Lyapunov Function Verification: MATLAB Implementation

Skuli Gudmundsson * **Sigurdur F. Hafstein** *

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

Abstract: Lyapunov functions are a mathematical generalization of the dissipative energy concept of physics. Lyapunov functions are the centerpiece of the Lyapunov-stability theory for dynamical systems in general. Here we present a simple method for checking the validity of a quadratic Lyapunov function, which is constructed for the linearization of a nonlinear system and does not in general satisfy the condition of having a negative orbital derivative on the origins basin of attraction. The current work also extends previous work done on continuous and piecewise affine (CPA) Lyapunov functions by permitting more general triangulations than have been used in that context, namely Delaunay triangulations. Delaunay triangulations have been studied intensively in the literature and allow local refinements of the triangulation. The third contribution of this paper is a freely available MATLAB implementation of the methods proposed.

Keywords: Lyapunov function, MATLAB, dynamical systems, Delaunay-triangulation.

1. INTRODUCTION

This article is the result of work which was initially undertaken as continuation of the CPA methods to compute Lyapunov functions, where a dynamical system arising from a system of nonlinear autonomous differential equations is considered and the stability of its equilibria investigated by constructing an associated CPA Lyapunov function by linear programming Marinósson (2002); Giesl and Hafstein (2014). The CPA method has already been modified to make a principled guess of appropriate values of a CPA Lyapunov function at the vertices of a suitable triangulation and then verify the conditions for a Lyapunov function of the CPA interpolation Björnsson et al. (2014,?).

Here, however, we do not consider the CPA interpolation of a candidate Lyapunov function, but verify the conditions for a Lyapunov function directly for the function at hand. The approach investigated is based on a triangulation scheme novel to the CPA method, namely the Delaunay triangulation Delaunay (1934), which has been studied at length in the literature. The Delaunay triangulation has the advantage over other triangulations that have been proposed in the context of the CPA method, in that it allows for a simple implementation of local refinements, where finer triangulations are obtained incrementally by increasing the resolution locally where it is required. We implement a Delaunay triangulation on the domain of the system and test the negativity of the orbital derivative of a quadratic Lyapunov function for the system. This quadratic Lyapunov function is constructed using standard methods of linearization of the system around the equilibrium at the origin. The MATLAB implementation and its use for the proposed method is described in detail.

1.1 Lyapunov Functions

A basic dynamical system is considered, derived from the following autonomous (and generally nonlinear) differential equation:

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

The stability of the equilibrium at the origin is to be investigated. In particular, we deliver an estimate of its basin of attraction.

The stability of (1) is closely tied to the existence of a Lyapunov function for the system, the sublevel-sets of which are entirely contained within the domain of attraction of the equilibrium at the origin. A locally Lipschitz function $V : \mathbb{R}^n \to \mathbb{R}_+$ with $V(\mathbf{0}) = 0$ and $V(\mathbf{x}) > 0$ for $\mathbf{x} \neq \mathbf{0}$ is a Lyapunov function for the system (1) if:

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

for all $\mathbf{x} \in \mathcal{U} \setminus \{\mathbf{0}\}$, where $\mathcal{U}$ is a simply connected neighborhood of the origin. If V is differentiable, then $V'(\mathbf{x}) = \boldsymbol{\nabla} V(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$. The function V' is called the *orbital derivative* of V (w.r.t. the system (1) if necessary). The inequality (2) implies that Lyapunov functions are decreasing along solution trajectories.

Consider a linear system of the form:

$$\dot{\mathbf{x}} = A\mathbf{x} \quad \text{where } A \in \mathbb{R}^{n \times n} \tag{3}$$

and let $Q \in \mathbb{R}^{n \times n}$ be an arbitrary positive definite matrix. The origin is an asymptotically stable equilibrium of system (3), if and only if the equation:

$$P\,A + A^T\,P + Q = 0 \tag{4}$$

⋆ Gudmundsson is supported by The Icelandic Research Fund, grant nr. 152429-051.

(called emphcontinuous-time Lyapunov equation) has a positive definite solution P. The solution P is then unique and

$$V_P(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T P\,\mathbf{x} \tag{5}$$

is a Lyapunov function for the system (3) with $\mathcal{U} = \mathbb{R}^n$.

For the general system (1) one can consider its linear approximation (3) with $A = J_{\mathbf{f}}(\mathbf{0})$ the Jacobi-matrix of $\mathbf{f}$ at the origin. It is well known that if (5) is a Lyapunov function for the linearized system, then it is also a valid Lyapunov function for the (nonlinear) system (1) in a small enough neighborhood $\mathcal{U}$ of the origin. In Hafstein (2004) it is e.g. shown that one can take $\mathcal{U} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty < a\}$, where,

$$\frac{1}{a} > \|P\|_2 \sqrt{\sum_{i=1}^{n} \left(\sum_{j,k=1}^{n} \sup_{\boldsymbol{\xi} \in [-a,a]^n} \left| \partial^2_{jk} f_i(\boldsymbol{\xi}) \right| \right)^2}, \tag{6}$$

and $\|P\|_2$ is the spectral-norm of the matrix P. This can be used to get a minimal region. However, the domain where the orbital derivative is negative might be much larger than obtained by this crude estimate.

2. NUMERICAL APPROACH

Because of the fundamental difficulty in finding a Lyapunov function for a given dynamical systems, we are drawn to approaching the problem numerically. There have been many proposals in the literature on how to generate Lyapunov functions computationally, e.g. by parameterizing sum-of-squared (SOS) polynomials Parrilo (2000); Peet and Papachristodoulou (2012), see also the MATLAB toolbox Papachristodoulou et al. (2013) and the review Anderson and Papachristodoulou (2015); using collocation to solve the Zubov equation numerically Giesl (2007) or other methods for computing polynomial Lyapunov functions Kamyar and Peet (2015). A review article has recently been published Giesl and Hafstein on the subject in general.

In the present article we will make extensive use of methods presented initially in Marinósson (2002) to parameterize continuous and piecewise affine (CPA) Lyapunov functions on a simplicial-triangulation of $\mathcal{U} \subset \mathbb{R}^n$. Further work on the CPA method has been undertaken in e.g. Hafstein (2007); Baier et al. (2012); Giesl and Hafstein (2013, 2014).

2.1 Simplicial Triangulations

For a finite set of vectors $C = \{\mathbf{x}_0, \ldots, \mathbf{x}_m\}$ the set consisting of all their convex-combinations is called the convex-hull of C and is denoted by:

$$\operatorname{co} C = \left\{ \sum_{k=0}^{m} \lambda_k \mathbf{x}_k \,:\, 0 \le \lambda_k \le 1, \sum_{k=0}^{m} \lambda_k = 1 \right\}.$$

The set $S = \operatorname{co} C = \operatorname{co}\{\mathbf{x}_0, \ldots, \mathbf{x}_m\}$ is called an m-simplex if the vectors $\mathbf{x}_k - \mathbf{x}_0$ are linearly-independent and the vectors $\mathbf{x}_0, \ldots, \mathbf{x}_m$ are then referred to as its vertices. It is easy to see that this does not depend upon the choice of $\mathbf{x}_0$ among the vertices. We are mainly interested in n-simplices in $\mathbb{R}^n$.

The *triangulation* $\mathcal{T}$ of a set $\mathcal{U} \subset \mathbb{R}^n$, is a family of n-simplices $(S_\nu)_{\nu \in I} = (\operatorname{co} C_\nu)_{\nu \in I}$ where $C_\mu = \{\mathbf{x}_0^\mu, \mathbf{x}_1^\mu, \ldots, \mathbf{x}_n^\mu\}$ such that

$$\mathcal{D}_{\mathcal{T}} := \bigcup_{\nu \in I} S_\nu = \mathcal{U}$$

and for which the pairwise intersection of simplices in $\mathcal{T}$ is a common face of the simplices or empty, i.e.

$$S_\nu \cap S_\mu = \operatorname{co} C_\mu \cap \operatorname{co} C_\nu = \operatorname{co}(C_\mu \cap C_\nu).$$

A continuous function $g : \mathcal{D}_{\mathcal{T}} \to \mathbb{R}$ that is affine on each simplex $S_\nu \in \mathcal{T}$ can now be uniquely defined by specifying its values at the vertices $\bigcup_{\nu \in I} C_\nu$. This fact is instrumental for the CPA method.

Analogous to using *Cartesian-*, *spherical-* or *cylindrical-coordinates* to indicate points in a set $\mathcal{U} \subset R^n$, there are so-called *Barycentric-coordinates* which can be used on a set $\mathcal{U} \subset \mathbb{R}^n$ which has a triangulation $\mathcal{T} = (S_\nu)_{\nu \in I}$. With Barycentric-coordinates one specifies a point $\mathbf{x} \in \mathcal{U}$ by first listing a simplex which contains the point $\mathbf{x} \in S_\mu = \operatorname{co}\{\mathbf{x}_0^\mu, \ldots, \mathbf{x}_n^\mu\}$ and with the simplex index μ given, listing the unique convex-combination which yields the point $\mathbf{x} = \sum_{k=0}^{n} \lambda_k^\mu \mathbf{x}_k^\mu$. A Barycentric-coordinate transformation is thus given by

$$\mathbf{x} = (x_1, \ldots, x_n)^T \rightleftarrows (\mu, \lambda_0, \lambda_1, \ldots, \lambda_n)$$

with $\mu \in I$ from the index set I of the family $\mathcal{T}$ and the λ_k-s satisfying: $0 \le \lambda_k \le 1$ and $\sum_{k=0}^{n} \lambda_k = 1$.

We have discussed triangulations in general but let us now consider a few explicit ways of achieving an actual triangulation of a set $\mathcal{U} \subset \mathbb{R}^n$. A so-called *standard* triangulation $\mathcal{T}_{\text{std}}$ was proposed in Giesl and Hafstein (2012b) which systematically triangulated $[0,1]^n$ with sums of unit vector as vertices and then repeated that construction, thereby extending the triangulation to an integer grid (a subset of $\mathbb{Z}^n$). This construction could then simply be scaled to fit into a given dynamical system domain. In Giesl and Hafstein (2014) this method was extended to using a fan-like triangulation at the origin. Another possibility that has been discussed is to map the vertices of $\mathcal{T}_{\text{std}}$ with an invertible transformation $\mathbb{R}^n \to \mathbb{R}^n$ of the form:

$$\mathbf{x} \mapsto \frac{\|\mathbf{x}\|_\infty}{\|\mathbf{x}\|_Q} \mathbf{x}$$

where Q is a positive definite matrix and $\|\mathbf{x}\|_Q = \sqrt{\mathbf{x}^T Q \mathbf{x}}$. The simplices of the resulting triangulation $\mathcal{T}_Q$ would now be the convex-hulls of the mapped vertices.

Although the triangulations $\mathcal{T}_{\text{std}}$ with a triangular fan at the origin and $\mathcal{T}_Q$ served well for some applications that have been investigated, there are some inherit difficulties with such very regular and strictly constructed triangulations. For example, it is difficult to come up with a systematic way of increasing vertex-resolution locally. Furthermore, it is difficult to continue with a development of methods which rely upon a triangulation scheme which is outside the domain of common usage. A completely different triangulation method, called *Delaunay-triangulations* Delaunay (1934) has been studied, developed and used extensively within the field of computational geometry, and it would be of great utility for future work in the field of CPA Lyapunov functions to apply this less restricted and widely used algorithms to create triangulations for domains of dynamical systems. There have been numerous applications for Delaunay triangulations, not least in the fields of

digital graphics and rendering, and for this reason many useful properties concerning Delaunay triangulations have been proven, but most importantly, very efficient incremental algorithms have been designed for the creation of Delaunay triangulations of a generically scattered set of points in $\mathbb{R}^n$ Guibas et al. (1992). For this purpose, let us review briefly what a Delaunay-triangulation is and then let us use this scheme in the present context of Lyapunov function generation and testing.

For $n+1$ points defining an n-simplex in $\mathbb{R}^n$ the n-sphere with all vertices on its surface is called the *circum-sphere* for the simplex and its center, the *simplex's circum-center*. A *Delaunay-triangulation* of a set of points $\mathcal{P} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \subset \mathbb{R}^n$ is a triangulation $\mathcal{T}_D = (S_\nu^D)_{\nu \in I}$ where the vertices of each *Delaunay-simplex* S_ν^D come from the set $\mathcal{P}$ and S_ν^D is such that the interior of its circum-sphere contains none of the points in $\mathcal{P}$. Delaunay-triangulations are widely used and algorithms for their creation are therefore available as part of many calculational tools. The MATLAB Computational Geometry Toolbox is no exception. They are however quite computationally expensive and the generation quickly becomes intractable as the dimension n increases (the number of simplices in a Delaunay-triangulation is $\mathcal{O}(N^{n/2})$ Seidel (1995)). However, this is to some extent made up for by the fact that Delaunay-triangulation algorithms are especially suited for sequential triangulations. By sequential or incremental algorithms we mean such where points can be added iteratively to an already triangulated point set and a new triangulation generated by local only alterations near the added point effectively retaining a small problem in each iteration. Such behavior is ideal for the purpose of our present work, namely the local refinement of a triangulation.

2.2 Condition for Negative Orbital Derivative

With $V_P(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T P \mathbf{x}$ as before, we have

$$V_P'(\mathbf{x}) = \mathbf{x}^T P \mathbf{f}(\mathbf{x}) = \sum_{i,j=1}^{n} x_i p_{ij} f_j(\mathbf{x}), \tag{7}$$

with x_i, f_j, and p_{ij} the entries of the vector $\mathbf{x}$, the function $\mathbf{f}$, and the matrix P respectively. Let $S_\nu = \mathrm{co}\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be an n-simplex and $g \in C^2(S_\nu, \mathbb{R})$. Then, for every convex combination $\mathbf{x} = \sum_{i=0}^n \lambda_i \mathbf{x}_i$ we have the following inequality:

$$\left| g(\mathbf{x}) - \sum_{i=0}^{n} \lambda_i g(\mathbf{x}_i) \right| \leq \sum_{i=0}^{n} \lambda_i E_{i,\nu}^g$$

where

$$\begin{aligned} E_{i,\nu}^g &= n B_\nu^g h_\nu^2, \\ h_\nu &= \max_{i,j \in J_0} \|\mathbf{x}_j - \mathbf{x}_i\|_2, \\ B_\nu^g &\geq \max_{r,s \in J} \sup_{\mathbf{x} \in S_\nu} \left| \partial_{rs}^2 g(\mathbf{x}) \right|, \end{aligned}$$

where we use the index sets $J = \{1, 2, \ldots, n\}$ and $J_0 = J \cup \{0\}$. For a proof of this fact cf. e.g. (Baier et al., 2012, Prop. 4.1).

By applying this theorem to the function $V'(\mathbf{x}) = \mathbf{x}^T P \mathbf{f}(\mathbf{x})$ on a simplex S_ν of a simplicial triangulation we have the following result: If for all $i = 0, 1, \ldots, n$ we have

$$\mathbf{x}_i^T P \mathbf{f}(\mathbf{x}_i) + n B_\nu h_\nu^2 < 0, \tag{8}$$

with

$$B_\nu \geq \max_{r,s \in J} \sup_{\mathbf{x} \in S_\nu} \left| \partial_{rs}^2 V'(\mathbf{x}) \right|, \tag{9}$$

then $V(\mathbf{x})$ has a negative orbital derivative on S_ν. Just note that for a convex combination $\mathbf{x} = \sum_{k=0}^n \lambda_k \mathbf{x}_i$ we have

$$\begin{aligned} V_P'(\mathbf{x}) &\leq \sum_{k=0}^{n} \lambda_k V_P'(\mathbf{x}_k) + \left| V_P'(\mathbf{x}) - \sum_{k=0}^{n} \lambda_k V_P'(\mathbf{x}_k) \right| \\ &\leq \sum_{k=0}^{n} \lambda_k \left(\mathbf{x}_k^T P \mathbf{f}(\mathbf{x}_k) + n B_\nu h_\nu^2 \right) \\ &< 0. \end{aligned}$$

The second derivatives of the orbital derivative (7) is easily calculated:

$$\begin{aligned} \partial_{rs}^2 V'(\mathbf{x}) = &\sum_{j=1}^{n} \left(p_{sj} \partial_r f_j(\mathbf{x}) + p_{rj} \partial_s f_j(\mathbf{x}) \right) \\ &+ \sum_{i,j=1}^{n} x_i p_{ij} \partial_{rs}^2 f_j(\mathbf{x}) \end{aligned}$$

so a sufficiently large B_ν is, e.g., given by the simple formula:

$$B_\nu = n |P|_{\max} \left(2 \max_{\substack{i,j \in J \\ \mathbf{x} \in S_\nu}} |\partial_i f_j(\mathbf{x})| + n x_\nu^{\max} \max_{\substack{i,j,k \in J \\ \mathbf{x} \in S_\nu}} \left| \partial_{ij}^2 f_k(\mathbf{x}) \right| \right)$$

where

$$|P|_{\max} = \max_{i,j \in J} |p_{ij}| \quad \text{and} \quad x_\nu^{\max} := \max_{\mathbf{x} \in S_\nu} \|\mathbf{x}\|_\infty.$$

This quantity B_ν depends on the system $\mathbf{f}$ and on the matrix P and by rewriting the condition (8) for a negative orbital derivative of $V(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T P \mathbf{x}$, we see that V' is negative on S_ν if

$$B_\nu < \min_{i \in J_0} \left[\frac{-\mathbf{x}_i^T P \mathbf{f}(\mathbf{x}_i)}{h_\nu^2} \right] =: B_\nu^\dagger. \tag{10}$$

3. LYAPUNOV VERIFICATION

Lots of work has been done in the area of Lyapunov function creation and verification of a CPA construction on a simplicial-triangulation as has been described. However, to date, this has not been done on a Delaunay triangulation to the knowledge of the present authors. In this paper, we have extracted a Lyapunov-condition specifically designed for a quadratic Lyapunov function, employed on a simplicial-triangulation in the anticipation of work along the lines of the Linear Programming approach of Giesl and Hafstein (2014, 2012a). Since this further work is out of the context of this short paper, we have decided to utilize the condition for a negative orbital derivative found in equation (10) independently. By implementing the associated Delaunay triangulation in this context, we have found a scheme which lends itself very naturally to local and iterative resolution refinements. The CPA construction on a Delaunay triangulation, which has been partially set up here, leads very naturally to the optimization approach cited, but we shall confine the current discussion to a simple MATLAB implementation of a Lyapunov-verification of a quadratic Lyapunov function obtained by linearization.

Below is an example of how to utilize the Lyapunov function condition to verify in what domain a quadratic Lyapunov function for a given system indeed has a negative

orbital derivative. Because we are refraining from extending the CPA construction on the Delaunay lattice and limiting the present study to testing only the negativity of the orbital derivative of a quadratic Lyapunov function for an exponentially stable equilibrium, it is easy to point out more straight forward ways to perform such a test, lets say for example to sample V' on a fine grid over the domain of the system. Such an approach would however remain heuristic because it would strictly be possible that negative values of V' were missed between sample points. The present method gives a mathematical guarantee as well as yields an explicit result in terms of how "fine" such a sample grid must be.

The numerical implementation below is in MATLAB and interested readers can confirm our results, or by using our code as a template, can experiment with their own dynamical systems and/or Lyapunov ansatz functions, with only the most basic MATLAB literacy. In fact, the MATLAB code has been organized in such a way as to make it particularly simple and easy to supply a different dynamical system for the calculation in the examples below. The implementation in the form of `m-files` can be found in its entirety here: http://www.ru.is/kennarar/sigurdurh/M2015M.rar.

4. ALGORITHM PROPOSED

Let us review the implementation first very schematically. The algorithm we use to verify the negativity of the orbital derivative of a quadratic Lyapunov function is as follows:

(1) Specify the dynamical system and the cube $\mathcal{D} = [-C, C]^n$ on which we want to study the negativity of the orbital derivative of our quadratic Lyapunov function candidate.
(2) Specify a negative definite matrix $Q \in \mathbb{R}^{n\times n}$ and obtain the quadratic Lyapunov function $V_P(\mathbf{x})$ from equation (5) by solving the Lyapunov equation (4).
(3) Obtain an estimate for B_ν (recall that this bound does not have to be tight, can in fact be a constant over the domain). The more generous the bound, the finer the required triangulation (smaller h_ν).
(4) Compute a (small) cube $[-a, a]^n$, for example by using (6), such that the orbital derivative $V'(\mathbf{x})$ of $V(\mathbf{x})$ is negative on $[-a, a]^n \setminus \{\mathbf{0}\}$.
(5) Generate points $\mathcal{P} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \subset \mathcal{D}$ randomly.
(6) Generate a Delaunay triangulation of the point set $\mathcal{P}$ and calculate the values of $\mathbf{f}$ and V_P on the triangulation vertices $\mathcal{P}$.
(7) Traverse the entire collection of simplices and evaluate the condition (9) for each simplex. For each simplex $S_\nu = \mathrm{co}\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$ where the condition (9) fails, we add the midpoint

$$\mathbf{y} = \frac{\mathbf{x}_0 + \mathbf{x}_1 + \ldots + \mathbf{x}_n}{n+1}$$

to $\mathcal{P}$.
(8) Return back to Step 6) with the enlarged set $\mathcal{P}$.

We do not specify a termination condition for the algorithm in general. Some obvious possibilities would be a fixed number of repetitions of Step 6), some given maximum size of $\mathcal{P}$, or some minimum of points added to $\mathcal{P}$ in Step 7).

4.1 Worked out Example

Let us now go through the above steps in detail for a concrete example:

(1) *Dynamical System* The implementation starts by specifying the dynamical system in question. As a simple explicit example, we consider a dynamical system of the form (1) with:

$$\mathbf{r} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \mathbf{f}(x, y) = \begin{pmatrix} x \\ 0.5(1 - x^2)y - x \end{pmatrix}.$$

This is the well known Van-Der-Pol oscillator converted from a second order ODE to planar system and with the time variable reversed in order to make the equilibrium at the origin exponentially stable. Lets look at the main results from our conditions applied to this case.

In MATLAB this is done using the Symbolic Math Toolbox as follows:

```
syms x y mu;
assume(x, 'real'); assume(y, 'real');
Fx(x,y,mu) = -y;
Fy(x,y,mu) = -1/2*(1-x^2)*y+x;
r = [x,y];
F = [Fx,Fy];
```

The MATLAB output for the first and second derivatives of $\mathbf{f}(\mathbf{x})$ yields:

```
>> DF=jacobian(F,r)
DF(x, y) =
        [       0,                    -1]
        [ x*y + 1,       1/2*(x^2 - 1)]
>> DDFx=hessian(Fx,r)
DDFx(x, y) =
        [ 0, 0]
        [ 0, 0]
>> DDFy=hessian(Fy,r)
DDFy(x, y) =
        [ y, x]
        [ x, 0]
```

We fix our domain of interest by setting $C = 3$, i.e. $\mathcal{D} = [-C, C]^2 = [-3, 3]^2$.

(2) *Generate a quadratic Lyapunov Function* We fix $Q = -I$, where I is the 2×2 identity matrix and then use the built in solver for the Lyapunov equation (4) with the matrix A as the Jacobian of $\mathbf{f}$ evaluated at $\mathbf{r} = \mathbf{0}$. The built in Lyapunov equation solver of MATLAB in fact directly employs a LAPACK routine for the more general Sylvester equation (see the example code for details).

For our example the result is

$$P = \begin{pmatrix} 2.25 & -0.50 \\ -0.50 & 2.00 \end{pmatrix}.$$

(3) *Estimate B_ν for the simplices S_ν* From the `jacobian` and `hessian` in 1) the following bounds are immediate:

$$\max_{\substack{i,j=1,2 \\ \mathbf{x}\in\mathcal{D}}} |\partial_i f_j(\mathbf{x})| \le C^2 + 1 \quad \text{and} \quad \max_{\substack{i,j,k=1,2 \\ \mathbf{x}\in\mathcal{D}}} \left|\partial^2_{ij} f_k(\mathbf{x})\right| \le C.$$

From this we see that sufficiently large B_ν from the formula derived in Section 2.2 are given by

$$B_\nu = 9\left[2\left(x_\nu^{\max}\right)^2 + 1\right]. \tag{11}$$

Note that we could substitute $x_\nu^{\max} = C = 3$ for all ν. These less tighter bound would, however, imply that we might need more triangles in our triangulation for comparable results.

(4) *Estimate a in* $[-a,a]^2$ By using the second-derivatives of the components of $\mathbf{f}$ from 1) and the formula (6) one easily obtains the sufficient condition

$$\frac{1}{a} < \|P\|_2\sqrt{0^2 + (a+a+a)^2} = 2.604 \cdot 3a$$

or, e.g. $a = 0.35$ as a sufficiently small such that the orbital derivative of V_P is guarantied to be negative on $[-a,a]^2 \setminus \{\mathbf{0}\}$.

(5) *Generate* $\mathcal{P}$ We generate $N = 8,000$ random points in $\mathcal{D} = [-3,3]^2$ for our example. Note however that in the figures presented, the domain has been "framed" by a regular grid of points, guaranteeing that the Delaunay triangulation covers the domain and to avoid degenerate triangles. This is not necessary but yields prettier figures (see 1).

(6) *Delaunay Triangulation* The Delaunay triangulation for the point set $\mathcal{P}$ is constructed by the command `T=delaunayn(P)`. The CPA method requires only the values of $\mathbf{f}$ at the vertices and the B_ν for the simplices of the triangulation. It would be possible to now follow along the lines of Hafstein (2007); Giesl and Hafstein (2014) and set up an LP optimization problem to compute a CPA Lyapunov function on this type of triangulation. Here, however, we simply compute $\mathbf{f}(\mathbf{x})$ and $V_p(\mathbf{x})$ at the vertices $\mathbf{x}$. For the explicit code please refer to the downloadable `MATLAB` program mentioned above.

(7) *Verifying the Orbital Derivative* So we continue with the schematic outline and now traverse through the simplices of the triangulation $\mathcal{T} = (S_\nu)_{\nu\in I}$. For each simplex we perform the calculations of B_ν from formula (11) $B_\nu^\dagger$ from formula (10) in the program and compare them. If $B_\nu < B_\nu^\dagger$ the orbital derivative of V_P on S_ν is negative. If $B_\nu \geq B_\nu^\dagger$ the orbital derivative might be positive in S_ν or the triangle S_ν might be too large. To verify which of both possibilities applies we add the midpoint of S_ν to the set $\mathcal{P}$ to increase the resolution of the triangulation locally at S_ν.

(8) *Iteration* If we cannot guaranty that the orbital derivative is negative in all triangles S_ν, then we added new points into $\mathcal{P}$ in Step 7), and we can return to Step 6) to triangulate again with the new set $\mathcal{P}$. For the explicit code please refer, again, to the downloadable `MATLAB` program mentioned above.

5. RESULTS AND CONCLUSIONS

The three figures: 1, 2, and 3 show the results of our implemented example. On the first two figures we draw the region of attraction secured by the quadratic Lyapunov function, a result proved by our algorithm.

In figures 2 and 3 we can see how smaller simplices result in a better result, i.e. we can extend the region where

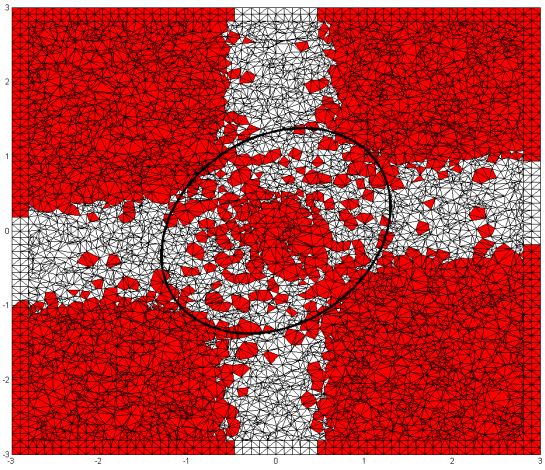

Fig. 1. First run: Delaunay-triangulation of the domain $D = [-3,3]^2$ where triangles are coloured red if the condition of negative orbital derivative for the quadratic Lyapunov function V_P is not satisfied. For the interior area points have been scattered at random but clearly not densely enough, judging from how widely the Lyapunov condition seems to be violated. At the boundary we used a more regular grid to avoid degenerated triangles.

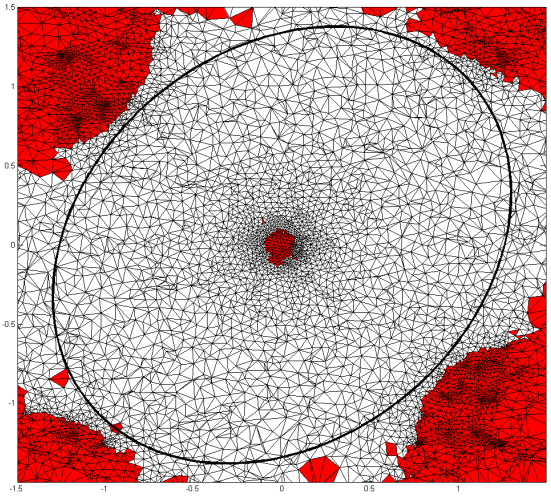

Fig. 2. Steps 6) and 7) have been repeated two times. We zoom into the Delaunay-triangulation on the domain $D = [-1.5,1.5]^2$ and triangles are coloured red if the condition of negative orbital derivative for the quadratic Lyapunov function V_P is not satisfied.

our quadratic Lyapunov function V_P is guaranteed to have a negative orbital derivative. Indeed, we can exhaust the area where the orbital derivative V_P' is negative. Still, it is clear that the condition is violated at the origin. This follows from the condition (10). To see this let $\mathcal{C} \subset \mathcal{D}$ be any compact subset such that $\mathbf{0} \notin \mathcal{C}$. If the orbital derivative V_P' is negative on $\mathcal{C}$, then $q := \max_{\mathbf{x}\in\mathcal{C}} V_P'(\mathbf{x}) <$

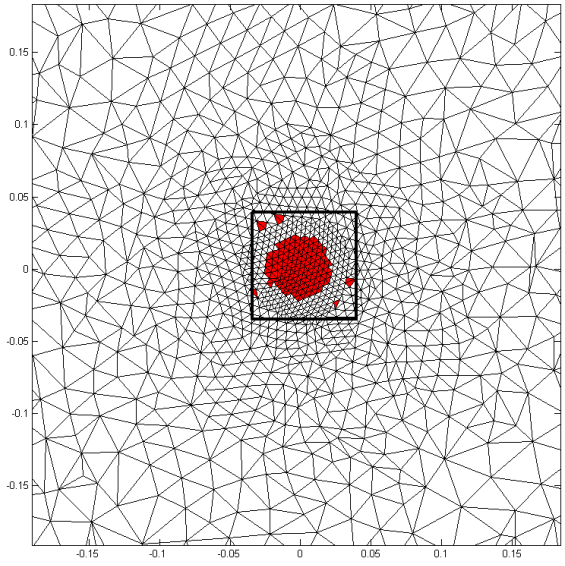

Fig. 3. Steps 6) and 7) have been repeated six times. We zoom into the Delaunay-triangulation on the domain $D = [-0.18, 0.18]^2$ and triangles are coloured red if the condition of negative orbital derivative for the quadratic Lyapunov function V_P is not satisfied.

0. Further, there is a constant $B > 0$ such that for any simplex $S_\nu \subset \mathcal{C}$ we can set $B_\nu = B$ and B_ν fulfills (9). Now, let $S_\nu \subset \mathcal{C}$ be such that $0 < h_\nu < \sqrt{-q/B}$. Then clearly for every vertex $\mathbf{x}$ of S_ν we have

$$B_\nu < \frac{-q}{h_\nu^2} \leq \frac{V'(\mathbf{x})}{h_\nu^2} = \frac{-\mathbf{x}^T P \mathbf{f}(\mathbf{x})}{h_\nu^2}$$

and the condition (10) is fulfilled.

The example presented constitute a "toy" model of sorts to test our Delaunay triangulation refinement algorithm. Note that although we have emphasized the incremental Delaunay algorithms widely available, we have not used such here. Instead we have simply re-triangulated the entire point-set $\mathcal{P}$ in each successive refinement. With the idea thusly tested, further work implementing a self-refining incremental Delaunay triangulation algorithm, for the CPA Lyapunov function context, is under way.

Lack of space only allows us to consider this simple initial example implementation of our method and MATLAB code.

REFERENCES

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

Baier, R., Grüne, L., and Hafstein, S. (2012). Linear programming based Lyapunov function computation for differential inclusions. *Discrete Contin. Dyn. Syst. Ser. B*, 17(1), 33–56.

Björnsson, J., Giesl, P., and Hafstein, S. (2014). 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. (2014). 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*, pp. 5506-5511. Los Angeles, California, USA.

Delaunay, B. (1934). Sur la sphère vide. a la mémoire de georges voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, (6), 793–800.

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

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

Giesl, P. and Hafstein, S. (2012a). Construction of Lyapunov functions for nonlinear planar systems by linear programming. *J. Math. Anal. Appl.*, 388, 463–479.

Giesl, P. and Hafstein, S. (2012b). Existence of piecewise linear Lyapunov functions in arbitrary dimensions. *Discrete Contin. Dyn. Syst.*, 32(10), 3539–3565.

Giesl, P. and Hafstein, S. (2013). Construction of a CPA contraction metric for periodic orbits using semidefinite optimization. *Nonlinear Anal.*, 86, 114–134.

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

Guibas, L., Knuth, D., and Sharir, M. (1992). Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(1-6), 381–413.

Hafstein, S. (2004). A constructive converse Lyapunov theorem on exponential stability. *Discrete Contin. Dyn. Syst.*, 10(3), 657–678.

Hafstein, S. (2007). *An algorithm for constructing Lyapunov functions*, volume 8 of *Electronic Journal of Differential Equations. Monograph.* Texas State University–San Marcos, Department of Mathematics, San Marcos, TX.

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.

Marinósson, S. (2002). *Stability Analysis of Nonlinear Systems with Linear Programming: A Lyapunov Functions Based Approach.* PhD thesis: Gerhard-Mercator-University Duisburg, Duisburg, Germany.

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

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

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

Seidel, R. (1995). The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5(2), 115 – 116.

APPENDIX

5.1 Errata

The following typos have survived into the finished article:

- Equation (10) is incorrect in the paper. It should read:

$$B_\nu < \min_{i \in J_0} \left[\frac{-\mathbf{x}_i^T P \mathbf{f}(\mathbf{x}_i)}{n\, h_\nu^2} \right] =: B_\nu^\dagger.$$

  This error does not have any qualitative effect on the results of the paper although it may in some instances make more iterations and finer triangulations necessary in order for the condition in the equation to be fulfilled. We thank Alexander Weber from the Department of Control Engineering at the University of the Federal Armed Forces in Munich for his pointing this error out to us!
- The error of equation (10) from above made its way into the MATLAB code which was made available online at: http://www.ru.is/kennarar/sigurdurh/M2015M.rar. The code has now been corrected (as of 17/8/2015).