Class library in C++ to compute Lyapunov functions for nonlinear systems

Jóhann Björnsson * **Skuli Gudmundsson** * **Sigurdur Hafstein** *

* *Reykjavik University, Menntavegur 1, 101 Reykjavik, Iceland*
e-mail: {johannbj, skuligu, sigurdurh}@ru.is

Abstract: Lyapunov functions are of fundamental importance in the stability analysis of dynamical systems. Unfortunately, the construction of Lyapunov functions for nonlinear systems is in general a very difficult problem. We present software written in C++ that computes Lyapunov functions for n-dimensional, nonlinear, time-continuous dynamical systems defined through systems of autonomous differential equations.
The software implements the computation of continuous and piecewise affine (CPA) Lyapunov functions through linear programming (LP) and the computation of Lyapunov functions using radial basis functions (RBF) and collocation. In the former case a common Lyapunov function for a finite set of nonlinear systems can be computed and it is guaranteed to be a true Lyapunov function, i.e. to fulfill the defining properties of a Lyapunov function exactly and rigorously (as opposed to approximately, which is so often the case with numerically contrived results). In the latter case the computed function is smooth and is guaranteed to be a Lyapunov function outside of an arbitrary small neighbourhood of the equilibrium, if the collocation points are close enough. However, it is not obvious how to determine what is close enough. A Lyapunov property test for the CPA interpolation of the computed RBF solution is therefore also part of the software.

Keywords: Lyapunov function, C++ software, stability, dynamical systems

1. INTRODUCTION

The computation of Lyapunov functions has been an active field of study for decades. Earlier, much attention was spent on linear differential inclusions Brayton and Tong (1979); Wang and Michel (1996); Polanski (2000) and this is still an active field of study Lazar and Jokić (2010); Lazar and Doban (2011). Since the turn of the millennium several different more general approaches have appeared in the literature. For polynomial vector-fields the computation of polynomial Lyapunov functions has received much attention, either using the sum-of-squares (SOS) relaxation and semi-definite programming Peet (2009); Peet and Papachristodoulou (2012), introduced in Parrilo (2000), or different methods like quantifier-elimination Burchardt and Ratschan (2007). For an overview of SOS methods we refer the reader to Anderson and Papachristodoulou (2015) and for an overview of other polynomial methods we refer the reader to Kamyar and Peet (2015). Other approaches use graph theoretic methods Kalies et al. (2005); Ban and Kalies (2006) or other diverse methods to solve the Zubov equation Zubov (1964); Vannelli and Vidyasagar (1985); Camilli et al. (2001); Grüne (2002). For a more detailed overview of different methods to compute Lyapunov functions numerically cf. the recent review Giesl and Hafstein (2015b).

There have been numerous publications on two other methods, the CPA method that uses LP to parameterize a CPA Lyapunov function and the RBF method, a collocation method that uses radial basis functions to solve a Zubov equation numerically, a first-order partial differential equation (PDE), whose solution is a Lyapunov function for the system at hand. Recently, the advantages of both these methods were combined in Giesl and Hafstein (2015a). The RBF method is used to propose a Lyapunov function and then the constraints of the LP problem of the CPA method are verified. This delivers a method that is exact, i.e. results in a true Lyapunov function and not an approximation, and has the numerical complexity of the RBF method, which is considerably lower than that of the CPA method.

In this paper we present a class library in C++ that implements both the CPA and the RBF method and the combined method from Giesl and Hafstein (2015a). In the next section we give a very brief overview of the CPA and RBF methods respectively. Then we discuss the structure of the software briefly and give five worked out examples of its use.

2. METHODS IMPLEMENTED

We give a short introduction of the methods implemented. For a more detailed description we refer the reader to Hafstein (2007); Giesl and Hafstein (2014) for the CPA method and Giesl (2007) for the RBF method. For either method we consider the following autonomous system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad \text{where } \mathbf{f} \in C^2(\mathbb{R}^n, \mathbb{R}^n) \text{ and } \mathbf{f}(\mathbf{0}) = \mathbf{0}. \qquad (1)$$

⋆ Björnsson and Gudmundsson are supported by The Icelandic Research Fund, grant nr. 130677-052 and 152429-051 respectively.

CPA method: To define a CPA function we first fix a suitable triangulation. A suitable triangulation of a subset $\mathcal{D}_\mathcal{T}$ of $\mathbb{R}^n$, breaks $\mathcal{D}_\mathcal{T}$ up into a finite collection $\mathcal{T} = (\mathfrak{S}_\nu)$ of n-simplices $\mathfrak{S}_\nu := \mathrm{co}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$, such that pairs of simplices intersect in a common face or not at all. Further, we demand that the interior $\mathcal{D}_\mathcal{T}^\circ$ of

$$\mathcal{D}_\mathcal{T} = \bigcup_{\mathfrak{S}_\nu \in \mathcal{T}} \mathfrak{S}_\nu$$

is a simply connected neighbourhood of the origin and that

$$\mathbf{0} \in \mathfrak{S}_\nu = \mathrm{co}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{T} \;\Rightarrow\; \mathbf{x}_0 = \mathbf{0}.$$

Denote by $\mathcal{V}_\mathcal{T}$ the set of all vertices of all simplices of $\mathcal{T}$.

A function $V : \mathcal{D}_\mathcal{T} \to \mathbb{R}$ is said to be a CPA function on the triangulation $\mathcal{T}$, written $V \in \mathrm{CPA}[\mathcal{T}]$, if V is continuous and affine on each simplex $\mathfrak{S}_\nu \in \mathcal{T}$. It is not difficult to see that V is uniquely determined by its values at the vertices $\mathcal{V}_\mathcal{T}$. Since V is affine on each $\mathfrak{S}_\nu \in \mathcal{T}$, there is a vector $\mathbf{w}_\nu \in \mathbb{R}^n$ and a number $a_\nu \in \mathbb{R}$, such that

$$V(\mathbf{x}) = \mathbf{w}_\nu \cdot \mathbf{x} + a_\nu, \quad \forall \mathbf{x} \in \mathfrak{S}_\nu.$$

We define $\nabla V_\nu := \mathbf{w}_\nu$. As shown in (Giesl and Hafstein, 2014, Remark 9) ∇V_ν is affine in the values $V_i = V(\mathbf{x}_i)$ of V at the vertices $\mathbf{x}_i$ of $\mathfrak{S}_\nu := \mathrm{co}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$. $V \in \mathrm{CPA}[\mathcal{T}]$ is positive definite, if and only if

$$V(\mathbf{0}) = 0 \text{ and } V(\mathbf{x}) > 0, \; \forall \mathbf{x} \in \mathcal{V}_\mathcal{T} \setminus \{\mathbf{0}\}. \tag{2}$$

The (Dini) orbital derivative

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

of $V \in \mathrm{CPA}[\mathcal{T}]$ is negative on $\mathfrak{S}_\nu = \mathrm{co}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)$, if

$$\nabla V_\nu \cdot \mathbf{f}(\mathbf{x}_i) + \|\nabla V_\nu\|_1 B_\nu h_{i,\nu} \frac{h_{i,\nu} + h_\nu}{2} < 0, \tag{3}$$

where

$$h_{i,\nu} := \|\mathbf{x}_i - \mathbf{x}_0\|_2 \;\text{ for }\; i = 1, 2, \ldots, n,$$
$$h_\nu := \max_{\mathbf{x}, \mathbf{y} \in \mathfrak{S}_\nu} \|\mathbf{x} - \mathbf{y}\|_2$$

is the diameter of $\mathfrak{S}_\nu$, and

$$B_\nu \geq \max_{\substack{i,r,s=1,2,\ldots,n \\ \mathbf{x} \in \mathfrak{S}_\nu}} \left| \frac{\partial^2 f_i}{\partial x_r \partial x_s}(\mathbf{x}) \right|$$

is an upper bound (not necessarily tight) on the second-derivatives of the components f_i of $\mathbf{f}$ in $\mathfrak{S}_\nu$. Especially, V is negative definite on $\mathcal{D}_\mathcal{T}^\circ$ if (3) holds true for all $\mathbf{x}_i \in \mathcal{V}_\mathcal{T} \setminus \{\mathbf{0}\}$.

The inequalities (2) and (3) can be treated as the linear constraints of an LP program with the values V_i of V on $\mathcal{V}_\mathcal{T} \setminus \{\mathbf{0}\}$ as variables. Because these values of V on $\mathcal{V}_\mathcal{T}$ determine $V \in \mathrm{CPA}[\mathcal{T}]$, an LP problem can be used to parameterize a CPA Lyapunov function. This is the idea of the CPA method.

Another possibility is to compute the values $V(\mathbf{x}_i)$ at the vertices $\mathbf{x}_i \in \mathcal{V}_\mathcal{T}$ by other means and then verify that the inequalities (2) and (3) hold. Obviously, the values $V_i = V(\mathbf{x}_i)$ should be computed in such a way that the corresponding $V \in \mathrm{CPA}[\mathcal{T}]$ (the CPA interpolation) has a likelihood of being Lyapunov. This has been the subject of several recent publications Hafstein et al. (2014); Björnsson et al. (2014b, 2015) using converse theorems by Massera Massera (1949) and Yoshizawa Yoshizawa (1966) and by using the RBF method Björnsson et al. (2014a); Giesl and Hafstein (2015a).

RBF method: The RBF method uses collocation to solve the Zubov equation

$$\nabla V(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) = -p(\mathbf{x}) \tag{4}$$

for V, where $\mathbf{f}$ is from the system (1) and p is a positive definite function.

First, a finite number N of collocation points $\mathcal{X} := \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \subset \mathbb{R}^n$ with $\mathbf{f}(\mathbf{y}_i) \neq \mathbf{0}$ and a basis-function ψ are fixed. We take the ψ to be a Wendland radial basis function Wendland (2005) and for $n = 2, 3$ a suitable choice is, cf. (Giesl, 2007, Table 3.1),

$$\psi_1(r) := r^{-1}\psi'(cr) = -22c^2(1-cr)_+^7[16(cr)^2 + 7cr + 1],$$
$$\psi_2(r) := r^{-1}\psi_1'(cr) = 528c^4(1-cr)_+^6[6cr + 1],$$

where $x_+ := \max(0, x)$ and $c > 0$ is a fixed parameter. These assignments for ψ_1 and ψ_2 are the default in the software.

Second, the matrix $A = (a_{jk})_{j,k=1,2,\ldots,N}$ is written. With $\mathbf{z}_{jk} := \mathbf{y}_j - \mathbf{y}_k$ and $z_{jk} := \|\mathbf{z}_{jk}\|_2$ we set

$$a_{jk} = \psi_2(z_{jk})[\mathbf{z}_{jk} \cdot \mathbf{f}(\mathbf{y}_j)][\mathbf{z}_{kj} \cdot \mathbf{f}(\mathbf{y}_k)] - \psi_1(z_{jk})[\mathbf{f}(\mathbf{y}_j) \cdot \mathbf{f}(\mathbf{y}_k)]$$

Third, the linear equation

$$A\boldsymbol{\beta} = \boldsymbol{\alpha}, \text{ where } \alpha_i = p(\mathbf{y}_i) \tag{5}$$

is solved for $\boldsymbol{\beta}$.

The function

$$V_R(\mathbf{x}) := \sum_{k=1}^N \beta_k (\mathbf{y}_k - \mathbf{x}) \cdot \mathbf{f}(\mathbf{y}_k) \psi_1(\|\mathbf{x} - \mathbf{y}_k\|_2) + V_0,$$

where the constant V_0 is chosen such that $V_R(\mathbf{0}) = 0$, is now a Lyapunov function for the system on $\mathrm{co}\,\mathcal{X}$, apart from a small neighborhood of the origin, if the fill-distance

$$h_\mathcal{X} := \max_{\mathbf{x} \in \mathrm{co}\,\mathcal{X}} \min_{k=1,2,\ldots,N} \|\mathbf{x} - \mathbf{y}_k\|_2$$

of the collocation points in $\mathcal{X}$ if small enough.

3. EXAMPLES

We present the software by five worked out examples. In all but the third example we consider the system (1) with different $\mathbf{f}$. In the third example we consider a differential inclusion. We make use of the classes *T_std_NK* and *ZGrid*, whose implementations are described in detail in Hafstein (2013). For the programming we used Visual C++ Express Desktop 2013 for Windows with heavy reliance upon the Armadillo Linear Algebra Library Sanderson (2010). To solve the LP problems the software is configured to use either Gurobi or Gnu Linear Programming Kit (GLPK). To make the figures we used Scilab and Matlab. Visual C++ Express, Armadillo, and Scilab are available at no cost and Gurobi is available at no cost for academics (registration required).

Example 1 - nonlinear planar system: The first example is the system (1) with

$$\mathbf{f}(x, y) = \begin{pmatrix} -1.5y \\ \frac{1}{1.5}x + y[\left(\frac{x}{1.5}\right)^2 + y^2 - 1] \end{pmatrix}.$$

It is simple to verify that the origin is an exponentially stable equilibrium point. To generate the LP problem we first create a simplicial complex, where the simplex (here triangle) vertices are the integer coordinates of the square $[-15, 15]^2$ outside the region $[-2, 2]^2$, where instead

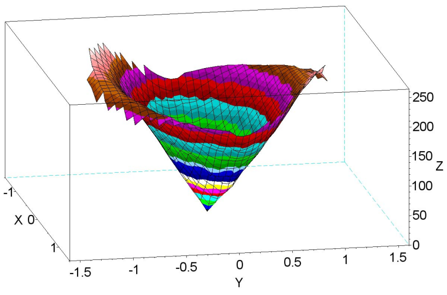

Fig. 1. Computed CPA Lyapunov function for Example 1.

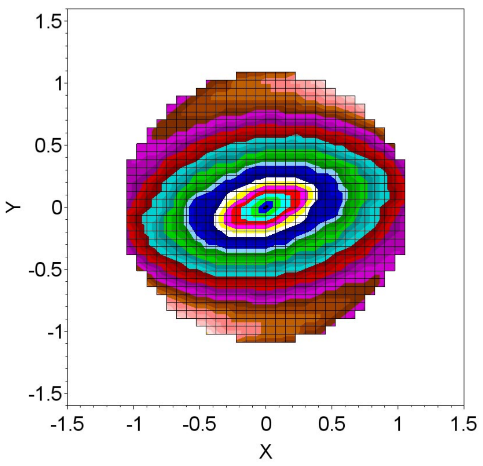

Fig. 2. Level-sets of the CPA Lyapunov function computed in Example 1.

a simplicial fan is used. A simplicial fan has the origin as one vertex and the integer coordinate points on the edge of $[-2,2]^2$ as the other vertices. Such a complex is constructed by the command

T_std_K BC(Nm, Np, Km, Kp);

where *Nm*$=(-15,-15)^T$, *Np*$=(15,15)^T$, *Km*$=(-2,-2)^T$, and *Kp*$=(2,2)^T$.

We then define the function $\mathbf{F}:\mathbb{R}^2\to\mathbb{R}^2$ by

$$\mathbf{F}(\mathbf{0}):=\mathbf{0} \text{ and } \mathbf{F}(\mathbf{x}):=0.075\cdot\frac{\|\mathbf{x}\|_\infty}{\|\mathbf{x}\|_2}\mathbf{x} \text{ for } \mathbf{x}\neq\mathbf{0}$$

and construct a new simplicial complex *SC* from the original *BC* with the command *FT SC(&BC, F)*. A simplex $\mathrm{co}(\mathbf{v}_0,\mathbf{v}_1,\mathbf{v}_3)$ in *BC* becomes $\mathrm{co}(\mathbf{F}(\mathbf{v}_0),\mathbf{F}(\mathbf{v}_1),\mathbf{F}(\mathbf{v}_3))$ in *SC*. The LP problem is now constructed by

LP_CA LP(&SC);
LP.AddSystem(new MonE23, true);
LP.MakeLPProblem( );

where the function $\mathbf{f}$ is implemented in the class *MonE23* in “DynSystem.cpp”. Additionally we need bounds on the second-order derivatives of the components of $\mathbf{f}$. This is implemented by defining the function $B:[0,\infty)^2\to[0,\infty)$ in *MonE23*, such that

$$B(x,y)\geq\max_{|x^*|\leq x,\,|y^*|\leq y}\left(\max_g|g(x^*,y^*)|\right),$$

where g runs over all second-order derivatives of the components f_1 or f_2 of $\mathbf{f}$. The only non-vanishing such second-order derivatives are

$$\frac{\partial^2 f_2}{\partial x^2}=\frac{8}{9}y,\ \frac{\partial^2 f_2}{\partial x\partial y}=\frac{\partial^2 f_2}{\partial y\partial x}=\frac{8}{9}x,\ \text{and}\ \frac{\partial^2 f_2}{\partial y^2}=6y$$

and we can thus set $B(x,y):=\max\{8/9x,6y\}$. Note that these bounds do not have to be tight, any finite upper bound will do the job!

To solve the LP problem we can use the command *LP.SolveLPwithGLPK( )* or *LP.SolveLPwithGUROBI( )*. Note that the Gurobi-solver is considerably faster. If the problem has a solution, as it does with these parameters, the above commands are completed and the solution becomes a part of the *LP* object. We set-up the CPA Lyapunov function from the solution using the command *LP.DefineV_C( )*. Files containing data in text format, required to plot the function are created with the command: *WriteMesh(“Lya1”,LP.V_C,x_min,x_max,x_steps)*

where *x_min* is the lowest-leftmost point and and *x_max* the highest-rightmost points of the domain of the function to be plotted and *x_steps* contains the resolution to be used. In Figure 1 the computed CPA Lyapunov function *LP.V_C* is depicted and in Figure 2 we plot its level-sets. This is done with the self-explanatory Scilab script in the file *plot2dmesh.sce*, so long as *Name=“Lya1”* matches the filename used by *WriteMesh*. The text-data files are in standard ascii-form and the data can therefore be accessed from other tools with a minimal effort.

Example 2 - systematic complex refinement: Here we consider the system (1) with

$$\mathbf{f}(x,y)=\begin{pmatrix}x^3(y-1)\\ -\dfrac{x^4}{(1+x^2)^2}-\dfrac{y}{1+y^2}\end{pmatrix}.$$

This system is taken from (Hafstein, 2007, Sec. 9.1) and an appropriate $B(x,y)$ and the fact that the origin is an asymptotically rather than exponentially stable equilibrium, is derived there. Hence, the system cannot possess a CPA Lyapunov function in a neighbourhood of the origin. We can, however, compute a CPA Lyapunov function if we exclude an arbitrary small neighbourhood of the origin.

Here we search iteratively for a simplicial complex such that the LP problem has a solution. More exactly, for $K=1,2,\ldots,10$ we set $N=15K$ and construct the initial simplicial complex by *T_std_NK BC(Nm, Np)*, where *Nm*$=(-N,-N)^T$ and *Np*$=(N,N)^T$. Notice that the command *T_std_NK* accepts the syntax shown here and results in a simplicial complex without the simplicial fan from before. We then construct a new simplicial complex with the command *FT SC(&BC, F)*, but now with $\mathbf{F}(\mathbf{x})=2\|\mathbf{x}\|_\infty/(N\|\mathbf{x}\|_2)\,\mathbf{x}$. We next load the system into the *LP* object with *LP.AddSystem(new MonE1,true)*. We can use the command *ZGrid Excl(Km, Kp)*, where *Km*$=(-K,-K)^T$ and *Kp*$=(K,K)^T$, to specify simplices in *BC* which we want to exclude from the LP problem. We construct the LP problem with these exclusions using the command *LP.MakeLPProblem(Excl)*. The exact functionality is that a simplex in *BC* is excluded from the constraints of the LP problem, if all of its vertices belong to $\{-K,-K+1,\ldots,K\}^2$.

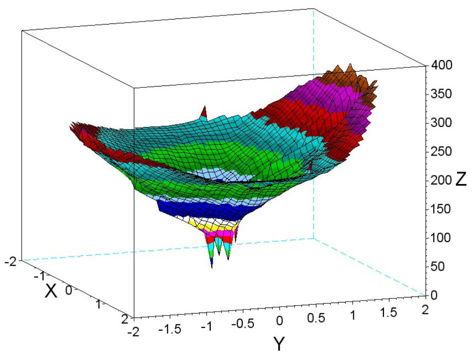

Fig. 3. Computed CPA Lyapunov function for Example 2.

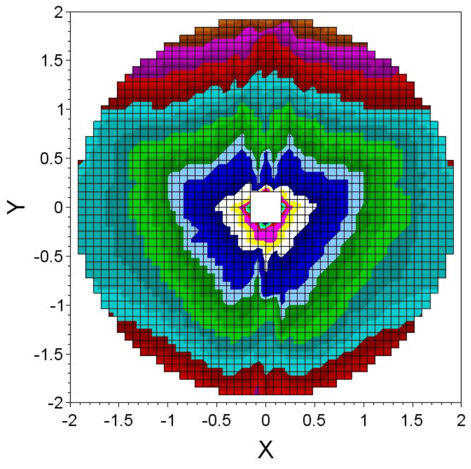

Fig. 4. Level-sets of the CPA Lyapunov function computed in Example 2.

When a solution to the LP problem is found, which should happen with $K = 3$, $N = 45$, the CPA Lyapunov function is defined by *LP.DefineV_C(Excl)*, where we have specified that the simplices excluded in the LP problem are not to be used in the definition of *LP.V_C*. In Figure 3 the computed Lyapunov function *LP.V_C* is depicted and in Figure 4 we plot its level-sets. Note that a neighbourhood of the origin is excluded from its definition.

Example 3 - differential inclusion: The third example is a differential inclusion from (Hafstein, 2007, Sec. 9.2),

$$\dot{\mathbf{x}} \in \text{co}\{\mathbf{f}_1(\mathbf{x}), \mathbf{f}_2(\mathbf{x}), \mathbf{f}_3(\mathbf{x})\}, \tag{6}$$

where

$$\mathbf{f}_1(x,y) = \begin{pmatrix} -y \\ x - y(1 - x^2 + 0.1x^4) \end{pmatrix}$$

$$\mathbf{f}_2(x,y) = \begin{pmatrix} -y + x(x^2+y^2-1) \\ x + y(x^2+y^2-1) \end{pmatrix},$$

and $\mathbf{f}_3$ is the $\mathbf{f}$ from Example 1.

To compute a CPA Lyapunov function for the differential inclusion one proceeds just as in Example 1 to construct the simplicial complex, but now one must add to the LP problem the entire collection of systems corresponding to $\dot{\mathbf{x}} = \mathbf{f}_1(\mathbf{x})$, $\dot{\mathbf{x}} = \mathbf{f}_2(\mathbf{x})$, and $\dot{\mathbf{x}} = \mathbf{f}_3(\mathbf{x})$ as shown below. Indeed, the computed CPA Lyapunov function will be a common Lyapunov function for these three systems. This is achieved by creating the $\mathbf{f}$'s and appropriate B's as part of the Dynamic-system classes *MonE21*, *MonE22*, and *MonE23* for the three systems in question. The explicit commands are: *LP.AddSystem(new MonE21, true)*, *LP.AddSystem(new MonE22, true)*, and *LP.AddSystem(new MonE23, true);* to add the systems to the LP object. The setting up of the problem is finalized with *LP.MakeLPProblem()*. Then one proceeds again as in Example 1. In Figure 5 the computed CPA Lyapunov function *LP.V_C* is depicted and in Figure 6 we plot its level-sets.

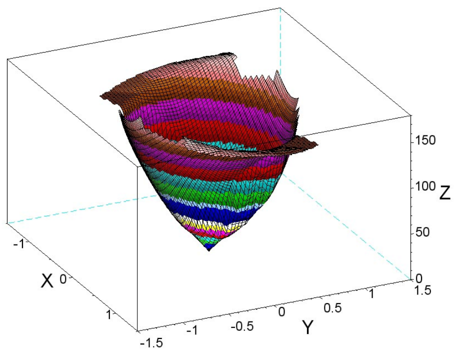

Fig. 5. Computed CPA Lyapunov function for Example 3.

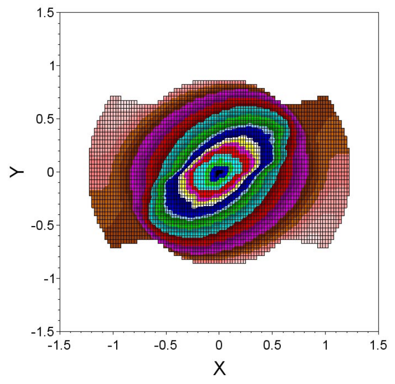

Fig. 6. Level-sets of the CPA Lyapunov function computed in Example 3.

Example 4 - The RBF method and CPA verification: We consider the computation of a Lyapunov function with the RBF method on a domain where the system has three equilibria, an exponentially stable one at the origin $(0,0)$ and two saddle points at $(0, \pm\sqrt{3})$. The system is (1) with

$$\mathbf{f}(x,y) = \begin{pmatrix} y \\ -x + \dfrac{1}{3}x^3 - y \end{pmatrix}.$$

First, the system is defined by defining $\mathbf{f}$ in the dynamic-system class *SE2*. For the RBF problem we do not need to define $B(x,y)$, but for the subsequent CPA verification we do. Straight-forward calculations show that we can set

$$2x =: B(x,y) \geq \max_{|x^*| \leq x, |y^*| \leq y} \left| \frac{\partial^2 f_2}{\partial x^2}(x^*, y^*) \right|$$

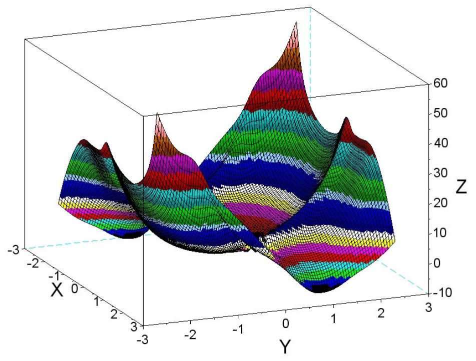

Fig. 7. Computed RBF Lyapunov function for Example 4.

because all other second-order derivatives vanish.

Second, we construct our RBF problem as an object of class *RBF*. The equation we are solving is:

$$V'(\mathbf{x}) := \nabla V(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) = -\|\mathbf{x}\|_2^2(1 + \|\mathbf{f}(\mathbf{x})\|_2^2). \quad (7)$$

and the arguments of the constructor *RBF V_R(·,·)* are an instance of the system *SE2* and the function on the right-hand side of (7). To determine the collocation points we first define the characteristic function *bool Domain(vec)* of the area $\{(x,y) \in \mathbb{R}^2 : \max(|x|,|y|) < 2.6\}$, where we want to put them. Then we add the points with the command *V_R.AddVertices(HexaGrid(ZGrid(Nm,Np),c_R,Domain))*, where the *ZGrid* should be large enough so that all potential points generated by *Hexagrid* are in the domain specified, *c_R* is a parameter fixing the fill distance of the grid, and *Hexagrid* generates the optimal grid for the RBF problem with the specified fill distance, cf. (Giesl, 2007, p. 133-134). The command *V_R.SolveRBF(c)* then solves the RBF problem using radial-basis functions with support of radius c^{-1}. The default is to use the Wendland functions $\psi_{5,3}$, cf. (Giesl, 2007, p. 74).

We now construct the LP problem similar to Example 1, but instead of solving the problem we read the values from the solution to the RBF problem with the command *LP.ReadSolution(V_R)*. With this command we assign the values of *V_R* to the CPA function *LP.V_C* at all vertices of the simplicial complex used by the LP problem *LP_CA LP*. We reefer to *LP.V_C* as the CPA interpolation of the function *V_R*. To examine where the orbital derivative of the CPA interpolation of *V_R* is negative, we use the command *LP.WriteOrbDerNN("NN4")*, which writes the coordinates of the midpoints of the simplices where the orbital derivative of *LP.V_C* is nonnegative in the file *NN4.txt*. We refer to this as a CPA verification. In the simplices, of which the midpoints are written to *"NN4.txt"* the CPA function *LP.V_C*, fails the condition for a Lyapunov function.

The function *V_R* computed by the RBF method is depicted in Figure 7. On Figure 8 we plot its level-sets and make black dots where the orbital derivative of its CPA interpolation is nonnegative. As suggested by the theory this happens close to all equilibria.

Example 5 - RBF method in 3D: The last example is similar to Example 4, but for a 3-dimensional system

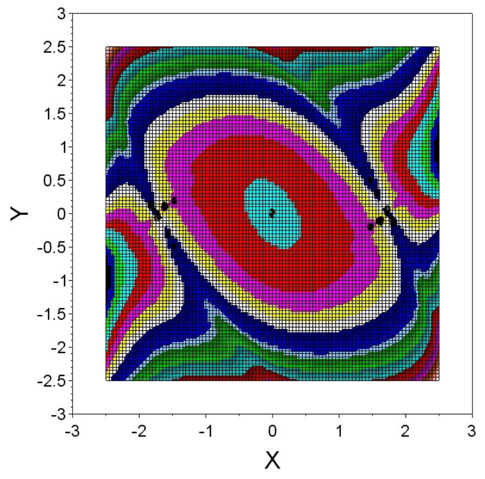

Fig. 8. Level-sets of the RBF Lyapunov function computed for Example 4. We plot black dots where its CPA interpolation fails to have a negative orbital derivative.

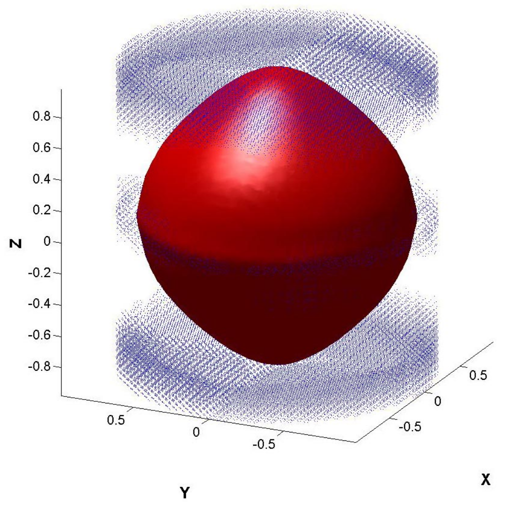

Fig. 9. Level-set of the RBF Lyapunov function computed for Example 5. We plot blue dots where its CPA interpolation fails to have a negative orbital derivative.

from from (Giesl, 2007, Ex. 6.4). We consider the system (1) with

$$\mathbf{f}(x,y,z) = \begin{pmatrix} x(x^2+y^2-1) - y(z^2+1) \\ y(x^2+y^2-1) + x(z^2+1) \\ 10z(z^2-1) \end{pmatrix}.$$

It is simple to verify that we can take

$$B(x,y,z) = 6\max\{x, y, 10z\}.$$

4. CONCLUSION

We gave a short description of the CPA and the RBF methods to compute Lyapunov functions. We outlined a C++ class library that applies these methods to compute Lyapunov functions for nonlinear systems. To present the capacities of the software, available freely at *www.ru.is/kennarar/sigurdurh/MICNON2015CPP.rar*, we thoroughly worked out five examples, where the class library is used to compute Lyapunov functions for nonlinear systems. It is the hope of the authors, that the interested reader is able to compute a Lyapunov function for a system of her/his interest by adapting the instructions in the worked out examples to her/his specific system.

REFERENCES

Anderson, J. and Papachristodoulou, A. (2015). Advances in computational Lyapunov analysis using sum-of-squares programming. *Discrete Contin. Dyn. Syst. Ser. B.* Accepted.

Ban, H. and Kalies, W. (2006). A computational approach to Conleys decomposition theorem. *J. Comput. Nonlinear Dynam.*, 1, 312–319.

Björnsson, J., Giesl, P., and Hafstein, S. (2014a). Algorithmic verification of approximations to complete Lyapunov functions. In *Proceedings of the 21st International Symposium on Mathematical Theory of Networks and Systems*, 1181–1188 (no. 0180). Groningen, The Netherlands.

Björnsson, J., Giesl, P., Hafstein, S., Kellett, C., and Li, H. (2014b). 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.

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

Brayton, R. and Tong, C. (1979). Stability of dynamical systems: A constructive approach. *IEEE Trans. Circuits and Systems*, 26(4), 224–234.

Burchardt, H. and Ratschan, S. (2007). Estimating the region of attraction of ordinary differential equations by quantified constraint solving. In *Proceedings Of The 3rd WSEAS International Conference On Dynamical Systems And Control*, 241–246.

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

Giesl, P. (2007). *Construction of Global Lyapunov Functions Using Radial Basis Functions*, volume 1904 of *Lecture Notes in Mathematics.* Springer-Verlag, Berlin.

Giesl, P. and Hafstein (2015a). Computation and verification of lyapunov functions. *submitted.*

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

Giesl, P. and Hafstein, S. (2015b). Review of computational methods for Lyapunov functions. *Discrete Contin. Dyn. Syst.-Series B.* Accepted.

Grüne, L. (2002). *Asymptotic behavior of dynamical and control systems under perturbation and discretization*, volume 1783 of *Lecture Notes in Mathematics.* Springer-Verlag, Berlin.

Hafstein, S. (2007). *An algorithm for constructing Lyapunov functions.* Monograph. Electron. J. Diff. Eqns.

Hafstein, S. (2013). Implementation of simplicial complexes for CPA functions in C++11 using the armadillo linear algebra library. In *Proceedings of SIMULTECH*, 49–57. Reykjavik, Iceland.

Hafstein, S., Kellett, C., and Li, H. (2014). Continuous and piecewise affine Lyapunov functions using the Yoshizawa construction. In *Proceedings of the 2014 American Control Conference*, 548–553 (no. 0170). Portland (OR), USA.

Kalies, W., Mischaikow, K., and VanderVorst, R. (2005). An algorithmic approach to chain recurrence. *Found. Comput. Math*, 5(4), 409–449.

Kamyar, R. and Peet, M. (2015). Polynomial optimization with applications to stability analysis and control – an alternative to sum of squares. *Discrete Contin. Dyn. Syst. Ser. B.* Accepted.

Lazar, M. and Doban, A. (2011). On infinity norms as Lyapunov functions for continuous-time dynamical systems. In *Proceedings of the 50th IEEE Conference on Decision and Control*, 7567–7572. Orlando (Florida), USA.

Lazar, M. and Jokić, A. (2010). On infinity norms as Lyapunov functions for piecewise affine systems. In *Proceedings of the Hybrid Systems: Computation and Control conference*, 131–141. Stockholm, Sweden.

Massera, J. (1949). On Liapounoff's conditions of stability. *Ann. of Math.*, 50(2), 705–721.

Parrilo, P. (2000). *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimiziation.* PhD thesis: California Institute of Technology Pasadena, California.

Peet, M. (2009). Exponentially stable nonlinear systems have polynomial Lyapunov functions on bounded regions. *IEEE Transactions on Automatic Control*, 54(5), 979 – 987.

Peet, M. and Papachristodoulou, A. (2012). A converse sum of squares Lyapunov result with a degree bound. *IEEE Transactions on Automatic Control*, 57(9), 2281–2293.

Polanski, A. (2000). On absolute stability analysis by polyhedral Lyapunov functions. *Automatica*, 36, 573–578.

Sanderson, C. (2010). Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA.

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

Wang, K. and Michel, A. (1996). On the stability of a family of nonlinear time-varying system. *IEEE Trans. Circuits and Systems*, 43(7), 517–531.

Wendland, H. (2005). *Scattered data approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, Cambridge.

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

Zubov, V.I. (1964). *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.