discrete Contin. Dyn. Syst. Ser. B*, 20(8):2333–2360, 2015.

[26] S. Marinósson. Lyapunov function construction for ordinary differential equations with linear programming. *Dynamical Systems: An International Journal*, 17:137–150, 2002.

[27] J. Massera. Contributions to stability theory. *Annals of Mathematics*, 64:182–206, 1956. (Erratum. Annals of Mathematics, 68:202, 1958).

[28] L. Matallana, A. Blanco, and J. Bandoni. Estimation of domains of attraction: A global optimization approach. *Math. Comput. Modelling*, 52(3-4):574–585, 2010.

[29] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Pranja, P. Seiler, and P. Parrilo. *SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB*. User's guide. Version 3.00 edition, 2013.

[30] P. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimiza*. PhD thesis: California Institute of Technology Pasadena, California, 2000.

[31] M. Peet and A. Papachristodoulou. A converse sum of squares Lyapunov result with a degree bound. *IEEE Trans. Automat. Control*, 57(9):2281–2293, 2012.

[32] S. Ratschan and Z. She. Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov-like functions. *SIAM J. Control Optim.*, 48(7):4377–4394, 2010.

[33] T. Sauer. *Numerical Analysis*. Pearson, 2nd edition, 2012.

[34] Z. She, H. Li, B. Xue, Z. Zheng, and B. Xia. Discovering polynomial Lyapunov functions for continuous dynamical systems. *J. Symbolic Comput.*, 58:41–63, 2013.

[35] J. Sylvester. Sur l'equation en matrices px=xq. *C. R. Acad. Sci. Paris.*, 99(2):67–71, 115–116, 1884.

[36] A. Vannelli and M. Vidyasagar. Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.

[37] W. Wang and S. Ruan. Bifurcations in an epidemic model with constant removal rate of infectives. *J. Math. Anal. Appl.*, 291(1):775–793, 2004.

[38] T. Yoshizawa. *Stability theory by Liapunov's second method*. Publications of the Mathematical Society of Japan, No. 9. The Mathematical Society of Japan, Tokyo, 1966.

[39] V. I. Zubov. *Methods of A. M. Lyapunov and their application*. Translation prepared under the auspices of the United States Atomic Energy Commission; edited by Leo F. Boron. P. Noordhoff Ltd, Groningen, 1964.