Advanced algorithm for interpolation with Wendland functions*

Hjortur Bjornsson and Sigurdur Hafstein[0000−0003−0073−2765]

Science Institute, University of Iceland, Dunhagi 3, 107 Reykjavík, Iceland
`hjb6@hi.is, shafstein@hi.is`

Abstract. We develop and study algorithms for computing Lyapunov functions using meshless collocation and Wendland functions. We present a software tool that generates a C/C++ library that implements Wendland functions of arbitrary order in a specified factorized form with advantageous numerical properties. Additionally, we describe the algorithm used by the tool to generate these Wendland functions. Our factorized form is more efficient and has higher numerical accuracy than previous implementations. We develop and implement optimal grid generation for the interpolation problem using the Wendland functions. Finally, we present software that calculates Lyapunov functions using these Wendland functions and the optimally generated grid. The software tool and library are available for download with examples of usage.

Keywords: Wendland function, Lyapunov functions, radial basis functions, code generation.

1 Introduction

Interpolation and collocation using Radial Basis Functions (RBF), in particular compactly supported RBFs, have been the subject of numerous research activities in the past decades [27, 9, 10, 24, 7, 6, 25, 26]. They are well suited as kernels of Reproducing Kernel Hilbert Spaces and their mathematical theory is mature. The authors and their collaborators have applied Wendland's compactly supported RBFs for computing Lyapunov functions for nonlinear systems, both deterministic [11, 12, 14] and stochastic [5], where Lyapunov functions are a useful tool to analyse stability of these systems, cf. e.g. [18, 22, 23, 19, 20]. Various numerical methods have been used to find Lyapunov functions for the systems at hand [14, 16]. Meshless collocation using RBFs is one such method and many different families of RBFs have been studied [25].

In the papers [12–14, 5] and the book [11] meshless collocation is used with Wendland functions, where the Wendland function family is defined in a recursive way and in order to determine the actual functions to use in a software implementation many calculations had to be done by hand. In [2] an algorithm

* This research was supported by the Icelandic Research Fund (Rannis), grant number 152429-051, Lyapunov Methods and Stochastic Stability.

is proposed that determines the Wendland polynomials in expanded form, that is: for each pair of integers $l, k \geq 0$, it finds a list of numbers $a_0, a_1, \ldots a_d$ such that the Wendland function $\psi_{l,k}(r) = \sum_{i=0}^{d} a_i r^i$. However, it was shown in [3] that the evaluations of these polynomials in this form using typical schemes, such as Horner's scheme, can lead to significant numerical errors.

Having the Wendland functions in factorized form [3] is more efficient and numerically accurate, so we propose an alternate method to determine the functions in factorized form. For that purpose, we have created a software tool that generates a reusable software library in C/C++, which implements these Wendland polynomials in factorized form. A first version of this software library was presented in [4]. We have now extended it considerably and added more functionality, most notably efficient grid generation and algorithms to solve interpolation problems for generating Lyapunov functions for stochastic and deterministic dynamical systems.

2 Background

Meshless collocation with RBFs is a method that can be used to calculate Lyapunov functions for either stochastic or deterministic dynamical systems. In paper [5] meshless collocation was used to calculate Lyapunov functions for Stochastic Differential Equations (SDE); see e.g. [11, 14] for a similar approach for deterministic systems.

The method revolves around solving a linear Partial Differential Equation (PDE). Let $\Omega \subset \mathbb{R}^n$ be a given domain and $\Gamma \subset \mathbb{R}^n$ its boundary. Then we want to solve the (PDE)

$$\begin{cases} LV(\mathbf{x}) = h(\mathbf{x}) & \mathbf{x} \in \Omega \\ V(\mathbf{x}) = c(\mathbf{x}) & \mathbf{x} \in \Gamma, \end{cases}$$

where L is a certain differential operator, and h and c are some appropriately chosen functions.

Using meshless collocation to solve the PDE above we choose points $X_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \Omega$ and $X_2 = \{\xi_1, \ldots, \xi_M\} \subset \Gamma$, and solve the interpolation problem

$$\begin{cases} LV(\mathbf{x}_i) = h(\mathbf{x}_i) & \text{for all } i = 1, \ldots, N \\ V(\xi_i) = c(\xi_i) & \text{for all } i = 1, \ldots, M. \end{cases}$$

The solution is then given in terms of a radial basis function ψ,

$$V(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k (\delta_{\mathbf{x}_k} \circ L)^{\mathbf{y}} \psi(\|\mathbf{x} - \mathbf{y}\|) + \sum_{k=1}^{M} \alpha_{N+k} (\delta_{\xi_k} \circ L^0)^{\mathbf{y}} \psi(\|\mathbf{x} - \mathbf{y}\|), \quad (1)$$

where L^0 is the identity operator, $\delta_{\mathbf{y}} V(\cdot) = V(\mathbf{y})$ and superscript $\mathbf{y}$ denotes that the operator is applied with respect to the variable $\mathbf{y}$.

The constants α_i are determined as a solution to the linear system

$$A\alpha = \gamma, \tag{2}$$

where A, called the interpolation matrix, is the symmetric matrix

$$A = \begin{bmatrix} B & C \\ C^T & D \end{bmatrix} \tag{3}$$

and the matrices $B = (b_{jk})_{j,k=1,\dots,N}$, $C = (c_{jk})_{j=1,\dots,N,k=1,\dots,M}$ and $D = (d_{jk})_{j,k=1,\dots,M}$ have elements:

$$\begin{aligned} b_{jk} &= (\delta_{\mathbf{x}_j} \circ L)^{\mathbf{x}}(\delta_{\mathbf{x}_k} \circ L)^{\mathbf{y}}\psi(\mathbf{x}-\mathbf{y}) \\ c_{jk} &= (\delta_{\mathbf{x}_j} \circ L)^{\mathbf{x}}(\delta_{\xi_k} \circ L^0)^{\mathbf{y}}\psi(\mathbf{x}-\mathbf{y}) \\ d_{jk} &= (\delta_{\xi_j} \circ L^0)^{\mathbf{x}}(\delta_{\xi_k} \circ L^0)^{\mathbf{y}}\psi(\mathbf{x}-\mathbf{y}). \end{aligned}$$

The vector γ has components given by

$$\begin{aligned} \gamma_j &= r(\mathbf{x}_j), \quad 1 \le j \le N \\ \gamma_{j+N} &= c(\xi_j), \quad 1 \le j \le M \end{aligned}$$

There are different choices for the radial basis function ψ. We want the interpolation matrix A to be symmetric and positive definite and choosing ψ to have compact support can make A sparse. Under a few mild conditions the choice of ψ as a Wendland function, i.e. a compactly supported radial basis function, ensures this [26].

This method works the same way for determining Lyapunov functions for both deterministic systems and SDEs, the difference is the choice of the differential operator L. For deterministic systems it is the orbital derivative, a first order differential operator, and for stochastic systems it is a second order differential operator.

To compute such a Lyapunov function a large number of evaluations of the function ψ and its derivatives is necessary, see e.g. the examples given in equations (16) and (19). To verify the properties of a Lyapunov function for the function computed, even more evaluations are necessary. Therefore, it turned out to be essential that these evaluations could be carried out in an efficient and accurate way.

3 Wendland functions

The Wendland functions are compactly supported radial basis functions that are polynomials on their support, which makes computations with them simple. They are a family of functions depending on two parameters $l, k \in \mathbb{N}_0$ defined by the recursive relations:

$$\psi_{l,0}(r) = [(1-r)_+]^l \tag{4}$$

and

$$\psi_{l,k+1}(r) = C_{l,k+1} \int_r^1 t\psi_{l,k}(t)\mathrm{d}t, \tag{5}$$

where $(1-r)_+ := \max\{1-r, 0\}$ and $C_{l,k+1} \neq 0$ is a constant.

Therefore these functions also satisfy the relation

$$-C_{l,k+1}\psi_{l,k}(r) = \frac{\frac{d}{dr}\psi_{l,k+1}(r)}{r}. \tag{6}$$

For interpolation using a particular Wendland function as the base function, the value of the constant $C_{l,k+1} \neq 0$ is not of importance because the Wendland function appears linearly on both sides of a linear equation. Therefore, one can just fix values that are convenient for the problem at hand and we will do this in the following section. However, when solving collocation problems, we apply a differential operator, see equations (1) and (3), so we get terms involving both the original Wendland function and its derivatives. The derivative of a Wendland function can be written in terms of a lower order Wendland function, using equation (6), and when doing this it is necessary to keep track of the constants $C_{l,k+1}$ for the derivatives. It is only the constant for the base function that can be chosen arbitrarily. After the choice has been made, we must keep track of it through all calculations.

First, we choose a particular function $\psi_{l,k}$ and by abuse of notation we denote it by $\psi_0 = \psi_{l,k}$. Then we define

$$\psi_i(r) = \frac{\frac{d}{dr}\psi_{i-1}(r)}{r} \quad \text{for } i = 1, 2, \ldots, k. \tag{7}$$

The function ψ_i is then a specific Wendland function of order $l, k-i$.
Now the functions

$$\Phi_{l,0}(r) = [(1-r)_+]^l \quad \text{and} \tag{8}$$

$$\Phi_{l,k}(r) = \int_r^1 \Phi_{l,0}(t)t(t^2-r^2)^{k-1}\mathrm{d}t \quad \text{for } k > 0 \tag{9}$$

also satisfy a relation of the form

$$-2(k-1)\Phi_{l,k}(r) = \frac{\frac{d}{dr}\Phi_{l,k+1}(r)}{r},$$

for all integers $k, l \geq 0$, i.e. a relation identical to equation (6) with $C_{l,k+1} = 2(k-1)$. Just note that

$$\frac{d}{dr}\int_r^1 \Phi_{l,0}(t)t(t^2-r^2)^{k-1}\mathrm{d}t = -2r(k-1)\int_r^1 \Phi_{l,0}(t)t(t^2-r^2)^{k-2}\mathrm{d}t.$$

Therefore equation (9) delivers an alternative way to define the Wendland functions, see [26]. Note that [26] uses a different numbering scheme of the functions than we do in this paper.

The Wendland functions have several important properties, cf. e.g. [11, Prop. 3.10]:

1) $\psi_{l,k}(r)$ is a polynomial of degree $l+2k$ for $r \in [0,1]$ and $\operatorname{supp}(\psi_{l,k}) = [0,1]$.
2) The radial function $\Psi(\mathbf{x}) := \psi_{l,k}(\|\mathbf{x}\|)$ is C^{2k} at 0.
3) $\psi_{l,k}$ is C^{k+l-1} at 1.

Frequently we fix the parameter $l := \left\lfloor \frac{n}{2} \right\rfloor + k + 1$, where n is the spacial dimension we are working in, and a constant $c > 0$ to fix the support. By the properties stated above, the radial function $\Psi(\mathbf{x}) := \psi_{l,k}(c\|\mathbf{x}\|)$ is then a C^{2k} function with $\operatorname{supp}(\Psi) = \mathcal{B}^d(0, c^{-1}) \subset \mathbb{R}^n$, where $\mathcal{B}^n(0, c^{-1})$ is the closed n-dimensional ball around the origin with radius c^{-1}.

4 Computing formulas for Wendland functions

In this section we introduce a method to generate Wendland functions of arbitrary degree. As a first step we discuss polynomial representations in software.

4.1 Polynomials Representation

We represent d-degree polynomials $\sum_{i=0}^{d} a_i t^i$ as a list of coefficients $(a_0, a_1, \ldots, a_d)$. Our implementation uses *Python* with *List* objects. Addition and multiplication of polynomials of this form are easily implemented as:

$$\sum_{i=0}^{d_1} a_i t^i + \sum_{j=0}^{d_2} b_j t^j = \sum_{i=0}^{\max\{d_1,d_2\}} (a_i + b_i) t^i,$$

where $a_i = 0$ for $i > d_1$ and $b_j = 0$ for $j > d_2$. Multiplication is given by

$$\left(\sum_{i=0}^{d_1} a_i t^i \right) \left(\sum_{j=0}^{d_2} b_j t^j \right) = \sum_{i=0}^{d_1+d_2} c_i t^i,$$

where

$$c_i = \sum_{k+j=i} a_k b_j.$$

An anti derivative of a polynomial is given by

$$(a_0, a_1, a_2, \ldots, a_d) \mapsto (0, a_0, \frac{a_1}{2}, \frac{a_2}{3}, \ldots, \frac{a_d}{d+1}),$$

corresponding to

$$\int \sum_{i=0}^{d} a_i t^i dt = \sum_{i=0}^{d} \frac{a_i}{i+1} t^{i+1},$$

and differentiation by

$$(a_0, a_1, \ldots, a_d) \mapsto (a_1, 2a_2, 3a_3, \ldots, da_d).$$

In order to maximise exact calculations up to computer limitations, we store the coefficients as tuples of Integers, numerator and denominator, avoiding the floating point approximation. Specifically, we used the *Rational* class provided in *Python*. Polynomials in two variables can be represented as a polynomial in one of the variables, where each coefficient is a polynomial in the second variable, and each of those coefficients is a rational number. This gives us then a list of lists.

4.2 The Method

To calculate a polynomial representing the Wendland function $\psi_{l,k}$ on the interval $[0,1]$ we start by fixing the derivative

$$p'(t) = (1-t)^l t(t^2-r^2)^{k-1},$$

see (9). This function is a polynomial in two variables, which we represent as a polynomial in t where each coefficient is a polynomial in r. Following equation (9), we integrate this function with respect to t, and we obtain a new polynomial $p(t)$ in t, again with coefficients that are polynomials in r. We evaluate the polynomial p at $t=1$ and at $t=r$, which in both cases result in a polynomial in r, and we obtain the polynomial $\psi(r) = p(1) - p(r)$. Note that $\psi(r)$ is a representative of a Wendland function of order l, k, that is $\psi(r) = C_1\psi_{l,k}(r)$ for some constant $C_1 \neq 0$.

We factor the polynomial ψ, using long division, into the form

$$\psi(r) = C_2(1-r)^{l+k}p_{l,k}(r) \tag{10}$$

such that $p_{l,k}(r)$ is a polynomial with co-prime integer coefficients. This is possible since $\psi_{l,0}$ has a zero of order l at 1, and by using the recursive relation in equation (5), we see that $\psi_{l,k}$ has a zero of order $l+k$ at 1. The Wendland function $\psi_{l,k}$ is only defined up to a multiplication by a non-zero constant, therefore we are free to ignore the constant C_2 and use $\psi(r) = (1-r)^{l+k}p_{l,k}(r)$, a polynomial with integer coefficients, as a starting point for our recursion.

Using the relation in (6) and discarding the constant $C_{l,k}$, we see that

$$\begin{aligned}\psi_{l,k-1}(r) &= \frac{\frac{d}{dr}\left[(1-r)^{l+k}p_{l,k}(r)\right]}{r} \\ &= \frac{1}{r}(1-r)^{l+k-1}((1-r)p'_{l,k}(r) - (l+k)p_{l,k}(r)).\end{aligned} \tag{11}$$

Writing the function $\psi_{l,k-1}$, as $\psi_{l,k-1}(r) = (1-r)^{l+k-1}p_{l,k-1}(r)$, then we see

$$\begin{aligned}p_{l,k-1}(r) &:= \frac{\psi_{l,k-1}(r)}{(1-r)^{l+k-1}} \\ &= \frac{1}{r}\left[(1-r)p'_{l,k}(r) - (l+k)p_{l,k}(r)\right].\end{aligned} \tag{12}$$

We know that $\psi_{l,k-1}$ is a polynomial, since it is a Wendland function of order l, k, therefore $\frac{d}{dr}\left[(1-r)^{l+k} p_{l,k}(r)\right]$ must by divisible by the monomial r. Since $(1-r)^{l+k-1}$ is not divisible by r, the right-hand side of (12) must be a polynomial in r. Therefore $p_{l,k-1}$ is a well defined polynomial.

By pulling out the common factor $b_{k-1} \in \mathbb{Z}$ of the coefficients in $p_{l,k-1}$ we obtain a new polynomial $\hat{p}_{l,k-1}$ and a constant b_{k-1} such that

$$p_{l,k-1} = b_{k-1}\hat{p}_{l,k-1}.$$

Repeating this step, until we arrive at $p_{l,0}$, we get a collection of polynomials in the form

$$\psi_i(r) = b_1 \cdots b_i (1-r)^{l+k-i} \hat{p}_{l,k-i}(r),\ i = 1, 2, \ldots, k. \tag{13}$$

where each of the polynomials $\hat{p}_{l,k-i}(r)$ has co-prime integer coefficients and each of the constants b_i is a negative integer.

The above list follows the notation in [11], where ψ_0 is the polynomial given in (10) and is equal to the Wendland function $\psi_{l,k}$, and $\psi_1, \ldots, \psi_i$ are the Wendland functions given by $\psi_{l,k-1}, \ldots, \psi_{l,k-i}$ respectively, see equation (5). It is important to keep track of the constants $b_1, \ldots, b_i$ in (13) as they are necessary for correct evaluation of formula (1).

4.3 Example

We will now demonstrate how the above method determines the Wendland function $\psi_{l,k}$ for $l = 6$ and $k = 4$. Here we start with the function $p'(t) = (1-t)^6 t(t^2 - r^2)^3$ and we obtain

$$\begin{aligned}
\psi(r) &= \int_r^1 (1-t)^6 t(t^2-r^2)^3 dt \\
&= \frac{1}{280}r^{14} - \frac{32}{1001}r^{13} + \frac{1}{8}r^{12} - \frac{64}{231}r^{11} \\
&\quad + \frac{3}{8}r^{10} - \frac{32}{105}r^9 + \frac{1}{8}r^8 - \frac{1}{56}r^6 \\
&\quad + \frac{1}{280}r^4 - \frac{1}{1848}r^2 + \frac{1}{24024} \\
&= \frac{1}{120120}(1-r)^{10}(429r^4 + 450r^3 + 210r^2 + 50r + 1).
\end{aligned}$$

We set $\psi_0(r) = (1-r)^{10}(429r^4 + 450r^3 + 210r^2 + 50r + 1)$. For $r \in [0,1]$ we have the formulas (recall that $\psi_{l,k}(r) = 0$ if $r \notin [0,1]$):

$$\begin{aligned}
\psi_{6,4}(r) &= \psi_0(r) \\
&= (1-r)^{10}(429r^4 + 450r^3 + 210r^2 + 50r + 1); \\
\psi_{6,3}(r) &= \psi_1(r) = \frac{\frac{d}{dr}\psi_0(r)}{r} \\
&= -26(1-r)^9(231r^3 + 159r^2 + 45r + 5); \\
\psi_{6,2}(r) &= \psi_2(r) = \frac{\frac{d}{dr}\psi_1(r)}{r} \\
&= 3,432(1-r)^8(21r^2 + 8r + 1); \\
\psi_{5,1}(r) &= \psi_3(r) = \frac{\frac{d}{dr}\psi_2(r)}{r} \\
&= -102,960(1-r)^7(7r + 1); \\
\psi_{5,0}(r) &= \psi_4(r) = \frac{\frac{d}{dr}\psi_3(r)}{r} \\
&= 5,765,770(1-r)^6.
\end{aligned}$$

Note that we have actually computed a lot more useful information than just a family of Wendland functions $\psi_{5,i}$, $i = 0,1,2,3,4$. In our algorithm, for a fixed l,k, we have

$$\begin{aligned}
\psi_{l,k-j} = \psi_j(r) &= \frac{\frac{d}{dr}\psi_{j-1}(r)}{r} \\
&= \frac{\frac{d}{dr}\psi_{l,k-j+1}(r)}{r}, \quad \text{for } j = 1,\dots,k,
\end{aligned}$$

and we have thus delivered all the radial basis functions needed for a collocation problem. This corresponds to computing a whole table as in [11, Table 3.1], but for a collocation problem with arbitrary high derivatives. The software tool, discussed in Section 6, also includes the constant $c > 0$ in the computations, which is used to fix the support of the Wendland functions.

5 Meshless collocation using Wendland functions

The method of meshless collocation can be used to calculate Lyapunov functions for both deterministic dynamical systems and stochastic dynamical systems. We just choose the operator L and the boundary values appropriately. The next two sections show some of the explicit formulas involved. We also talk about the optimal grid for the interpolation problem, and some aspects of solving the resulting linear systems using software.

5.1 Deterministic systems

Consider an autonomous deterministic system, that is a dynamical system of the form

$$\mathbf{x}'(t) = f(\mathbf{x})$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$, for which the origin is an asymptotically stable equilibrium. We can generate a Lyapunov function, $V : \mathbb{R}^n \to \mathbb{R}$, for this system by solving the interpolation problem (2) with the differential operator L being given by

$$LV(\mathbf{x}) = \langle \nabla V(\mathbf{x}), f(\mathbf{x})\rangle,$$

setting the boundary $\Gamma = \emptyset$ and choosing the function h appropriately. Setting the radial basis function ψ to be the Wendland function $\psi_0(r) = \psi_{l,k}(r)$ for some constants l, k, and then fixing ψ_1 and ψ_2 according to equation (7), the matrix obtained in equation (3) is given by elements of the form (see [11]):

$$\begin{aligned} b_{kl} = {} & \psi_2(\|\mathbf{x}_k - \mathbf{x}_l\|)\langle \mathbf{x}_k - \mathbf{x}_l, f(\mathbf{x}_k)\rangle\langle \mathbf{x}_l - \mathbf{x}_k, f(\mathbf{x}_l)\rangle \\ & - \psi_1(\|\mathbf{x}_k - \mathbf{x}_l\|)\langle f(\mathbf{x}_k), f(\mathbf{x}_l)\rangle. \end{aligned} \tag{14}$$

The components of the vector γ are given by

$$\gamma_j = r(\mathbf{x}_j). \tag{15}$$

Then the solution to the interpolation problem has the formula, see equation (1),

$$V(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k \psi_1(\|\mathbf{x} - \mathbf{x}_k\|)\langle \mathbf{x}_k - \mathbf{x}, f(\mathbf{x}_k)\rangle, \tag{16}$$

where α is the solution of

$$A\alpha = \gamma$$

and ψ_0 and ψ_1 are given by equation (7).

5.2 Stochastic Systems

For SDEs of the form

$$d\mathbf{x}(t) = f(\mathbf{x}(t))dt + g(\mathbf{x}(t))dW(t), \tag{17}$$

$f : \mathbb{R}^d \to \mathbb{R}^d$, $g : \mathbb{R}^d \to \mathbb{R}^{d\times Q}$, we consider the operator L given by the associated generator of the SDE:

$$\begin{aligned} LV(\mathbf{x}) := {} & \nabla V(\mathbf{x}) \cdot f(\mathbf{x}) + \frac{1}{2}\sum_{i,j=1}^{d} [g(\mathbf{x})g(\mathbf{x})^\top]\frac{\partial^2 V}{\partial x_i \partial x_j}(\mathbf{x}) \\ = {} & \nabla V(\mathbf{x}) \cdot f(\mathbf{x}) + \frac{1}{2}\sum_{i,j}^{d} m_{ij}(\mathbf{x})\frac{\partial^2}{\partial x_i \partial x_j}V(\mathbf{x}). \end{aligned} \tag{18}$$

Here $(m_{ij}(\mathbf{x}))_{i,j=1,\dots,d} = g(\mathbf{x})g(\mathbf{x})^\top$, that is $m_{ij}(\mathbf{x}) = \sum_{q=1}^Q g_{iq}(\mathbf{x})g_{jq}(\mathbf{x})$. We choose a Wendland function $\psi_0 = \psi_{l,k}$, for some constants l, k, and set $\psi(\mathbf{x}) = \psi_0(\|\mathbf{x}\|)$. We define ψ_i according to equation (7) for $i = \{1, 2, 3, 4\}$ and we get that the solution to the interpolation problem is given by, see equation (1),

$$\begin{aligned} V(\mathbf{x}) = \sum_{k=1}^N \alpha_k \Big[&- \psi_1(\|\mathbf{x}-\mathbf{x}_k\|)\langle \mathbf{x}-\mathbf{x}_k, f(\mathbf{x}_k)\rangle \\ &+ \frac{1}{2}\sum_{i,j}^d m_{ij}(\mathbf{x}_k)[\psi_2(\|\mathbf{x}-\mathbf{x}_k\|)(\mathbf{x}-\mathbf{x}_k)_i(\mathbf{x}-\mathbf{x}_k)_j + \delta_{i,j}\psi_1(\|\mathbf{x}-\mathbf{x}_k\|)]\Big] \\ &+ \sum_{k=1}^M \alpha_{N+k}\psi_0(\|\mathbf{x}-\xi_k\|). \end{aligned} \tag{19}$$

In this formula the vector α is the solution to the linear system in equation (2). The formulas for the matrix elements are

$$\begin{aligned} d_{kl} &= \psi_0(\|\xi_k - \xi_l\|), \\ c_{kl} &= -\psi_1(\|\xi_l - \mathbf{x}_k\|)\langle \xi_l - \mathbf{x}_k, f(\mathbf{x}_k)\rangle \\ &\quad + \frac{1}{2}\sum_{i,j=1}^d m_{ij}(\mathbf{x}_k)[\psi_2(\|\xi_l - \mathbf{x}_k\|)(\xi_l - \mathbf{x}_k)_i(\xi_l - \mathbf{x}_k)_j \\ &\quad + \delta_{ij}\psi_1(\|\xi_l - \mathbf{x}_k\|)], \end{aligned} \tag{20}$$

and, abbreviating $\beta = \mathbf{x} - \mathbf{x}_k$,

$$\begin{aligned} b_{kl} = &-\psi_2(\|\beta\|)\langle \beta, f(\mathbf{x}_k)\rangle\langle \beta, f(\mathbf{x}_l)\rangle - \psi_1(\|\beta\|)\langle f(\mathbf{x}_k), f(\mathbf{x}_l)\rangle \\ &+ \frac{1}{2}\sum_{i,j=1}^d m_{ij}(\mathbf{x}_l)\Big[\psi_3(\|\beta\|)\langle \beta, f(\mathbf{x}_k)\rangle\beta_i\beta_j + \psi_2(\|\ \beta\|)f_j(\mathbf{x}_k)\beta_i \\ &\qquad + \psi_2(\|\beta\|)f_i(\mathbf{x}_k)\beta_j + \delta_{ij}\psi_2(\|\beta\|)\langle \beta, f(\mathbf{x}_k)\rangle\Big] \\ &+ \frac{1}{2}\sum_{i,j=1}^d m_{ij}(\mathbf{x}_k)\Big[-\psi_3(\|\beta\|)\langle \beta, f(\mathbf{x}_l)\rangle\beta_i\beta_j - \psi_2(\|\beta\|)f_j(\mathbf{x}_l)\beta_i \\ &\qquad - \psi_2(\|\beta\|)f_i(\mathbf{x}_l)\beta_j - \delta_{ij}\psi_2(\|\beta\|)\langle \beta, f(\mathbf{x}_l)\rangle\Big] \\ &+ \frac{1}{4}\sum_{r,s=1}^d\sum_{i,j=1}^d m_{rs}(\mathbf{x}_k)m_{ij}(\mathbf{x}_l)\Big[\psi_4(\|\beta\|)\beta_i\beta_j\beta_r\beta_s \\ &\qquad + \psi_3(\|\beta\|)[\delta_{ij}\beta_r\beta_s + \delta_{ir}\beta_j\beta_s + \delta_{is}\beta_j\beta_r \\ &\qquad\qquad + \delta_{jr}\beta_i\beta_s + \delta_{js}\beta_i\beta_r + \delta_{rs}\beta_i\beta_j] \\ &\qquad + \psi_2(\|\beta\|)[\delta_{ij}\delta_{rs} + \delta_{ir}\delta_{js} + \delta_{is}\delta_{jr}]\Big]. \end{aligned} \tag{21}$$

5.3 Generating the grid

The optimal grid in $\Omega \subset \mathbb{R}^n$ for the interpolation problem (2) was studied in [17]. The grid that delivers the smallest fill-distance, the parameter which determines the accuracy of the solution, is defined using the basis vectors $w_1, w_2, \ldots, w_n \in \mathbb{R}^n$, where (the e_is denote the usual orthonormal basis in $\mathbb{R}^n$)

$$w_k = \sum_{j=1}^{k-1} \epsilon_j e_j + (k+1)\epsilon_k e_k \quad \text{and} \quad \epsilon_k = \frac{1}{\sqrt{2k(k+1)}}.$$

The grid-points $G_{\alpha,z} = \{g_{\mathbf{i}} : \mathbf{i} \in \mathbb{Z}^n\} \subset \mathbb{R}^n$ with fill=distance parameter $\alpha > 0$ and offset $z \in \mathbb{R}^n$ are then given by

$$g_{\mathbf{i}} := z + \alpha \sum_{k=1}^{n} i_k w_k, \quad \mathbf{i} = (i_1, i_2, \ldots, i_n) \in \mathbb{Z}^n.$$

Given two vectors $a, b \in \mathbb{R}^n$ such that $a_i < b_i$ for $i = 1, 2, \ldots, n$, we want to compute the coordinates of the grid-points $g_{\mathbf{i}} \in G_{\alpha,z}$ that are in the cube

$$C_{a,b} := [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n].$$

By observing that w_n is the only basis vector with a nonzero entry in its last component, w_{n-1} and w_n are the only basis vectors with nonzero entries in their second to last component, etc. , these can be computed efficiently in a recursive manner. Let us illustrate this with $n = 3$, the general strategy can be read from the code below.

Given the offset vector $z = (z_1, z_2, z_3) \in \mathbb{R}^3$ and $a_3 < b_3$, we see that only those $\mathbf{i} = (i_1, i_2, i_3) \in \mathbb{Z}^3$ with i_3 fulfilling

$$a_3 \leq z_3 + i_3 \cdot \alpha \cdot (3+1) \frac{1}{\sqrt{2 \cdot 3 \cdot (3+1)}} = z_3 + i_3 \cdot \alpha \sqrt{\frac{3+1}{2 \cdot 3}} \leq b_3,$$

i.e.

$$\left\lceil \frac{a_3 - z_3}{\alpha} \sqrt{\frac{2 \cdot 3}{3+1}} \right\rceil \leq i_3 \leq \left\lfloor \frac{b_3 - z_3}{\alpha} \sqrt{\frac{2 \cdot 3}{3+1}} \right\rfloor, \tag{22}$$

have to be considered, because all other choices of i_3 deliver an entry in the third component that is not in the interval $[a_3, b_3]$. For each i_3 fulfilling this inequality, let us denote it i_3^*, we can generate appropriate i_2 components by observing that $g_{\mathbf{i}}$ with $\mathbf{i} = (0, 0, i_3^*)$ has the entry

$$z_2^* := z_2 + \alpha \cdot i_3^* \cdot \epsilon_2 = z_2 + \alpha \cdot i_3^* \cdot \frac{1}{\sqrt{2 \cdot 2 \cdot (2+1)}}$$

in its second component. The appropriate i_2s for i_3^* are thus given by considering the formula for w_2 and are easily seen to fulfill

$$a_2 \leq z_2^* + \alpha \cdot i_2 \cdot (2+1)\epsilon_2 = z_2 + \alpha \cdot i_3^* \cdot \epsilon_2 + \alpha \cdot i_2 \cdot (2+1)\epsilon_2 \leq b_2,$$

which can be written similarly to before as

$$\left\lceil \frac{a_2 - z_2^*}{\alpha}\sqrt{\frac{2\cdot 2}{2+1}} \right\rceil \leq i_2 \leq \left\lfloor \frac{b_2 - z_2^*}{\alpha}\sqrt{\frac{2\cdot 2}{2+1}} \right\rfloor . \tag{23}$$

Now, having fixed an i_3^* fulfilling (22) and subsequently an i_2^* for this i_3^* fulfilling (23), we can in a similar manner compute appropriate i_1s. The vector $g_{\mathbf{i}}$ with $\mathbf{i} = (0, i_2^*, i_3^*)$ has the entry

$$z_1^* := z_1 + \alpha \cdot (i_2^*\epsilon_1 + i_3^*\epsilon_1) = z_1 + \alpha \cdot (i_2^* + i_3^*) \cdot \frac{1}{\sqrt{2\cdot 1\cdot (1+1)}}$$

in its first component. Similarly to before the appropriate i_1s in (i_1, i_2^*, i_3^*) are read from an inequality:

$$a_1 \leq z_1^* + \alpha \cdot i_1 \cdot (1+1)\epsilon_1 = z_1 + \alpha \cdot (i_2^* + i_3^*) \cdot \frac{1}{\sqrt{2\cdot 1\cdot (1+1)}} + \alpha \cdot i_1 \cdot (1+1)\epsilon_1 \leq b_1$$

or

$$\left\lceil \frac{a_1 - z_1^*}{\alpha}\sqrt{\frac{2\cdot 1}{1+1}} \right\rceil \leq i_1 \leq \left\lfloor \frac{b_1 - z_1^*}{\alpha}\sqrt{\frac{2\cdot 1}{1+1}} \right\rfloor .$$

This recursive procedure computes all grid vectors $g_{\mathbf{i}} \in G_{\alpha,z}$ in the cube $C_{a,b}$ and is implemented in C++ using the Armadillo library [21] is given in Listing 1.1.

Listing 1.1. Code in C++ that generates the optimal grid

```
list<arma::vec> HexaGridnew(arma::vec a, arma::vec b, ...
    double c, int N) {
    a = a(span(0, N - 1));
    b = b(span(0, N - 1));
    double tol = 1e-10; // add a small tolerance to the cube
    a -= tol*ones<vec>(N);
    b += tol*ones<vec>(N);
    list<vec> Ret;
    unsigned int i, k;
    vec e(N, fill::zeros);

    for (k = 1; k <= N; k++) {
        e(k - 1) = sqrt(1.0 / (2.0*k*(k + 1)));
    }
    vector<vec> w(N);
    for (i = 1; i <= N; i++) {
        vec v(N, fill::zeros);
        for (int k = 1; k < i; k++) {
            v(k - 1) = e(k - 1);
        }
        v(i - 1) = (i + 1)*e(i - 1);
        w[i - 1] = v;
    }

    function<void(int, vec)> ML = [&](int r, vec x) {
        for (int i = int(ceil((a(r) - x(r)) / (c*(r + ...
            2)*e(r)))); i <= int(floor((b(r) - x(r)) / ...
            (c*(r + 2)*e(r)))); i++) {
            if (r == 0) {
                Ret.push_back(x+i*c*w[r]);
            }
            else {
                ML(r - 1, x + i*c*w[r]);
            }
        }
    };
    ML(N - 1, 0.5*c*w[N-1]);
    return Ret;
}
```

5.4 Solving the linear system

The linear system of equations that we obtain when solving the interpolation problem is defined by a symmetric and positive definite matrix A, see equation (3). LAPACK [1] has specific methods for solving these types of equations, that use the Cholesky decomposition of the matrix $A = U^\top U$, where U is upper triangular with positive diagonal entries. The function DPOSV overwrites the contents of the matrix A with the Cholesky decomposition U and solves the system $A\alpha = \gamma$. The acronym is understood in the following way, D stands for Double, PO stands for Symmetric or Hermitian positive definite and SV stands for solve. This has much better numerical properties than solving the system with e.g. LU-decomposition.

It is also possible to store the matrix A in packed format, that is, since A is a symmetric matrix, we can store just the upper triangular part of it. This saves a considerable amount of memory. LAPACK has functions for computing the Cholesky decomposition of the matrix A in packed format. The function DPPTRF calculates the Cholesky decomposition of A in packed format, overwriting the contents of matrix A, and the function DPPTRS solves then the system $A\alpha = \gamma$ using the Cholesky factor computed by DPPTRF. Here the letters PP stand for Symmetric or Hermitian positive definite in packed storage, TRF means factorize to a product of triangular matrices, and TRS stands for solving the factorized system using forward or backwards substitution.

For ease of usage we have implemented functions that calculate the interpolation matrices described before, i.e. equations (3),(14),(20) and (21), for both stochastic and deterministic dynamical systems. These are available in the software repository.

6 Software library

We have implemented the algorithm described in Section 4.2 in a software tool[1] that generates C/C++ code versions of the Wendland functions in factorized form. In a previous work [3], we determined that the most efficient and accurate way to evaluate these Wendland functions was to use this factorized form. Evaluating these polynomials in fully expanded format using Horner's scheme [8], can lead to very large numerical errors as shown in [3]. We give a brief summary of these results in a later section and in Table 1. Below in Listing 1.2 is a part of the library generated by our tool, which shows the family of Wendland functions obtained when starting with $\Psi_0(\mathbf{x}) = \psi_{5,4}(c\|\mathbf{x}\|)$, where $c > 0$ is the constant that controls the support of the radial function Ψ.

Listing 1.2. Generated code for the $\psi_{6,4}$ family

```
double __wendlandpsi_6_4_0(double x, double c){
    double t=__ipow((1.0-x),10);
    t=1.0*t*(((((429)*x + 450)*x + 210)*x + 50)*x + 5);
    return t;
}
double __wendlandpsi_6_4_1(double x, double c){
    double t=__ipow((1.0-x),9);
    t=-26.0*t*__ipow(c,2)*((((231)*x + 159)*x + 45)*x + 5);
    return t;
}
double __wendlandpsi_6_4_2(double x, double c){
    double t=__ipow((1.0-x),8);
    t=3432.0*t*__ipow(c,4)*(((21)*x + 8)*x + 1);
    return t;
}
double __wendlandpsi_6_4_3(double x, double c){
    double t=__ipow((1.0-x),7);
    t=-102960.0*t*__ipow(c,6)*((7)*x + 1);
    return t;
}
double __wendlandpsi_6_4_4(double x, double c){
    double t=__ipow((1.0-x),6);
    t=5765760.0*t*__ipow(c,8)*(1);
    return t;
}
```

Note that `__wendlandpsi_6_4_j` corresponds to ψ_j in the example, but with $x = cr$ as argument.

When starting with $\Psi_0(x) = \psi_{6,3}(c\|x\|)$ instead, the relevant definitions are given in Listing 1.3.

[1] The tool is available at https://gitlab.com/hjortur/wendland-function-generator/ with example outputs.

Listing 1.3. Generated code for the $\psi_{6,3}$ family

```
double __wendlandpsi_6_3_0(double x, double c){
    double t=__ipow((1.0-x),9);
    t=1.0*t*((((231)*x + 159)*x + 45)*x + 5);
    return t;
}
double __wendlandpsi_6_3_1(double x, double c){
    double t=__ipow((1.0-x),8);
    t=-132.0*t*__ipow(c,2)*(((21)*x + 8)*x + 1);
    return t;
}
double __wendlandpsi_6_3_2(double x, double c){
    double t=__ipow((1.0-x),7);
    t=3960.0*t*__ipow(c,4)*((7)*x + 1);
    return t;
}
double __wendlandpsi_6_3_3(double x, double c){
    double t=__ipow((1.0-x),6);
    t=-221760.0*t*__ipow(c,6)*(1);
    return t;
}
```

Note that the polynomials `__wendlandpsi_6_3_1` and `__wendlandpsi_6_4_2` differ only by a multiplication of a constant and a power of c, and both polynomials are a representative of the Wendland function $\psi_{6,2}$.

The function `__ipow(x,i)` evaluates x^i where x is a double and i is a positive integer. We have "flattened" the functions `__wendlandpsi_x_y_z` in the sense that their domain is $[0, 1]$. They require the user to pre-multiply the x value with the chosen RBF-constant $c > 0$, that is for $\Psi(\mathbf{x}) = \psi_{l,k}(c\|\mathbf{x}\|)$, the user needs to pass in the value $c\|\mathbf{x}\|$ and c after ensuring that $c\|\mathbf{x}\| \in [0, 1]$. A possible implementation using the Armadillo library [21] can be see in listing 1.4.

Listing 1.4. Example usage

```
double psi3(const arma::vec &x, double c){
    double cx=c*arma::norm(x,2);
    return ( cx < 1.0 ) ? __wendlandpsi_6_4_3(cx,c) : 0.0;
}
```

The tool is a simple Python script named *wendlandfunctions.py*. When the script is run it outputs text for code- and header-files, which contain the Wendland function definitions. The user can supply the script with a parameter `--l` and an integer value $m \geq 2$, in order to output code for Wendland functions from $\psi_{2,1}$ up to $\psi_{m,i}$ for all $0 \leq i < m$.

6.1 Example Lyapunov functions

Included in the repository are example outputs and example programs that calculate Lyapunov functions for deterministic and stochastic systems, using the software library that our tool generates and functions that generate the optimal interpolation grid and the interpolation matrices. Figures 1, 2, 3, and 4 show graphs of Lyapunov functions obtained from these example programs, where the systems considered are:

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ -x(t) - (1 - x(t)^2)y(t) \end{bmatrix}, \tag{24}$$

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ -x(t) + \frac{1}{3}x(t)^3 - y(t) \end{bmatrix}, \tag{25}$$

and the stochastic systems

$$dx = \sin(x)dt + \frac{3x}{1+x^2}dW, \tag{26}$$

$$d\mathbf{x} = \begin{bmatrix} \|\mathbf{x}\| - 1.0 & 1.0 \\ -1.0 & \|\mathbf{x}\| - 1.0 \end{bmatrix} \mathbf{x}dt + \|\mathbf{x}\|(\|\mathbf{x}\| - 0.5)(\|\mathbf{x}\| - 1.5)\,\mathbf{x}\,dW. \tag{27}$$

For the systems in the above equations we have used the optimal grid as described in Section 5.3 as the collocation/interpolation points. Denoting by $\mathcal{B}^2(\mathbf{x}, r)$ the 2 dimensional open ball around $\mathbf{x}$ with radius r and

$$\Gamma_r(N) = \left\{ r\left(\cos\left(\frac{j2\pi}{N}\right), \sin\left(\frac{j2\pi}{N}\right)\right) \quad | \quad j \in \{1, \dots, N\} \right\} \subset \mathbb{R}^2,$$

we used the following data to calculate the Lyapunov functions:

- Collocation grid on $[-2, 2] \times [-2, 2] \setminus \mathcal{B}^2(0, 0.1)$ with fill-distance parameter $\alpha = \frac{4}{15}$, $LV(\mathbf{x}) = -\|\mathbf{x}\|$, and $\psi_0(\mathbf{x}) = \psi_{6,4}(\|\mathbf{x}\|)$ for the system in equation (24);
- Collocation grid on $[-1.4, 1.4] \times [-1.4, 1.4] \setminus \mathcal{B}^2(0, 0.1)$ with fill distance parameter $\alpha = \frac{2.8}{20}$, $LV(\mathbf{x}) = -\|\mathbf{x}\|$ and $\psi_0(\mathbf{x}) = \psi_{5,3}(\|\mathbf{x}\|)$ for the system in equation (25);
- Collocation grid on $[0.1, 8.0]$ with fill distance parameter $\alpha = \frac{1}{400}$, $LV(x) = 10^{-4}$, $V(0.1) = 0$, $V(8.0) = 1.0$ and $\psi_0(\mathbf{x}) = \psi_{7,6}(2\|\mathbf{x}\|)$ for the system in equation (26);
- Collocation grid on $[-2, 2] \times [-2, 2] \setminus \mathcal{B}^2(0, 0.4)$ with fill distance parameter $\alpha = \frac{1}{25}$, $LV(\mathbf{x}) = 10^{-2}$, $V(\xi_j) = 0$ and $V(\beta_j) = 1$ for all $\xi_j \in \Gamma_{0.4}(4)$ and all $\beta_j \in \Gamma_{1.9}(80)$. Furthermore we set $\psi_0(\mathbf{x}) = \psi_{6,4}(\|\mathbf{x}\|)$, for the system in equation (27).

Note that the resulting Lyapunov functions for the systems in equations (26) and (27) in figures 3 and 4 are comparable to the results obtained in [5] and [15].

6.2 Comparison of evaluation methods

In the paper [3] we compared different methods of evaluation for Wendland functions at a point. The methods used where:

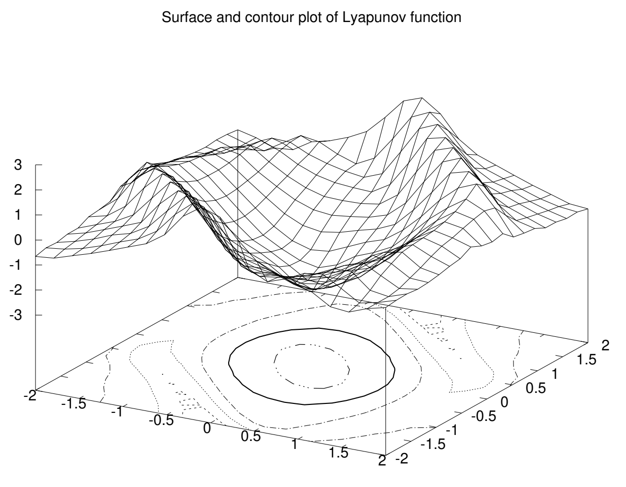

Fig. 1. Lyapunov function for the system in equation (24)

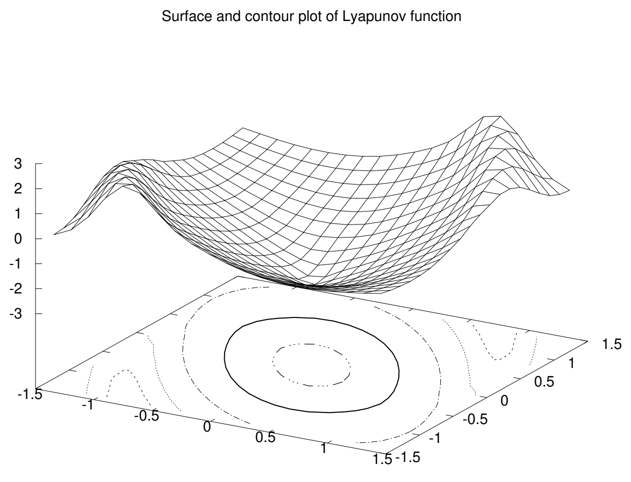

Fig. 2. Lyapunov function for the system in equation (25)

- Having them in factorized form, as our software tool provides, see Listing 1.2;
- Fully expanded polynomials and evaluated using Horner's Scheme, as in [2];
- Pre-computing the function in high precision (see below) at 10^7 evenly spaced points on the interval $[0, 1]$ and using them as a lookup table. That is, round to the closest value;

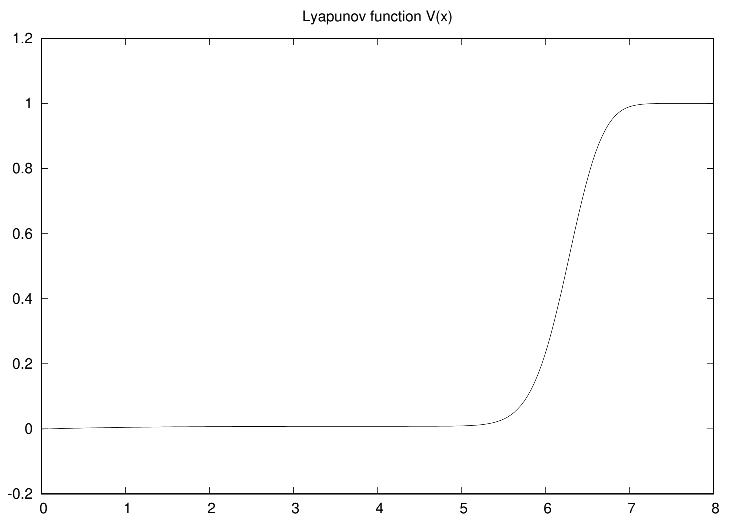

Fig. 3. Lyapunov function for the stochastic system in equation (26)

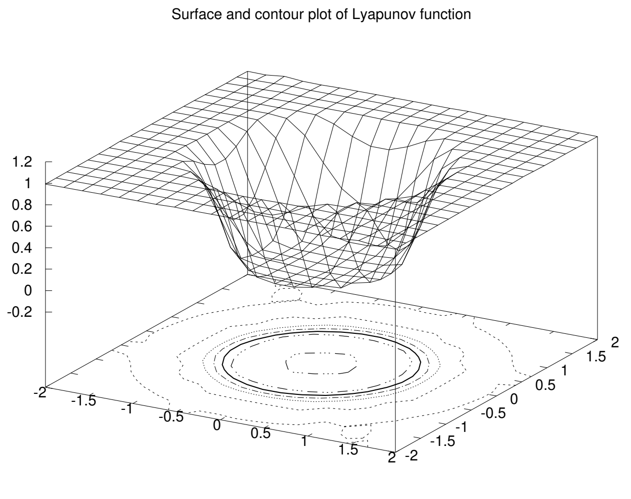

Fig. 4. Lyapunov function for the stochastic system in equation (27)

– Using the same lookup table but additionally linearly interpolate between two nearest neighbours to improve accuracy.

Table 1 shows time elapsed to evaluate the Wendland function $\psi_{7,2}$ at 10^7 different points on the interval $[0, 1]$, and the scale of the relative error obtained on this interval. For further analysis see [3].

Method / Processor	i5-8250U	i7-4790K	Rel.error
Factorized form	171.5ms	107ms	10^{-13}
Horner's scheme	548.1ms	395ms	1
Lookup table	125.8ms	105ms	10^{-5}
Lookup table with interpolation	165.5ms	128ms	10^{-9}

Table 1. Evaluation of $\psi_{7,2}$ at 10^7 different points, for different CPUs

7 Conclusion

In this paper we have presented a software tool for generating Wendland's compactly supported Radial Basis Functions in an optimal form. This tool generates a C/C++ library for Wendland functions of arbitrary degree in factorized form. Furthermore, this tool generates an entire family of these functions, used for solving collocation problesm, for each initial Wendland function $\psi_{l,k}$. We have also presented an algorithm that this software tool uses for generating Wendland functions in this factorized form, for accurate and efficient evaluations. Finally, we have created a software library for calculating Lyapunov functions for both stochastic and deterministic dynamical systems, using these factorized Wendland functions that our tool generates. All the software, with example usage, is available for download at https://gitlab.com/hjortur/wendland-function-generator/.

Acknowledgement: This research was supported by the Icelandic Research Fund (Rannis), grant number 152429-051, Lyapunov Methods and Stochastic Stability.

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Society for Industrial and Applied Mathematics, 3 edn. (1999)
2. Argaez, C., Hafstein, S., Giesl, P.: Wendland functions - a C++ code to compute them. In: Proceedings of the 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications - Volume 1: SIMULTECH,. pp. 323–330. INSTICC, SciTePress (2017)
3. Bjornsson, H., Hafstein, S.: Verification of a numerical solution to a collocation problem. In: Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics - Volume 1: CTDE,. pp. 587–594. INSTICC, SciTePress (2018)
4. Bjornsson, H., Hafstein, S.: Algorithm and software to generate code for Wendland functions in factorized form. In: Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics. pp. 156–162. INSTICC, SciTePress (2019)
5. Bjornsson, H., Hafstein, S., Giesl, P., Gudmundsson, S., Scalas, E.: Computation of the stochastic basin of attraction by rigorous construction of a Lyapunov function. Discrete and Continuous Dynamical Systems-Series B **24**(8), 4247–4269 (2019)

6. Buhmann, M.: Radial basis functions. In: Acta numerica, 2000, Acta Numer., vol. 9, pp. 1–38. Cambridge Univ. Press, Cambridge (2000)
7. Buhmann, M.: Radial basis functions: theory and implementations, Cambridge Monographs on Applied and Computational Mathematics, vol. 12. Cambridge University Press, Cambridge (2003)
8. Burrus, C.S., Fox, J.W., Sitton, G.A., Treitel, S.: Horner's method for evaluating and deflating polynomials. DSP Software Notes, Rice University, Nov **26** (2003)
9. Floater, M., Iske, A.: Multistep scattered data interpolation using compactly supported Radial Basis Functions. J. Comput. Appl. Math. **73**(1-2), 65–78 (1996)
10. Franke, C., Schaback, R.: Solving partial differential equations by collocation using radial basis functions. Appl. Math. Comput. **93**(1), 73–82 (1998)
11. Giesl, P.: Construction of Global Lyapunov Functions Using Radial Basis Functions, Lecture Notes in Mathematics, vol. 1904. Springer-Verlag, Berlin (2007)
12. Giesl, P.: Construction of a local and global Lyapunov function using radial basis functions. IMA J. Appl. Math. **73**(5), 782–802 (2008)
13. Giesl, P.: Construction of a finite-time Lyapunov function by meshless collocation. Discrete Contin. Dyn. Syst. Ser. B **17**(7), 2387–2412 (2012)
14. Giesl, P., Hafstein: Computation and verification of Lyapunov functions. SIAM Journal on Applied Dynamical Systems **14**(4), 1663–1698 (2015)
15. Grüne, L., Camilli, F.: Characterizing attraction probabilities via the stochastic Zubov equation. Discrete Contin. Dyn. Syst. Ser. B **3**(3), 457–468 (2003)
16. Hafstein, S., Gudmundsson, S., Giesl, P., Scalas, E.: Lyapunov function computation for autonomous linear stochastic differential equations using sum-of-squares programming. Discrete and Continuous Dynamical Systems - Series B **23**(2), 939–950 (2018)
17. Iske, A.: Perfect centre placement for radial basis function methods. Tech. Rep. TUM-M9809, TU Munich, Germany (1998)
18. Khalil, H.: Nonlinear systems. Pear, 3. edn. (2002)
19. Khasminskii, R.: Stochastic stability of differential equations. Springer, 2nd edn. (2012)
20. Mao, X.: Stochastic Differential Equations and Applications. Woodhead Publishing, 2nd edn. (2008)
21. Sanderson, C.: Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. Tech. rep., NICTA. (2010)
22. Sastry, S.: Nonlinear Systems: Analysis, Stability, and Control. Springer (1999)
23. Vidyasagar, M.: Nonlinear System Analysis. Classics in applied mathematics, SIAM, 2. edn. (2002)
24. Wendland, H.: Error estimates for interpolation by compactly supported Radial Basis Functions of minimal degree. J. Approx. Theory **93**, 258–272 (1998)
25. Wendland, H.: Scattered data approximation, Cambridge Monographs on Applied and Computational Mathematics, vol. 17. Cambridge University Press, Cambridge (2005)
26. Wendland, H.: Multiscale radial basis functions. In: Frames and other bases in abstract and function spaces, pp. 265–299. Appl. Numer. Harmon. Anal., Birkhäuser/Springer, Cham (2017)
27. Wu, Z.: Hermite-Birkhoff interpolation of scattered data by radial basis functions. Approx. Theory Appl. **8**(2), 1–10 (1992)