Actually Sparse Variational Gaussian Processes

Jake Cunningham[*]
Centre for Artificial Intelligence
University College London

So Takao
Centre for Artificial Intelligence
University College London

Mark van der Wilk
Imperial College London

Marc Peter Deisenroth
Centre for Artificial Intelligence
University College London

Abstract

In this work, we propose a new class of inter-domain variational Gaussian process, constructed by projecting onto a set of compactly supported B-Spline basis functions. Our model is akin to variational Fourier features. However, due to the compact support of the B-Spline basis, we produce sparse covariance matrices. This enables us to make use of sparse linear algebra to efficiently compute matrix operations. After a one-off pre-computation, we show that our method reduces both the memory requirement and the per-iteration computational complexity to linear in the number of inducing points.

1 Introduction

Gaussian processes (GPs) scale infamously as $\mathcal{O}(N^3)$ in computational complexity and $\mathcal{O}(N^2)$ in memory, where N is the number of training inputs, making them unfeasible for large datasets. To overcome this limitation, so-called sparse GPs condition on a set of $M \ll N$ inducing points $\mathbf{u}$, which efficiently represent the input data (see [10] for a review). Amongst these methods, variational approximations have proved popular [12, 5, 6, 4, 11, 3]. Introduced by [12], Sparse Variational Gaussian processes (SVGP) work by minimising the Kullback-Leibler (KL) divergence between the approximate and the true posterior, allowing us to learn the model parameters via gradient descent. The resulting approximation scales as $\mathcal{O}(NM^2 + M^3)$ in computational complexity and $\mathcal{O}(NM)$ in memory, which practically limits these methods to $\sim$10,000 inducing points.

Inter-domain GPs [13, 8] generalize the idea of inducing variables, allowing for more expressive features and computationally efficient matrix algebra. Variational Fourier Features (VFFs) [4] constructs inter-domain inducing features by projecting the GP onto a Fourier basis using the RKHS inner product. This results in inducing features that span the width of the domain and thus have global influence on the prediction. Further, the inducing variables are almost independent, providing computationally efficient block-diagonal covariance matrices. In one dimension, this can be exploited to reduce the computational complexity to $\mathcal{O}(M^3)$ after an initial one-off pre-computation of $\mathcal{O}(NM^2)$.

However, like SVGP, for large numbers of inducing points, VFF also becomes intractable. Variational Inducing Spherical Harmonics (VISH) [3] improves upon VFF by first projecting the data onto the unit hypersphere and then using a basis of spherical harmonics as inter-domain inducing features. As the basis functions are orthogonal, this reduces the cost of matrix inversion to $\mathcal{O}(M)$ and the total cost of inference to $\mathcal{O}(N_b M^2)$ when using minibatching [5].

In this work, we wish to improve upon VFF for fast large-scale regression. We define a new inter-domain approximation by projecting onto a basis of compactly supported B-splines. Due to the

[*]Correspondence to `jake.cunningham.21@ucl.ac.uk`

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

compact support of the B-spline basis, our method results in sparse band-diagonal covariance matrices. This allows us to use operations from sparse linear algebra to reduce the cost of precomputation to linear in the number of data points and the per-iteration cost to linear in the number of inducing points, enabling us to scale to both large numbers of datapoints and large numbers of inducing points.

2 Background

2.1 Sparse Variational Gaussian Processes (SVGP)

Let $f \sim \mathcal{GP}(0, k(\cdot,\cdot))$. Given a dataset $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, SVGP inference [5, 12] approximates the GP posterior $p(f|\mathbf{y})$ in terms of a set of M inducing features $\mathbf{u} = \{f(\mathbf{z}_m)\}_{m=1}^M$, where $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$ are input locations that may be learned or fixed. At inference, the approximate posterior is a GP

$$\mathcal{GP}\left(\mathbf{k_u}(\cdot)^\top \mathbf{K_{uu}^{-1}} \mathbf{m},\ k(\cdot,\cdot) + \mathbf{k_u}(\cdot)^\top \mathbf{K_{uu}^{-1}} (\mathbf{S} - \mathbf{K_{uu}}) \mathbf{K_{uu}^{-1}} \mathbf{k_u}(\cdot)\right), \tag{1}$$

where $\mathbf{K_{uu}} := [k(\mathbf{z}_m, \mathbf{z}_{m'})]_{m,m'=1}^M = [\mathrm{Cov}(\mathbf{u}_m, \mathbf{u}_{m'})]_{m,m'=1}^M$ is the cross-covariance of the inducing features, $\mathbf{k_u}(\cdot) := [k(\mathbf{z}_m, \cdot)]_{m=1}^M = [\mathrm{Cov}(\mathbf{u}_m, f(\cdot))]_{m=1}^M$ is the vector of feature maps and $\mathbf{m}, \mathbf{S}$ are free variational parameters. The variational parameters $\mathbf{Z}$, $\mathbf{m}$ and $\mathbf{S}$ are learnt by maximising the Evidence Lower Bound (ELBO)

$$\mathrm{ELBO} \leq \log p(\mathbf{y}) - \mathrm{KL}\left[q(f) \,\|\, p(f|\mathbf{y})\right], \tag{2}$$

which minimises the Kullback-Leibler (KL) divergence $\mathrm{KL}\left[q(f) \,\|\, p(f|\mathbf{y})\right]$ between the approximate and true posterior. This is often learnt together with the kernel hyperparemeters. For the case of a Gaussian likelihood, the ELBO is analytically given by

$$\mathrm{ELBO} = \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{K_{uf}} + \sigma_n^2 \mathbf{I}\right) - \frac{1}{2}\sigma_n^{-2} \mathrm{tr}\left(\mathbf{K_{ff}} - \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{K_{uf}}\right), \tag{3}$$

where $\mathbf{K_{ff}} := [k(\mathbf{x}_n, \mathbf{x}_{n'})]_{n,n'=1}^N$ and $\mathbf{K_{uf}} := [\mathbf{k_u}(\mathbf{x}_n)]_{n=1}^N$.

2.2 Variational Fourier Features (VFF)

VFF [4] is an inter-domain variational GP approximation that constructs inducing features as a Matérn RKHS projection of the GP onto a set of Fourier basis functions $u_m = \langle f, \phi_m \rangle_{\mathcal{H}}, \quad m = 1, \ldots, M$, where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the Matérn RKHS inner product and $\phi_0(x) = 1$, $\phi_{2i-1}(x) = \cos(\omega_i x)$, $\phi_{2i} = \sin(\omega_i x)$ are the Fourier basis functions. This results in the matrices

$$\mathbf{K_{uu}} = [\langle \phi_i, \phi_j \rangle_{\mathcal{H}}]_{i,j=1}^M \quad \text{and} \quad \mathbf{k_u}(x) = [\phi_i(x)]_{i=1}^M, \tag{4}$$

where, due to the reproducing property, the feature vector $\mathbf{k_u}(x)$ are elements of the Fourier basis and are independent of kernel hyperparameters. This leads to several computational benefits. Firstly, we can precompute $\mathbf{k_u}(x)$ as it remains constant throughout training. Secondly, due to the orthogonality of the Fourier basis, $\mathbf{K_{uu}}$ is the sum of a block diagonal matrix plus low rank matrices. This structure can be exploited to significantly reduce the computational complexity, resulting in orders of magnitude speed up in training and prediction when compared to standard sparse GP methods.

However, VFF has two main flaws. Firstly, VFF generalises poorly to higher dimensions due to the use of a Kronecker product basis. Secondly, whilst $\mathbf{K_{uu}}$ has a computationally efficient structure, $\mathbf{K_{uf}}$ is still a dense matrix. Thus, for example in the special case when the likelihood is Gaussian, we are still required to compute a dense Cholesky factor of the $M \times M$ matrix $\mathbf{K_{uu}} + \sigma^{-2} \mathbf{K_{uf}} \mathbf{K_{fu}}$, which has a per iteration cost of $\mathcal{O}(M^3)$. For the non-Gaussian case, the computational cost is dominated by the dense matrix multiplication $\mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{S}$ costing $\mathcal{O}(NM^2)$ per iteration, although this can be reduced to $\mathcal{O}(N_b M^2)$ using stochastic optimization where N_b is the size of the minibatch.

3 Actually Sparse Variational Gaussian Process

In this section, we propose Actually Sparse Variational Gaussian Processes (AS-VGP). The core idea is to utilise the same RKHS projections defined in VFF, except to project a GP onto a set of compactly supported *B-spline basis functions*, instead of the Fourier basis functions. Unlike in VFF, the resulting inducing features $\{u_m\}_{m=1}^M$ are highly localised by the nature of their compact support, resulting in both $\mathbf{K_{uu}}$ and $\mathbf{K_{uf}}$ becoming sparse matrices. We will see that these covariance structures allows us to gain further computational benefits.

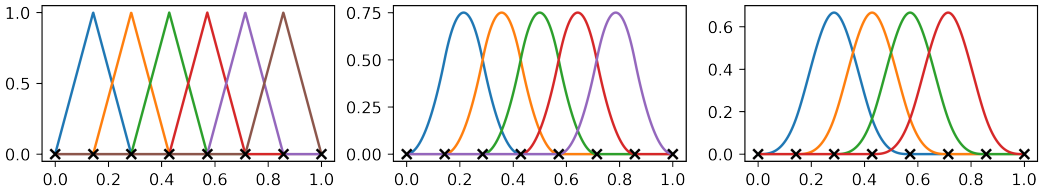

Figure 1: a) 1st order B-spline basis b) 2nd order B-spline basis c) 3rd order B-spline basis. Note that for the same set of knots, the support of the B-splines increases in width with increasing order. This has the effect that each B-spline basis function has intersecting support with an increasing number of basis functions as the order increases. Consequently, this reduces the sparsity of the covariance matrix as the B-spline basis order increases.

Table 1: Complexity of variational inference algorithms for sparse variational GP regression in 1D with a Gaussian likelihood. N: number of datapoints; M: number of inducing points; k: bandwidth of the covariance matrix; N_b: size of the mini-batch in stochastic variational inference.

Algorithm	Pre-computation	Computational Complexity	Storage
SGPR	✗	$\mathcal{O}(NM^2 + M^3)$	$\mathcal{O}(NM)$
SVGP	✗	$\mathcal{O}(N_b M^2 + M^3)$	$\mathcal{O}(M^2)$
VFF	$\mathcal{O}(NM^2)$	$\mathcal{O}(M^3)$	$\mathcal{O}(M^2)$
VISH	$\mathcal{O}(NM^2)$	$\mathcal{O}(M^3)$	$\mathcal{O}(M^2)$
AS-VGP (Ours)	$\mathcal{O}(N)$	$\mathcal{O}(M(k+1)^2)$	$\mathcal{O}(M(k+1))$

3.1 B-Spline Inducing Features

B-spline basis functions of order k are a set of compactly supported piece-wise polynomial functions of degree k. Their shape is controlled by an increasing sequence of knots $V = \{v_m\}_{m=0}^{M} \in \mathbb{R}$ that partition the domain into M sub-intervals. We denote the m-th B-spline basis function of order k by $B_{m,k}(x)$, whose implementation details can be found in Appendix C. Since a k-th order B-spline has support over $k+1$ sub-intervals, it has intersecting support with at most $k+1$ other B-spline basis functions (see Figure 1). This leads to the sparsity structure in the matrices $\mathbf{K_{uu}}$ and $\mathbf{K_{uf}}$, as we will show next.

We define the *B-spline inducing features* as the RKHS projection $u_m = \langle f, \phi_m(\cdot)\rangle_{\mathcal{H}}$ onto the B-spline basis, where $\phi_m(x) = B_{m,k}(x)$. Under this choice, the covariance between the inducing features u_m and the GP f reduces to evaluating the B-spline basis

$$[\mathbf{k_u}(x)]_m = \mathrm{Cov}[f(x), u_m] = \langle k(x,\cdot), \phi_m\rangle_{\mathcal{H}} = \phi_m(x) = B_{m,k}(x). \tag{5}$$

This is a sparse vector with at most $k+1$ non-zero entries, since $B_{m,k}(x)$ is non-zero if and only if $x \in [v_m, v_{m+k+1}]$ and is independent of the kernel hyperparameters, as in VFF. Next, the covariance between the inducing features is given by

$$[\mathbf{K_{uu}}]_{m,m'} = \mathrm{Cov}[u_m, u_{m'}] = \langle \phi_m, \phi_{m'}\rangle_{\mathcal{H}}, \tag{6}$$

which is only non-zero when ϕ_m and $\phi_{m'}$ have intersecting support. This produces sparse band-diagonal $\mathbf{K_{uu}}$ matrices with bandwidth equal to $k+1$. Since the B-spline basis functions are piecewise polynomials, we are able to analytically evaluate the inner product in closed form.

3.2 Actually Sparse Variational GP Inference

In the special case of 1-dimensional GP regression with a Gaussian likelihood, since $\mathbf{K_{uf}}$ does not depend on kernel hyperparameters, we can precompute the matrix product $\mathbf{K_{uf}}\mathbf{K_{fu}}$. However, unlike in VFF, due to the sparsity pattern in $\mathbf{K_{uf}}$, the matrix product is now a band diagonal matrix with bandwidth equal to that of $\mathbf{K_{uu}}$. Computing the ELBO in equation (3), therefore has a per iteration cost of $\mathcal{O}(M(k+1)^2)$, required to take the Cholesky decomposition of a banded matrix with bandwidth $k+1$. However, the inverse of a banded matrix is typically not banded, making it difficult to avoid handling a dense matrix when computing the trace of $\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}\mathbf{K_{fu}}$ in the ELBO.

Table 2: Predictive mean squared errors (MSEs), negative log predictive densities (NLPDs) and wall-clock time in seconds with one standard deviation based on 5 random splits for a number of UCI regression datasets. All models use a Matérn-3/2 kernel and L-BFGS optimiser.

			MSE			NLPD		
Dataset	N	M	SGPR	VFF	AS-VGP	SGPR	VFF	AS-VGP
Air Quality	9k	500	0.643 ± 0.035	0.664 ± 0.037	0.668 ± 0.038	1.235 ± 0.003	1.247 ± 0.003	1.254 ± 0.003
Synthetic	10k	50	0.040 ± 0.000	0.039 ± 0.000	0.039 ± 0.000	-0.162 ± 0.000	-0.152 ± 0.000	-0.146 ± 0.000
Rainfall	43k	700	0.048 ± 0.001	0.083 ± 0.002	0.084 ± 0.002	0.104 ± 0.001	0.246 ± 0.001	0.287 ± 0.001
Traffic	48k	300	0.996 ± 0.014	1.001 ± 0.012	1.002 ± 0.012	1.415 ± 0.001	1.416 ± 0.001	1.416 ± 0.001

Table 3: Predictive mean squared errors (MSEs), negative log predictive densities (NLPDs) and wall-clock time in seconds with one standard deviation based on 5 random splits of the household electric power consumption dataset containing $2,049,279$ data points. Number of inducing points used is given by M.

Method	$M = 1000$	$M = 10,000$	$M = 25,000$
AS-VGP (MSE $\times 10^{-1}$)	8.67 ± 0.01	4.53 ± 0.00	2.94 ± 0.00
SVGP (MSE $\times 10^{-1}$)	8.98 ± 0.45	/	/
AS-VGP (NLPD)	1.35 ± 0.00	1.04 ± 0.00	0.86 ± 0.00
SVGP (NLPD)	1.37 ± 0.25	/	/
AS-VGP (Time in s)	3.90 ± 0.05	18.62 ± 0.09	51.69 ± 0.23
SVGP (Time in s)	932 ± 1.18	/	/

Fortunately, using the banded operators introduced by [1], given a banded Cholesky factor of $\mathbf{K_{uu}}$, we are able to compute only the band elements of its inverse at $\mathcal{O}(M(k+1)^2)$. Given that $\mathbf{K_{uf}}\mathbf{K_{fu}}$ is a banded matrix, we can then compute the trace by computing only the bands of the matrix product as $\mathcal{O}(N(k+1)^2)$ allowing us to avoid instantiating a dense matrix. Table 1 highlights the linear scaling in both memory and computational complexity with inducing points of our method compared to other sparse variational methods.

4 Experiments

Here we showcase the performance of AS-VGP on two experiments, with details in Appendix A.

Regression Benchmarks. We first test our method against both SGPR and VFF on 3 UCI datasets and a toy synthetic dataset. Comparing results in Table 2, we show that AS-VGP is comparative in performance to VFF on every dataset, whilst being both less memory and computationally intensive. We note that SGPR performs slighlty better than both VFF and AS-VGP, but at a higher computational complexity.

Large-Scale Regression. In this example, we illustrate the scalability of our method both in the number of data points and in the number of inducing points, using the household electric power consumption dataset, where $N = 2,049,279$. Results are displayed in Table 3. When M=1000, AS-VGP outperforms SVGP both in predictive performance (MSE) and uncertainty quantification (NLPD). AS-VGP also shows an order of magnitude improvement over SVGP using minibatching in terms of wall-clock time. Our results also show that AS-VGP scales approximately linearly in computational time with the number of inducing points.

5 Conclusion

We introduced a novel inter-domain GP model wherein the inducing features are defined as RKHS projections of the GP onto the B-spline basis functions. This results in covariance matrices that are sparse, allowing us to draw entirely on techniques from sparse linear algebra to do training and inference. Our experiments in 1D demonstrate that we get significant computational speed up and memory savings without sacrificing the accuracy.

References

[1] Nicolas Durrande, Vincent Adam, Lucas Bordeaux, Stefanos Eleftheriadis, and James Hensman. Banded matrix operators for Gaussian Markov models in the automatic differentiation era. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[2] Nicolas Durrande, James Hensman, Magnus Rattray, and Neil D Lawrence. Detecting periodicities with Gaussian processes. *PeerJ Computer Science*, 2:e50, 2016.

[3] Vincent Dutordoir, Nicolas Durrande, and James Hensman. Sparse Gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*, 2020.

[4] James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(1):5537–5588, 2017.

[5] James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. In *International Conference on Uncertainty in Artificial Intelligence*, 2013.

[6] James Hensman, Alexander G Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics*, 2015.

[7] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[8] Miguel Lázaro-Gredilla and Anibal Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. *Advances in Neural Information Processing Systems*, 2009.

[9] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bézier and B-spline Techniques*, volume 6. Springer, 2002.

[10] Joaquin Quinonero-Candela and Carl E Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[11] Hugh Salimbeni and Marc P Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems*, 2017.

[12] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial intelligence and Statistics*, 2009.

[13] Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv preprint arXiv:2003.01115*, 2020.

A Experiment Details

All experiments were performed using a AMD Ryzen 2920X 12-Core CPU and a NVIDIA Titan V GPU. Below, we include specific details on the two experiments conducted.

Regression Benchmarks For each dataset (air quality, synthetic, rainfall, traffic), we randomly sample 90% of the data for training and 10% for testing, repeating this 5 times to get means and standard deviations. When using AS-VGP, we normalise the inputs such that the space between knots is equal to 1 to avoid numerical issues caused by large gradients.

For the synthetic dataset, we generate 10,000 random noisy observations from the test function

$$f(x) = \sin(3\pi x) + 0.3\cos(9\pi x) + \frac{\sin(7\pi x)}{2}.$$

Large-scale Regression We use the household electric power consumption dataset, which, after removing missing values has 2,049,279 entries. We repeat each experiment 5 times by randomly sampling 95% of the data for training and use the remaining 5% for evaluation. For each experiment, we use the Matérn-3/2 kernel. We note that we couldn't pre-compute $\mathbf{K_{uf}}\mathbf{K_{fu}}$ in VFF, due to memory constraints.

Metrics We use the mean-squared error (MSE) and the negative log-predictive density (NLPD) to evaluate the performance of our model. These are defined as

$$\text{MSE}(\{X_n, y_n\}_{n=1}^N) = \frac{1}{N}\sum_{n=1}^{N} \|y_n - \mu(X_n)\|^2, \tag{7}$$

$$\text{NLPD}(\{X_n, y_n\}_{n=1}^N) = -\frac{1}{N}\sum_{n=1}^{N} \log \int p(y_n|f_n)\,\mathcal{N}(f_n|\mu(X_n), \xi(X_n))\,\mathrm{d}f_n, \tag{8}$$

where μ, ξ are the posterior mean and variance, respectively.

B RKHS Inner Products

The inner products corresponding to the Matérn-1/2 and Matérn-3/2 RKHS defined over the domain $\mathcal{D} = [a, b]$, as given in [2, 4], are

$$\langle f, g\rangle_{\mathcal{H}_{k_{1/2}}} = \frac{l}{2\sigma^2}\int_a^b f'g'dx + \frac{1}{2l\sigma^2}\int_a^b fgdx + \frac{1}{2\sigma^2}[f(a)g(a) + f(b)g(b)], \tag{9}$$

$$\begin{aligned}\langle f, g\rangle_{\mathcal{H}_{k_{3/2}}} =& \frac{l^3}{12\sqrt{3}\sigma^2}\int_a^b f''g''dx + \frac{l}{2\sqrt{3}\sigma^2}\int_a^b f'g'dx + \frac{\sqrt{3}}{4l\sigma^2}\int_a^b fgdx \qquad (10)\\ &+ \frac{1}{2\sigma^2}[f(a)g(a) + f(b)g(b)] + \frac{l^2}{2\sigma^2}[f'(a)g'(a) + f'(b)g'(b)], \qquad (11)\end{aligned}$$

respectively, where l, σ are the lengthscale and amplitude hyperparameters.

When performing RKHS projections, it is important to note that we must use B-splines that belong to the RKHS defined by the our choice of kernel. As stated in [7], the RKHS generated by the Matérn-ν kernel k is norm-equivalent to the Sobolev space $\mathcal{H}^{\nu+1/2}$. Due to their polynomial form, B-splines of order k are C^{k+1} and therefore belong to the Sobolev space $\mathcal{H}^{k+1}$.

From Section 3.2, to minimise computational complexity we wish to use B-spline basis functions that minimise the bandwidth of the $\mathbf{K_{uu}}$ matrix. As a result for the Matérn-ν kernel we project onto B-splines of order $\nu - 1/2$.

C Implementation of the B-Splines

The m-th B-spline basis function of order k, which we denote by $B_{m,k}(x)$ can be computed according to the Cox-de-Boor recursion formula [9]

$$B_{m,0}(x) = \begin{cases}1, & \text{if } v_m \le x \le v_{m+1},\\ 0, & \text{otherwise,}\end{cases} \tag{12}$$

$$B_{m,k}(x) = \frac{x - v_m}{v_{m+k} - v_m}B_{m,k-1}(x) + \frac{v_{m+k+1} - x}{v_{m+k+1} - v_{m+1}}B_{m+1,k-1}(x). \tag{13}$$

The case $k = 0$ corresponds to a top-hat function $\mathbf{1}_{[v_m, v_{m+1}]}(x)$, with support spanning a single sub-interval. In the case $k = 1$, we have the piecewise linear function

$$B_{m,1}(x) = \begin{cases}\dfrac{x - v_m}{v_{m+1} - v_m}, & \text{for } x \in [v_m, v_{m+1}],\\ \dfrac{v_{m+2} - x}{v_{m+2} - v_{m+1}}, & \text{for } x \in [v_m, v_{m+1}],\\ 0, & \text{otherwise,}\end{cases} \tag{14}$$

which corresponds to the tent map, spanning two sub-intervals (see Figure 1 (a)). In the case $k = 2$, we have the piecewise quadradic function

$$B_{m,2}(x) = \begin{cases} \dfrac{(x - v_m)^2}{(v_{m+2} - v_m)(v_{m+1} - v_m)}, & \text{for } x \in [v_m, v_{m+1}], \\ \dfrac{(x - v_m)(v_{m+2} - x)}{(v_{m+2} - v_m)(v_{m+2} - v_{m+1})} + \dfrac{(v_{m+3} - x)(x - v_{m+1})}{(v_{m+3} - v_{m+1})(v_{m+2} - v_{m+1})}, & \text{for } x \in [v_{m+1}, v_{m+2}], \\ \dfrac{(v_{m+3} - x)^2}{(v_{m+3} - v_{m+1})(v_{m+3} - v_{m+2})}, & \text{for } x \in [v_{m+2}, v_{m+3}], \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

spanning three sub-intervals (see Figure 1 (b)).