Verification of a Numerical Solution to a Collocation Problem

Hjortur Bjornsson and Sigurdur Hafstein
Science Institute and Faculty of Physical Sciences, University of Iceland, Dunhagi 5, 107 Reykjavík, Iceland

Keywords: Numerical Computation, Lyapunov Function, Radial Basis Functions.

Abstract: In a recent method to compute Lyapunov functions for nonlinear stochastic differential equations a subsequent verification of the results is needed. The theory has been developed but there are several practical difficulties in its implementation because of the huge amount of function evaluations needed during verification. We study several different methods and compare their accuracy and efficiency.

1 INTRODUCTION

We will discuss numerical solutions to Partial Differential Equations (PDE) that arise when computing Lyapunov functions for Stochastic Differential Equations (SDE) and, in particular, how the validity of the computed Lyapunov functions can be verified numerically. In a novel numerical method (Bjornsson et al., 2018) we obtain a numerical solution to a PDE, and that solution is supposed to be a Lyapunov function for a certain SDE. To guarantee that the numerical solution is in fact a Lyapunov function, we have an error estimate which states that if the value of the numerical solution on a certain grid of points is lower then some constant, then the numerical solution is indeed a Lyapunov function for the system. The theory supporting this novel method was developed in (Gudmundsson and Hafstein, 2018; Hafstein et al., 2018), and in (Bjornsson et al., 2018) the method is developed and it is shown that it converges to a true Lyapunov function if the collocation grid used for the numerical solution of the PDE is sufficiently dense. However, one must verify a posteriori on an evaluation grid that the collocation grid was indeed adequate. An issue with the method is that the evaluation grid is so dense that we need to evaluate the computed Lyapunov function at typically 10^9, and even up to 10^{16}, points. Note that the Lyapunov function is computed using Radial Basis Functions (RBF) and to evaluate it at a point, one must sum over all the RBFs used in the computation, i.e. the sum contains a number of terms that is equal to the number of the collocation points used. Here we will compare the numerical errors and performances of various methods used for this nontrivial and involved evaluation.

2 BACKGROUND

For completeness we give a quick background with many of the details omitted. For full details see (Gudmundsson and Hafstein, 2018; Hafstein et al., 2018; Bjornsson et al., 2018). We consider a d-dimensional SDE of the form

$$dX(t) = f(X(t))dt + g(X(t))dW(t), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$, $g : \mathbb{R}^d \to \mathbb{R}^{d\times Q}$, $f(0) = g(0) = 0$, and $W(t)$ is a Q-dimensional Brownian motion. We are specifically interested in the stability of the trivial solution $X = 0$ of the system.

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with a smooth boundary $\Gamma = \partial\Omega$. We solve numerically the boundary problem of the PDE given by:

$$\begin{cases} LV(x) = r(x) & \text{for } x \in \Omega, \\ V(x) = c(x) & \text{for } x \in \Gamma, \end{cases} \tag{2}$$

where L denotes the following differential operator associated with the system given in equation (1):

$$LV(x) = \frac{1}{2}\sum_{i,j=1}^{d} m_{ij}(x)\frac{\partial^2 v}{\partial x_i \partial x_j}(x) + \sum_{i=1}^{d} f_i(x)\frac{\partial v}{\partial x_i}(x), \tag{3}$$

where $(m_{ij}(x))_{i,j} = g(x)g(x)^\top$. For suitable functions $r(x)$ and $c(x)$ the solution to this PDE will be a Lyapunov function asserting the asymptotic stability in probability of the trivial solution and we can use it to estimate its probabilistic basin of attraction.

To solve this PDE numerically we use the RBF method similar to (Giesl, 2007; Giesl, 2008; Giesl and Wendland, 2007), where Lyapunov functions are computed for deterministic ordinary differential equations (ODEs), but adapted to SDEs. Given a set

Bjornsson, H. and Hafstein, S.
Verification of a Numerical Solution to a Collocation Problem.
DOI: 10.5220/0006945405870594
In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018) - Volume 1*, pages 587-594
ISBN: 978-989-758-321-6

of points $X_1 = \{x_1, \ldots, x_N\} \subset \Omega \subset \mathbb{R}^d$ and $X_2 = \{\xi_1, \ldots, \xi_M\} \subset \Gamma$, we solve the interpolation problem

$$\begin{cases} LV(x_i) = r(x_i) & \text{for all } i = 1, \ldots, N, \\ V(\xi_i) = c(\xi_i) & \text{for all } i = 1, \ldots, M. \end{cases} \tag{4}$$

The solution to the interpolation problem is given by

$$\begin{aligned} V(x) = &\sum_{k=1}^{N} \alpha_k (\delta_{x_k} \circ L)^y \psi(\|x-y\|) \\ &+ \sum_{k=1}^{M} \alpha_{N+k} (\delta_{\xi_k} \circ L^0)^y \psi(\|x-y\|) \end{aligned} \tag{5}$$

where $\delta_y V(x) = V(y)$ and the superscript y denotes that the operator is applied with respect to the variable y, $L^0 = id$ and $\psi = \psi_{l,k}$ is a so-called Wendland function, which are compactly supported RBFs (Wendland, 1998). The constants α_k in equation (5) can be determined as the solution to a certain linear system $A\alpha = \gamma$, where the matrix A is symmetric and positive definite; see (Bjornsson et al., 2018) for full details on the matrix A and the vector γ.

2.1 Lyapunov Function

The method described in the preceding section is used to compute a certain Lyapunov function for the system (1), whose domain does not include the equilibrium at the origin, hence the name *non-local Lyapunov function*. Again see (Gudmundsson and Hafstein, 2018; Hafstein et al., 2018; Bjornsson et al., 2018) for details.

Definition (Non-Local Lyapunov Function)**.** Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^d$, $\mathcal{B} \subset \mathcal{A}^\circ$, be simply connected compact neighbourhoods of the origin with C^2 boundaries and set $\mathcal{U} := \mathcal{A} \setminus \mathcal{B}^\circ$. A function $V \in C^2(\mathcal{U})$ for the system (1) such that

(1) $0 \le V(x) \le 1$ for all $x \in \mathcal{U}$,

(2) $LV(x) < 0$ for all $x \in \mathcal{U}$, and

(3) $V^{-1}(0) = \partial\mathcal{B}$ and $V^{-1}(1) = \partial\mathcal{A}$,

is called a *non-local Lyapunov function* for the system (1), and we refer to $\partial\mathcal{B}$ and $\partial\mathcal{A}$ is the inner and outer boundary of $\mathcal{U}$, respectively.

To compute a non-local Lyapunov function for the system (1) we look for solutions of the PDE (2) with $r(x) = -h$ where $h > 0$ is a small constant, $c(x) = 1$ for $x \in \partial\mathcal{A}$ and $c(x) = 0$ for $x \in \partial\mathcal{B}$. Because we compute a numerical approximation to the solution of the system we can not expect $LV(x) = -h$ to hold for all $x \in \mathcal{U}$, but since the definition of a non-local Lyapunov function only requires $LV(x) < 0$, our numerical approximation will still be a true Lyapunov function as long as there is no point in $x \in \mathcal{U}$ with $LV(x) \ge 0$. In this paper we are concerned with the numerical verification of the condition that $LV(x) < 0$ holds for all $x \in \mathcal{U}$. The theory for this verification is developed in (Bjornsson et al., 2018). Here we are interested in the nontrivial technical details of its efficient implementation.

2.2 Explicit Formulas

For the explicit computations of our Lyapunov function, one must choose a specific Wendland function $\psi_{k,l}(x)$, where the indices k, l are non-negative integers. In the one dimensional case we have used $\psi_{7,6}$ and for two dimensional cases we used $\psi_{8,6}$. Note that there are certain restriction on the indices of the Wendland function for the problem at hand and in our computations we need at least these rather large indices.

For example, assuming we are using the Wendland function $\psi_{7,6}$ and the RBF constant $c > 0$, we define by some abuse of notation $\psi_0(r) := \psi_{7,6}(cr)$ for $r \ge 0$. Then we define $\psi_{i+1}(r) = \frac{1}{r}\frac{\partial}{\partial r}\psi_i(r)$ recursively for $i = 0, 1, 2, 3$. Finally, we define $W_i(x) = \psi_i(x/c)$. The formulas for the functions W_i starting with $\psi_0 := \psi_{7,6}$ are the following:

$$\begin{aligned} W_0(x) = &(1-x)^{13}(4096x^6 + 7059x^5 + 5751x^4 \\ &+ 2782x^3 + 830x^2 + 143x + 11), \end{aligned} \tag{6}$$

$$\begin{aligned} W_1(x) = &-38c^2(1-x)^{12}(2048x^5 + 2697x^4 \\ &+ 1644x^3 + 566x^2 + 108x + 9), \end{aligned} \tag{7}$$

$$\begin{aligned} W_2(x) = &10336c^4(1-x)^{11}(128x^4 \\ &+ 121x^3 + 51x^2 + 11x + 1), \end{aligned} \tag{8}$$

$$\begin{aligned} W_3(x) = &-62016c^6(1-x)^{10}(320x^3 \\ &+ 197x^2 + 50x + 5), \text{ and} \end{aligned} \tag{9}$$

$$W_4(x) = 3224832c^8(1-x)^9(80x^2 + 27x + 3). \tag{10}$$

Note that these formulas are only valid for $0 \le x \le 1$ and for $x > 1$ we have $W_i(x) = 0$. The parameter c is referred to as the RBF constant and is used to control the size of the support of the functions $\mathbb{R}^d \to \mathbb{R}^d$, $x \mapsto \psi_i(\|x\|)$, i.e. the support is a ball of radius $1/c$ around the origin.

The explicit formulas for $V(x)$ and $LV(x)$ from our numerical solution to the interpolation problem (4), and by writing $\beta = x - x_k$ for brevity, are given by the

following expressions:

$$V(x) = \sum_{k=1}^{N} \alpha_k \Big[-\psi_1(\|\beta\|)\langle \beta, f(x_k)\rangle + \frac{1}{2}\sum_{i,j=1}^{d} m_{ij}[\psi_2(\|\beta\|)\beta_i\beta_j + \delta_{ij}\psi_1(\|\beta\|)]\Big] + \sum_{k=1}^{M} \alpha_{N+k}\psi_0(\|\beta\|) \tag{11}$$

and

$$\begin{aligned} LV(x) &= \sum_{k=1}^{N} \alpha_k \Big[-\psi_2(\|\beta\|)\langle \beta, f(x)\rangle\langle \beta, f(x_k)\rangle \\ &\quad - \psi_1(\|\beta\|)\langle (f(x), f(x_k))\rangle \\ &\quad + \frac{1}{2}\sum_{i,j=1}^{d} m_{ij}(x_k)\Big[\psi_3(\|\beta\|)\langle \beta, f(x)\rangle\beta_i\beta_j \\ &\quad + \psi_2(\|\beta\|)f_j(x)\beta_i \\ &\quad + \psi_2(\|\beta\|)f_i(x)\beta_j + \delta_{ij}\psi_2(\|\beta\|)\langle \beta, f(x)\rangle\Big] \\ &\quad + \frac{1}{2}\sum_{i,j=1}^{d} m_{ij}(x)\Big[-\psi_3(\|\beta\|)\langle \beta, f(x_k)\rangle\beta_i\beta_j \\ &\quad - \psi_2(\|\beta\|)f_j(x_k)\beta_i \\ &\quad - \psi_2(\|\beta\|)f_i(x_k)\beta_j - \delta_{ij}\psi_2(\|\beta\|)\langle \beta, f(x_k)\rangle\Big] \\ &\quad + \frac{1}{4}\sum_{r,s=1}^{d}\sum_{i,j=1}^{d} m_{rs}(x)m_{ij}(x_k)\Big[\psi_4(\|\beta\|)\beta_i\beta_j\beta_r\beta_s \\ &\quad + \psi_3(\|\beta\|)[\delta_{ij}\beta_r\beta_s + \delta_{ir}\beta_j\beta_s \\ &\quad + \delta_{is}\beta_j\beta_r + \delta_{jr}\beta_i\beta_s + \delta_{js}\beta_i\beta_s + \delta_{rs}\beta_i\beta_j] \\ &\quad + \psi_2(\|\beta\|)[\delta_{ij}\delta_{rs} + \delta_{ir}\delta_{js} + \delta_{is}\delta_{jr}]\Big]\Big] \\ &\quad + \sum_{k=1}^{M} \alpha_{N+k}\Big[-\psi_1(\|\xi_k - x\|)\langle \xi_k - x, f(x)\rangle \\ &\quad + \frac{1}{2}\sum_{i,j=1}^{d} m_{ij}(x)[\psi_2(\|\xi_k - x\|)(\xi_k - x)_i(\xi_k - x)_j \\ &\quad + \delta_{ij}\psi_1(\|\xi_k - x\|)]\Big]. \end{aligned} \tag{12}$$

In these formulas $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{N+M})^\top$ is the solution to $A\alpha = \gamma$ associated with the interpolation problem (4) and β_i is the i-th component of the vector $\beta = x - x_k$. Note the above formulas are independent of which Wendland function $\psi_{k,l}$ we start with in the beginning.

3 VERIFICATION

Let $\mathcal{A}, \mathcal{B}$, and $\mathcal{U}$ be as in the definition of non-local Lyapunov functions and let $V(x)$ be a numerical approximation like is described afterwards. Now by (Bjornsson et al., 2018, Theorem 4.3), if

$$\nu := \max_{y \in Y_{\mathcal{U}}} LV(y) + C_u \frac{d^2}{4} h^2 < 0, \tag{13}$$

then V is a non-local Lyapunov function for the system. Here d is the dimension of the system, $h > 0$ is a parameter controlling the density of the evaluation grid, and $Y_{\mathcal{U}}$ is the evaluation grid that covers $\mathcal{U}$. Finally the constant $C_{\mathcal{U}}$ is an upper estimate on the second derivatives of our function LV, for further details see (Bjornsson et al., 2018; Mohammed and Giesl, pted).

3.1 One Dimensional Example

For an explicit example let us consider the one dimensional SDE from (Bjornsson et al., 2018)

$$dx(t) = \sin(x(t))\,dt + \frac{3x(t)}{1+x(t)^2}\,dW(t), \tag{14}$$

where W is a one dimensional Brownian motion. We determine an approximate solution to the PDE:

$$\begin{cases} LV(x) &= -10^{-3} \text{ for } 10^{-2} < x < 8, \\ V(x) &= 0 \text{ for } x = 10^{-1}, \\ V(x) &= 1 \text{ for } x = 8. \end{cases} \tag{15}$$

The approximate solution was determined using the Wendland function $\psi_{7,6}$, the RBF constant $c = 2$, and 700 evenly spaced collocation points on the interval $[1.1 \cdot 10^{-2}, 7.99]$. Figure 1 shows the numerical solution to PDE (15), for the system in equation (14). For this system and these values the estimate $C_{\mathcal{U}}$ is equal to $1.6846 \cdot 10^{12}$.

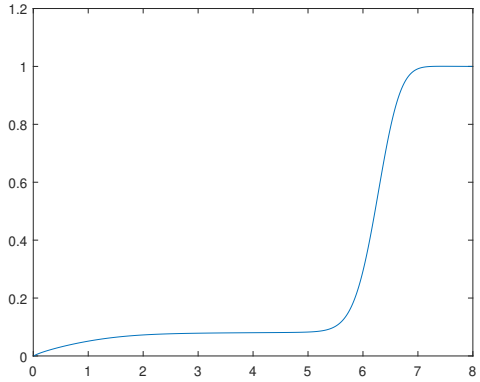

Figure 1: Numerical solution of $V(x)$ in equation (15).

By estimating through rough computations that the maximum value of our numerical approximation to $LV(x)$ is 0.3×10^{-3}, we get by setting $\nu = 0$ in equation (13)

$$h = \sqrt{\frac{4 \cdot 0.3 \cdot 10^{-3}}{1.6846}} = 2.6 \cdot \times 10^{-8}.$$

To give us some safety, which is needed since ν has to be strictly negative, we choose $h = 2.18 \cdot 10^{-8}$. This corresponds to evaluating the function LV at $7.5 \cdot 10^8$ evenly spaced points $Y_{\mathcal{U}}$ on the interval $[10^{-2} - h, 8 + h]$. This large number of points that needs to be evaluated takes about 10 seconds on a normal desktop computer (i7 4790k). The maximum value of LV found on this grid is $-0.281 \cdot 10^{-3}$ and thus

$$\nu = -0.281 \cdot 10^{-3} + 0.19119 \cdot 10^{-3} < 0,$$

so our approximation is a non-local Lyapunov function.

3.2 Two Dimensional Example

For a second explicit example consider the two dimensional system from (Grüne and Camilli, 2003) also studied in (Bjornsson et al., 2018), given by

$$dx(t) = (M + \rho(x(t))I)x(t)\,dt + g(x(t))\,dW(t), \quad (16)$$

where W is a one dimensional Brownian motion, I the 2×2-identity matrix,

$$M = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \ \rho(x) = \|x\| - 1,$$

and

$$g(x) = \|x\| \left(\|x\| - \frac{1}{2} \right) \left(\|x\| - \frac{3}{2} \right) x.$$

We find an approximate solution to the PDE, $LV(x) = -10^{-2}$, on an annulus around the origin with inner radius 0.4 and outer radius 1.9 with $V(x) = 0$ for $\|x\| = 0.4$ and $V(x) = 1$ for $\|x\| = 1.9$. To calculate the approximation we used the Wendland function $\psi_{8,6}$ and a grid of 80×80 points evenly spaced on the annulus. Figure 2 shows the resulting numerical approximation for the system in equation (16). For this system, and using $\psi_{8,6}$, the constant $C_{\mathcal{U}}$ is determined to be $4.3220 \cdot 10^{12}$, and following similar calculations as in the preceding section we estimate the maximum value of LV to be ≈ -0.005, this gives us $h = 3.4013 \cdot 10^{-8}$, so we need to evaluate LV on a grid with $(1.1760 \cdot 10^8)^2 \approx 10^{16}$ points.

3.3 Comparison of Methods

In the inner-most loop of our program we have to evaluate W_i, $i = 1,2,3,4$. For simplicity we consider only the polynomials resulting from the Wendland function $\psi_{7,6}$ given by equations (7)-(10) and we

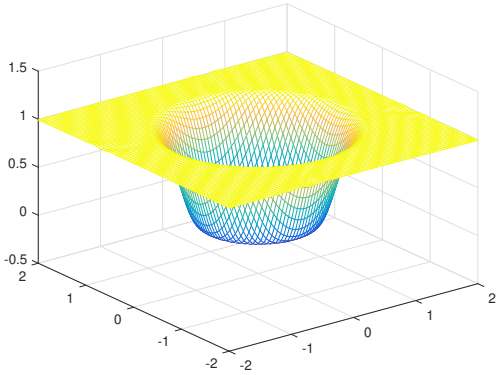

Figure 2: Numerical solution of $V(x)$ for the two dimensional system (16).

fixed $c = 2$; the function (6) is unused as LV does not depend on it. We tried different Wendland functions but the results were comparable. Fast evaluation of these functions is critical for the performance of our verification, therefore we tested 5 different evaluation methods: having these functions hardcoded in factorized form as in (7)-(10), expanding the polynomials and using Horner's method for the evaluation, using Lookup-tables, using a Lookup-table and additionally applying linear interpolation, and combining the two previous approaches on different subintervals. All tests were written in C and compiled using *gcc* with optimization flag *-O2*. The figures in the following section only show function (10) since it has the largest numerical errors. The errors for the other functions, i.e. (7)-(9), are qualitatively identical but of lower magnitude.

3.3.1 Evaluation with Hardcoded Functions

We hardcoded the polynomials as they are written in equations (7)-(10).

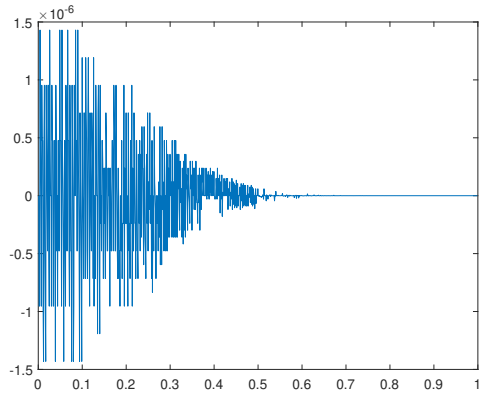

Figure 3: Absolute error as a function of x of W_4 using Hardcoded Functions. Note the scale on the y-axis is $\times 10^{-6}$.

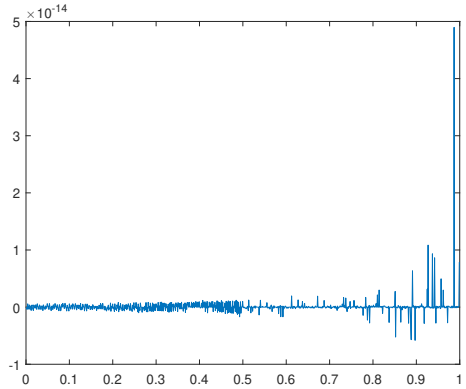

Figure 4: Relative error as a function of x of W_4 using Hardcoded Functions. Note the scale on the y-axis is $\times 10^{-14}$.

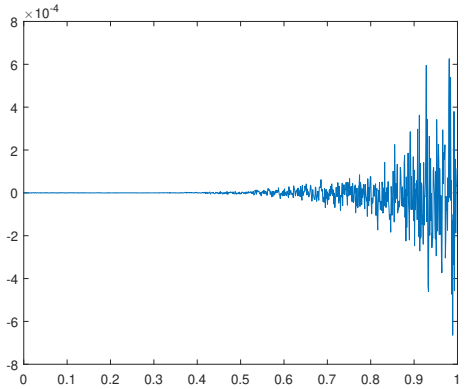

Figure 5: Absolute error as a function of x of W_4 using Horner's method. Note the scale on the y-axis is $\times 10^{-4}$.

Figures 3 and 4 show the absolute and relative error, respectively, of the function W_4 compared to the values obtained from infinite precision arithmetic truncated to 64-bit floating point values. These figures show us that this method, i.e. having hardcoded functions, gives us the best accuracy out of all of the methods tested.

3.3.2 Evaluation using Horner's Method

We expanded the polynomials, i.e. obtained coefficients $a_n, a_{n-1}, \ldots, a_0$ such that

$$W_i(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0.$$

Obviously the coefficients a_i depend on which Wendland function $\psi_{k,l}$ we started with. Having the polynomials in expanded form allows us to evaluate them at any point x using the following scheme (Horner's method):

```
Horner(x, [a_n])
    acc:=0;
    for(i=n;i>=0;i--)
        acc=acc*x;
        acc=acc+a_i;
    return acc;
```

By taking advantage of SIMD-instructions (Single Instruction, Multiple Data) we can evaluate two of these polynomials at a time in double precision arithmetic, or even all four at the same time on machines that support 256-bit wide SIMD registers (AVX2 or later).

Since the polynomial functions have a high order zero at $x = 1$, and the coefficients are relatively large, we get significant absolute errors in the evaluation when x is close to 1, see figure 5. For the function W_4 the relative error close to $x = 1$ explodes, as the value of the function is close to 0 there. Using higher values for the Wendland RBF constant c exaggerates this behaviour, so it is virtually impossible to use Horner's method to evaluate the polynomials with sufficient accuracy with $c = 10$ or higher.

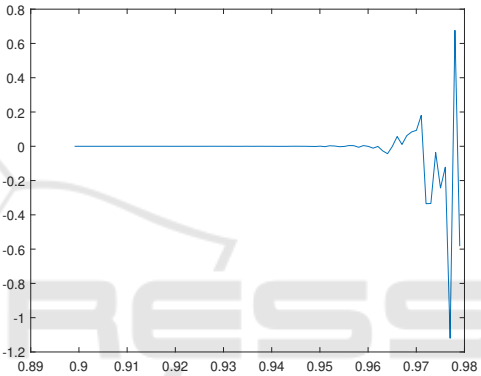

Figure 6: Relative error as a function of x of W_4 using Horner's method. Note the scale on the y-axis is $\times 10^{0}$.

Figure 6 shows us how the relative error of the Horner's method explodes the closer we get to $x = 1$.

3.3.3 Evaluation using Lookup-tables

We pre-evaluate the polynomials at $K = 10^7$ evenly spaced points, $x_0, \ldots, x_K$ between 0 and 1, using infinite precision arithmetic and then truncate and store the results. At runtime we evaluate $W_j(x)$ by finding i such that x_i is the closest value to x and return $W_j(x) \approx W_j(x_i)$. Here is a pseudo-code of the full procedure:

```
//j selects W_j
Lookuptable(x, j)
    i=round(x*(K-1));
    return W_j[i];
```

The tables are constructed in such a way that we can use the same index to get the values of all of the W_j functions, and furthermore by weaving the tables together we can evaluate all four of them by read-

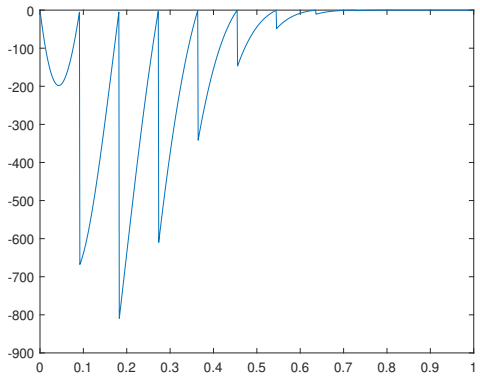

Figure 7: Absolute error as a function of x of W_4 using Lookup-table. Note the scale on the y-axis is $\times 10^0$.

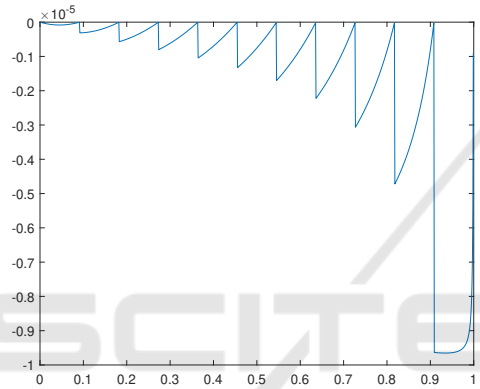

Figure 8: Relative error as a function of x of W_4 using Lookup-table. Note the scale on the y-axis is $\times 10^{-5}$.

ing twice from memory, or even once on AVX2 capable machines. This is a significant performance boost compared to the Horner's method, see table 1. As for the absolute error see figure 7. This kind of sawtooth shape is typical for Lookup-tables, however, note that the absolute error is high for x close to 0 but stabilizes when x gets closer 1. Since the true value of the function W_4 close to 0 is very high, this translates to a low relative error around 0, see figure 8.

3.3.4 Lookup-tables with Linear Interpolation

One improvement on the Lookup-table method described in the previous section is to use linear interpolation between the lookup values to get more accurate evaluations at the cost of some processing time. Here is a pseudo-code for the procedure:

```
Lookuptable_interpolate(x,j)
    i=floor(x*(K-1));
    interpolant:=(x-x_n[i])/  ...
       ...(x_n[i+1]-x_n[i]);
    return W_j[i] + ...
       ...interpolant*(W_j[i+1]-W_j[i]);
```

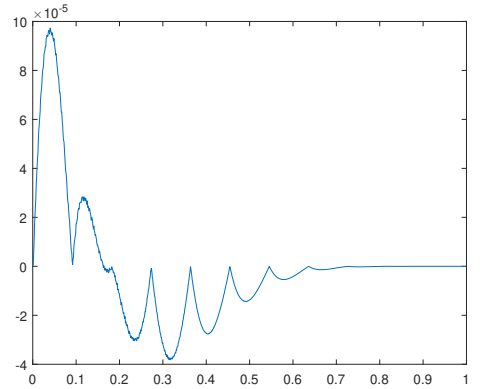

Figure 9: Absolute error as a function of x of W_4 using Lookup-table with interpolation. Note the scale on the y-axis is $\times 10^{-5}$.

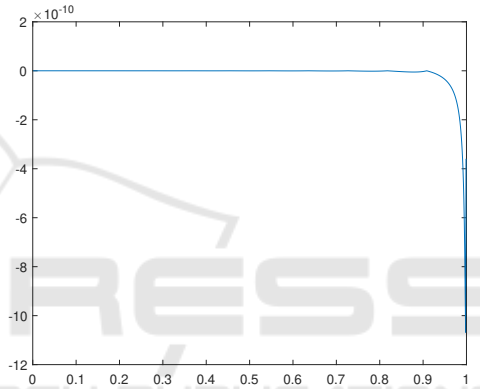

Figure 10: Relative error as a function of x of W_4 using Lookup-table with interpolation. Note the scale on the y-axis is $\times 10^{-10}$.

There is a significant increase in accuracy, see figure 9, but this method requires more memory access than the previous method.

3.3.5 Combination of Lookup-table Methods

As noted in the previous sections, the accuracy of the Lookup-table method is quite good when x is close to 1, therefore we propose the following method: we start by selecting a cutoff value b between 0 and 1, then when we evaluate $W_j(x)$ we use the simple lookup method if $x \geq b$, otherwise we use the Lookup-table with interpolation.

```
Lookuptable_combined(x,j)
    if(x>=b)
        return Lookuptable(x,j);
    else
        return Lookuptable_interpolate(x,j);
```

Note the large absolute error around the value $x = 0.8$ in figure 11 translates to a low relative error of

Table 1: Total execution time to evaluate the functions W_i, $i = 1,2,3,4$, at 10^7 different points, using a single thread.

Method / Processor	i5-8250U	i5-3210M	i5-2500k	i7-4790K
Horner's method	548.1ms	721ms	657.8ms	395ms
Simple Lookup-table	125.8ms	152.7ms	146.7ms	105ms
Lookup-table interp.	165.5ms	231.9ms	216.9ms	128ms
Lookup-table comb.	148.2ms	187.6ms	172.5ms	105ms
Hardcoded	171.5ms	214.8ms	199.8ms	107ms

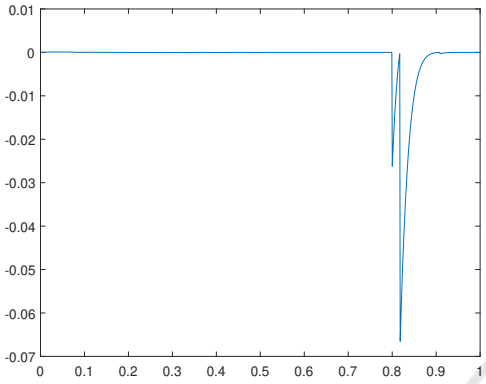

Figure 11: Absolute error as a function of x of W_4 using Lookup-table with $b = 0.8$. Note the scale on the y-axis is $\times 10^0$.

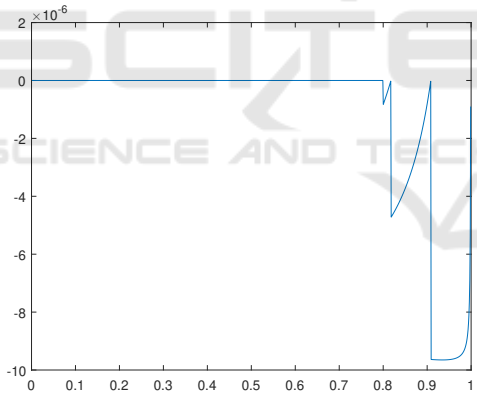

Figure 12: Relative error as a function of x of W_4 using Lookup-table with $b = 0.8$. Note the scale on the y-axis is $\times 10^{-6}$.

$8 \cdot 10^{-6}$, see figure 12. The value of $b = 0.8$ was empirically determined to give the best tradeoff between speed and accuracy.

4 CONCLUSIONS

We compared different methods to evaluate polynomials stemming from Wendland's compactly supported radial basis functions. In our application for rigidly verifying the negativity of Lyapunov functions computed for stochastic differential equations these polynomials have to be evaluated at numerous points and this has to be done with sufficient accuracy. Table 1 shows how long it takes to evaluate the polynomials W_1, W_2, W_3, and W_4 at 10^7 different points on different processors. The fastest method is to use a Lookup-table, but it is too inaccurate for practical use, at least in our application. Using linear interpolation between the lookup-values produced much more accurate results, but the method is considerably slower. A more efficient method that is sufficiently accurate is to use linear interpolation between the lookup-values in the most troublesome areas of the Lookup-table, and just use simple Lookup-table otherwise. Hardcoding the polynomials in factorized form as in equations (7)-(10) is both very fast, although not as fast as using a Lookup-table, and very accurate. Horner's method should be avoided since it produces inaccurate results and is slow. The inaccuracy is supposedly due to the multiple zero at 1 of the functions W_i.

ACKNOWLEDGEMENTS

The research for this paper was supported by the Icelandic Research Fund (Rannís) in the project 'Lyapunov Methods and Stochastic Stability' (152429-051), which is gratefully acknowledged.

REFERENCES

Bjornsson, H., Gudmundsson, S., Giesl, P., Hafstein, S., and Scalas, E. (2018). Computation of the stochastic basin of attraction by rigorous construction of a Lyapunov function. submitted.

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

Giesl, P. (2008). Construction of a local and global Lyapunov function using Radial Basis Functions. *IMA J. Appl. Math.*, 73(5):782–802.

Giesl, P. and Wendland, H. (2007). Meshless collocation: error estimates with application to dynamical systems. *SIAM J. Numer. Anal.*, 45(4):1723–1741.

Grüne, L. and Camilli, F. (2003). Characterizing attraction probabilities via the stochastic Zubov equation. *Discrete Contin. Dyn. Syst. Ser. B*, 3(3):457–468.

Gudmundsson, S. and Hafstein, S. (2018). Probabilistic basin of attraction and its estimation using two Lyapunov functions. *Complexity*, Article ID:2895658.

Hafstein, S., Gudmundsson, S., Giesl, P., and Scalas, E. (2018). Lyapunov function computation for autonomous linear stochastic differential equations using sum-of-squares programming. *Discrete Contin. Dyn. Syst. Ser. B*, 2(23):939–956.

Mohammed, N. and Giesl, P. (accepted). Grid refinement in the construction of Lyapunov functions using Radial Basis Functions. *Discrete Contin. Dyn. Syst. Ser. B*.

Wendland, H. (1998). Error estimates for interpolation by compactly supported Radial Basis Functions of minimal degree. *J. Approx. Theory*, 93:258–272.