Computational Statistics (2021) 36:155–191
https://doi.org/10.1007/s00180-020-01022-x

ORIGINAL PAPER

Adaptive spline fitting with particle swarm optimization

Soumya D. Mohanty[1] · **Ethan Fahnestock**[2]

Received: 29 July 2019 / Accepted: 28 July 2020 / Published online: 6 August 2020

Abstract
In fitting data with a spline, finding the optimal placement of knots can significantly improve the quality of the fit. However, the challenging high-dimensional and non-convex optimization problem associated with completely free knot placement has been a major roadblock in using this approach. We present a method that uses particle swarm optimization (PSO) combined with model selection to address this challenge. The problem of overfitting due to knot clustering that accompanies free knot placement is mitigated in this method by explicit regularization, resulting in a significantly improved performance on highly noisy data. The principal design choices available in the method are delineated and a statistically rigorous study of their effect on performance is carried out using simulated data and a wide variety of benchmark functions. Our results demonstrate that PSO-based free knot placement leads to a viable and flexible adaptive spline fitting approach that allows the fitting of both smooth and non-smooth functions.

1 Introduction

A spline of order k is a piecewise polynomial function that obeys continuity conditions on its value and its first $k-2$ derivatives at the points, called *knots*, where the pieces join (de Boor 2001). Splines play an important role in nonparametric regression (Wegman and Wright 1983; Wahba 1990; Härdle 1990), simply called curve fitting when the data is one dimensional, where the outcome is not assumed to have a predetermined form of functional dependence on the predictor.

It has long been recognized (Wold 1974; Burchard 1974; Jupp 1978; Luo and Wahba 1997) that the quality of a spline fit depends significantly on the locations of

✉ Soumya D. Mohanty
soumya.mohanty@utrgv.edu

Ethan Fahnestock
efahnest@u.rochester.edu

[1] Department of Physics and Astronomy, The University of Texas Rio Grande Valley, One West University Blvd., Brownsville, TX 78520, USA

[2] Department of Physics and Astronomy, The University of Rochester, 500 Wilson Blvd., Rochester, NY 14627, USA

the knots defining the spline. Determining the placement of knots that is best adapted to given data has proven to be a challenging non-linear and non-convex, not to mention high-dimensional, optimization problem that has resisted a satisfactory solution.

A diverse set of methods have been proposed that either attempt this optimization problem head-on or solve an approximation to it in order to get a reasonable solution. In the latter category, methods based on knot insertion and deletion (Smith 1982; Lyche and Mørken 1988; Friedman and Silverman 1989; Friedman 1991; Stone et al. 1997) have been studied extensively. In these methods, one starts with a fixed set of sites for knots and performs a step-wise addition or removal of knots at these sites. The best number of knots is determined by a model selection criterion such as generalized cross validation (GCV) (Luo and Wahba 1997; Golub et al. 1979). Step-wise change in knot placement is not an efficient exploration of the continuous space of possible knot positions and the end result, while computationally inexpensive to obtain and tractable to mathematical analysis, is not necessarily the best possible (Zhou and Shen 2001). Another approach explored in the literature is the two-stage framework in which the first stage identifies a subset of active or dominant knots and the second stage merges them in a data dependent way to obtain a reduced set of knots (Park and Lee 2007; Kang et al. 2015; Luo et al. 2019). These methods have shown good performance for low noise applications.

In attempts at solving the optimization challenge directly, general purpose stochastic optimization algorithms (metaheuristics) such as genetic algorithm (GA) (Mitchell 1998), artificial immune system (AIS) (Ülker and Arslan 2009) or those based on Markov chain Monte Carlo (MCMC) (Green 1995), have been studied (Pittman 2002; DiMatteo et al. 2001; Yoshimoto et al. 2003; Miyata and Shen 2003). These methods have proven quite successful in solving many challenging high-dimensional optimization problems in other fields and it is only natural to employ them for the problem of free knot placement. However, GA and AIS are more suited to discrete optimization problems rather than the inherently continuous one in knot optimization, and MCMC is computationally expensive. Thus, there is plenty of scope for using other metaheuristics to find better solutions.

It was shown in Gálvez and Iglesias (2011), and independently in Mohanty (2012), that particle swarm optimization (PSO) (Kennedy and Eberhart 1995), a relatively recent entrant to the field of nature inspired metaheuristics such as GA, is a promising method for the free knot placement problem. PSO is governed by a much smaller set of parameters than GA or MCMC and most of these do not appear to require much tuning from one problem to another. In fact, as discussed later in the paper, essentially two parameters are all that need to be explored to find a robust operating point for PSO.

An advantage of free knot placement is that a subset of knots can move close enough to be considered as a single knot with a higher multiplicity. A knot with multiplicity > 1 can be used to construct splines that can fit curves with discontinuities. Thus, allowing knots to move and merge opens up the possibility of modeling even non-smooth curves. That PSO can handle regression models requiring knot merging was demonstrated in Gálvez and Iglesias (2011) albeit for examples with very low noise levels.

It was found in Mohanty (2017), and later in a simplified model problem (Mohanty 2018), that the advantage engendered by free knot placement turns into a liability as the level of noise increases: knots can form spurious clusters to fit outliers arising from noise, producing spikes in the resulting estimate and making it worse than useless. This problem was found to be mitigated (Mohanty 2018) by introducing a suitable regulator (Ruppert et al. 2003). Regularization has also been used in combination with knot addition (Luo and Wahba 1997) but its role there—suppression of numerical instability arising from a large numbers of knots—is very different.

The progress on free knot placement described above has happened over decades and in somewhat isolated steps that were often limited by the available computing power. However, the tremendous growth in computing power and the development of more powerful metaheuristics has finally brought us to the doorstep of a satisfactory resolution of this problem, at least for one-dimensional regression.

In this paper, we combine PSO based knot placement with regularization into a single algorithm for adaptive spline fitting. The algorithm, called Swarm Heuristics based adaptive and penalized estimation of splines (`SHAPES`), has the flexibility to fit non-smooth functions as well as smooth ones without any change in algorithm settings. It uses model selection to determine the best number of knots, and reduces estimation bias arising from the regularization using a least squares derived rescaling. Some of the elements of `SHAPES` outlined above were explored in Mohanty (2018) in the context of a single example with a simple and smooth function. However, the crucial feature of allowing knots to merge was missing there along with the step of bias reduction. (The bias reduction step does not seem to have been used elsewhere to the best of our knowledge.)

Various design choices involved in `SHAPES` are identified clearly and their effects are examined using large-scale simulations and a diverse set of benchmark functions. Most importantly, `SHAPES` is applied to data with a much higher noise level than has traditionally been considered in the field of adaptive spline fitting and found to have promising performance. This sets the stage for further development of the adaptive spline methodology for new application domains.

The rest of the paper is organized as follows. Section 2 provides a brief review of pertinent topics in spline fitting. The PSO metaheuristic and the particular variant used in this paper are reviewed in Sect. 3. Details of `SHAPES` are described in Sect. 4 along with the principal design choices. The setup used for our simulations is described in Sect. 5. Computational aspects of `SHAPES` are addressed in Sect. 6. This is followed by the presentation of results in Sect. 7. Our conclusions are summarized in Sect. 8.

2 Fitting splines to noisy data

In this paper, we consider the one-dimensional regression problem

$$y_i = f(x_i) + \epsilon_i, \tag{1}$$

$i = 0, 1, \ldots, N-1$, $x_0 = 0$, $x_{N-1} = 1$, $x_{i+1} > x_i$, with $f(x)$ unknown and ϵ_i drawn independently from $N(0, 1)$. The task is to find an estimate $\widehat{f}(x)$, given $\{y_i\}$, of $f(x)$.

To obtain a non-trivial solution, the estimation problem must be regularized by restricting $\widehat{f}(x)$ to some specified class of functions. One reasonable approach is to require that this be the class of "smooth" functions, and obtain the estimate as the solution of the variational problem,

$$\widehat{f} = \arg\min_f \left[\sum_{i=0}^{N-1} (y_i - f(x_i))^2 + \lambda \int_0^1 dx \left(f''(x) \right)^2 \right]. \tag{2}$$

It can be shown that the solution belongs to the space of cubic splines defined by $\{x_i\}$ as the set of knots. Consequently, $\widehat{f}$ is known as the *smoothing spline* estimate (Wahba 1990; Reinsch 1967). In Eq. (2), the first term on the right measures the fidelity of the model to the observations and the second term penalizes the "roughness", measured by the average squared curvature, of the model. The trade-off between these competing requirements is controlled by $\lambda \geq 0$, called the regulator gain or smoothing parameter.

The best choice for λ is the principle issue in practical applications of smoothing spline. The use of GCV to adaptively determine the value of λ was introduced in Craven and Wahba (1978) and is used, for example, in the implementation of smoothing spline in the R (Core Team 2019) `stats` package. A scalar λ, adaptively selected or otherwise, is not well suited to handle a function with a heterogeneous roughness distribution across its domain. The use of a spatially adaptive gain function, $\lambda(x)$, has been investigated in different forms (Wahba 2002; Storlie et al. 2010; Liu and Guo 2010; Wang et al. 2013) to address this issue.

A different regularization approach is to eschew an explicit penalty term and regularize the fitting problem by restricting the number of knots to be $\ll N$. This leads to the *regression spline* (Wold 1974) estimate in which $\widehat{f}(x)$ is represented as a linear combination of a finite set of basis functions—the so-called B-spline functions (de Boor 2001; Curry and Schoenberg 1947) being a popular choice—that span the space of splines associated with the chosen knot sequence and polynomial order. Different methods for adaptive selection of the number of knots, which is the main free parameter in regression spline, have been compared in Wand (2000). The asymptotic properties of smoothing and regression spline estimates have been analyzed theoretically in Claeskens et al. (2009).

Smoothing and regression splines are hybridized in the *penalized spline* (Ruppert et al. 2003; Eilers and Marx 1996; Eilers et al. 2015) approach: the deviation of the spline model from the data is measured by the least squares function as in the first term of Eq. (2) but the penalty becomes a quadratic form in the coefficients of the spline in the chosen basis set. As in the case of smoothing spline, adaptive selection of the scalar regulator gain can be performed using GCV (Ruppert et al. 2003) and locally adaptive gain coefficients have been proposed in Ruppert and Carroll (2000), Krivobokova et al. (2008), Scheipl and Kneib (2009) and Yang and Hong (2017). The performance of alternatives to GCV for selection of a scalar regulator gain have been investigated and compared in Krivobokova (2013).

While penalized spline is less sensitive to the number of knots, it is still a free parameter of the algorithm that must be specified. Joint adaptive selection of the number of knots and regulator gain has been investigated in Luo and Wahba (1997)

and Ruppert (2002) using GCV. Other model selection methods can also be used for adaptive determination of the number of knots (see Sect. 2.3).

2.1 B-spline functions

Given a set of M knots $\overline{b} = (b_0, b_1, \ldots, b_{M-1})$, $b_i \in [0, 1]$, $b_{i+1} > b_i$, and given order k of the spline polynomials, the set of splines that interpolates $\{(y_i, b_i)\}$, $y_i \in \mathbb{R}$, forms a linear vector space of dimensionality $M + k - 2$. A convenient choice for a basis of this vector space is the set of B-spline functions (Curry and Schoenberg 1947).

In this paper, we need B-spline functions for the more general case of a knot sequence $\overline{\tau} = (\tau_0, \tau_1, \ldots, \tau_{P-1})$, $\tau_{i+1} \geq \tau_i$ with $P > M$ knots, in which a knot can appear more than once. The number of repetitions of any knot cannot be greater than k. Also, $\tau_j = b_0$ for $0 \leq j \leq k - 1$, and $\tau_j = b_{M-1}$ for $P - k \leq j \leq P - 1$. The span of B-spline functions defined over a knot sequence with repetitions can contain functions that have jump discontinuities in their values or in their derivatives. (The dimensionality of the span is $P - k$.)

The Cox–de Boor recursion relations (de Boor 1972) given below provide an efficient way to compute the set of B-spline functions, $\{B_{i,k}(x; \overline{\tau})\}$, for any given order. The recursions start with B-splines of order 1, which are piecewise constant functions

$$B_{j,1}(x; \overline{\tau}) = \begin{cases} 1, & \tau_j \leq x < \tau_{j+1} \\ 0 & \text{else} \end{cases}. \tag{3}$$

For $2 \leq k' \leq k$,

$$B_{j,k'}(x) = \omega_{j,k'}(x) B_{j,k'-1}(x) + \gamma_{j+1,k'}(x) B_{j+1,k'-1}(x), \tag{4}$$

$$\omega_{j,k'}(x) = \begin{cases} \frac{x - \tau_j}{\tau_{j+k'-1} - \tau_j}, & \tau_{j+k'-1} \neq \tau_j \\ 0, & \tau_{j+k'-1} = \tau_j \end{cases}, \tag{5}$$

$$\gamma_{j,k'}(x) = \begin{cases} 1 - \omega_{j,k'}(x), & \tau_{j+k'-1} \neq \tau_j \\ 0, & \tau_{j+k'-1} = \tau_j \end{cases}. \tag{6}$$

In the recursion above, $0 \leq j \leq P - k' - 1$. Figure 1 provides an illustration of B-spline functions.

The regression spline method is elegantly formulated in terms of B-spline functions. The estimate is assumed to belong to the parametrized family of linearly combined B-spline functions,

$$f(x; \overline{\alpha}, \overline{\tau}) = \sum_{j=0}^{P-k-1} \alpha_j B_{j,k}(x_i; \overline{\tau}), \tag{7}$$

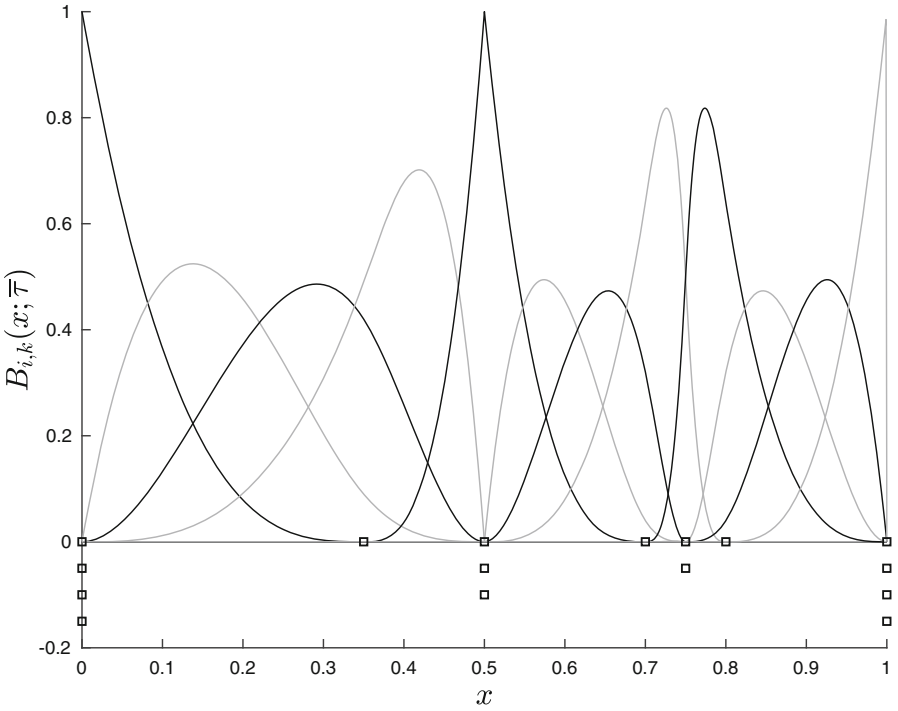

Fig. 1 Cubic B-spline functions $\{B_{i,4}(x;\overline{\tau})\}, i = 0, 1, \ldots, 11$, for an arbitrary choice of 16 knots ($\overline{\tau}$) marked by squares. For visual clarity, alternate B-spline functions are shown in black and gray. Knots with multiplicity > 1 result in B-splines that are discontinuous in value or derivatives

where $\overline{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_{P-k-1})$. The least-squares estimate is given by $\widehat{f}(x) = f(x; \widehat{\alpha}, \widehat{\tau})$, where $\widehat{\alpha}$ and $\widehat{\tau}$ minimize

$$L(\overline{\alpha}, \overline{\tau}) = \sum_{i=0}^{N-1} \left(y_i - f(x_i; \overline{\alpha}, \overline{\tau})\right)^2 . \tag{8}$$

2.2 Regression and penalized spline with free knot placement

The penalized spline estimate is found by minimizing

$$L_\lambda(\overline{\alpha}, \overline{\tau}) = L(\overline{\alpha}, \overline{\tau}) + \lambda R(\overline{\alpha}), \tag{9}$$

over the spline coefficients (c.f. Eq. 7), where $R(\overline{\alpha})$ is the penalty, while keeping the number of knots and knot locations fixed. In this paper, we choose

$$R(\overline{\alpha}) = \sum_{j=0}^{P-k-1} \alpha_j^2, \tag{10}$$

for reasons explained below.

Formally, the penalty function can be derived by substituting Eq. (7) in the roughness penalty. This would lead to a quadratic form similar to the penalty in Eq. (10) but with a kernel matrix that is not the identity matrix (Ramsay and Silverman 1997). The elements of this matrix would be Euclidean inner products of B-spline derivatives. However, using such a penalty adds a substantial computational burden in free knot placement because it has to be recomputed every time the knot placement changes. Computational aspects of this problem are discussed in Eilers and Marx (1996), where a simplified form of the roughness penalty is used that is based on the differences of coefficients of adjacent B-splines. This is a good approximation for the case considered in Eilers and Marx (1996) of a large number of fixed knots and closely spaced B-splines, but not necessarily for free knots that may be small in number and widely spread out. Another perhaps more important consideration is that repeated knots in free knot placement result in B-splines with discontinuous derivatives. This makes the kernel matrix particularly challenging for numerical evaluation and increases code complexity. In this paper, we avoid the above issues by using the simple form of the penalty function in Eq. (10) and leave the investigation of more appropriate forms to future work. We note that the exploration of innovative penalty functions is an active topic of research (e.g., Lindstrom 1999; Eilers et al. 2015; Goepp et al. 2018).

While the reduction of the number of knots in regression spline coupled with the explicit regularization of penalized spline reduces overfitting, the fit is now sensitized to where the knots are placed. Thus, the complete method involves the minimization of $L_\lambda(\overline{\alpha}, \overline{\tau})$ (c.f., Eq. 9) over both $\overline{\alpha}$ and $\overline{\tau}$. (The method of regression spline with knot optimization and explicit regularization will be referred to as *adaptive spline* in the following.)

Minimization of L_λ over $\overline{\alpha}$ and $\overline{\tau}$ can be nested as follows.

$$\min_{\overline{\tau},\overline{\alpha}} L_\lambda(\overline{\alpha}, \overline{\tau}) = \min_{\overline{\tau}} \left(\min_{\overline{\alpha}} L_\lambda(\overline{\alpha}, \overline{\tau}) \right). \tag{11}$$

The solution, $\widehat{\alpha}(\overline{\tau})$, of the inner minimization is expressed in terms of the $(P-k)$-by-N matrix $\mathbf{B}(\overline{\tau})$, with

$$B_{m,n}(\overline{\tau}) = B_{m,k}(x_n; \overline{\tau}), \tag{12}$$

as

$$\widehat{\alpha}(\overline{\tau}) = \overline{y}\mathbf{B}^T\mathbf{G}^{-1}, \tag{13}$$

$$\mathbf{G} = \mathbf{B}\mathbf{B}^T + \lambda\mathbf{I}, \tag{14}$$

where $\mathbf{I}$ is the $(P-k)$-by-$(P-k)$ identity matrix. The outer minimization over $\overline{\tau}$ of

$$F_\lambda(\overline{\tau}) = L_\lambda(\widehat{\alpha}(\overline{\tau}), \overline{\tau}), \tag{15}$$

needs to be performed numerically.

Due to the fact that freely moveable knots can coincide, and that this produces discontinuities in B-spline functions as outlined earlier, curve fitting by adaptive spline can accommodate a broader class of functions—smooth with localized discontinuities—than smoothing or penalized spline.

The main bottleneck in implementing the adaptive spline method is the global minimization of $F_\lambda(\overline{\tau})$ since it is a high-dimensional non-convex function having multiple local minima. Trapping by local minima renders greedy methods ineffective and high dimensionality makes a brute force search for the global minimum computationally infeasible. This is where PSO enters the picture and, as shown later, offers a way forward.

2.3 Model selection

In addition to the parameters $\overline{\alpha}$ and $\overline{\tau}$, adaptive spline has two hyper-parameters, namely the regulator gain λ and the number of interior knots $P - 2(k-1)$, that affect the outcome of fitting. Model selection methods can be employed to fix these hyper-parameters based on the data.

In this paper, we restrict ourselves to the adaptive selection of only the number of knots. This is done by minimizing the Akaike information criterion (AIC) Akaike (1998): For a regression model with K parameters $\overline{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K-1})$,

$$\mathrm{AIC} = 2K - 2 \max_{\overline{\theta}} \ln\left(\Lambda(\overline{\theta})\right), \tag{16}$$

where $\Lambda(\overline{\theta})$ is the likelihood function. The specific expression for AIC used in `SHAPES` is provided in Sect. 4.

3 Particle swarm optimization

Under the PSO metaheuristic, the function to be optimized (called the *fitness function*) is sampled at a fixed number of locations (called *particles*). The set of particles is called a *swarm*. The particles move in the search space following stochastic iterative rules called *dynamical equations*. The dynamical equations implement two essential features called *cognitive* and *social* forces. They serve to retain "memories" of the best locations found by the particle and the swarm (or a subset thereof) respectively.

Since its introduction by Kennedy and Eberhart (1995), the PSO metaheuristic has expanded to include a large diversity of algorithms (Engelbrecht 2005). In this paper, we consider the variant called local-best (or *lbest*) PSO (Kennedy 1999). We begin with the notation (Normandin et al. 2018) for describing lbest PSO.

- $F(x)$: the scalar fitness function to be minimized, with $x = (x^1, x^2, \ldots, x^d) \in \mathbb{R}^d$. In our case, x is $\overline{\tau}$, F is $F_\lambda(\overline{\tau})$ (c.f., Eq. 15), and $d = P - 2(k-1)$.
- $\mathcal{S} \subset \mathbb{R}^d$: the search space defined by the hypercube $a^j \leq x^j \leq b^j, = 1, 2, \ldots, d$ in which the global minimum of the fitness function must be found.
- N_p: the number of particles in the swarm.
- $x_i[k] \in \mathbb{R}^d$: the position of the ith particle at the kth iteration.

- $v_i[k] \in \mathbb{R}^d$: a vector called the *velocity* of the ith particle that is used for updating the position of a particle.
- $p_i[k] \in \mathbb{R}^d$: the best location found by the ith particle over all iterations up to and including the kth. $p_i[k]$ is called the *personal best* position of the ith particle.

$$F(p_i[k]) = \min_{1 \le j \le k} F(x_i[j]). \tag{17}$$

- $n_i[k]$: a set of particles, called the *neighborhood* of particle i, $n_i[k] \subseteq \{1, 2, \ldots, N_p\} \setminus \{i\}$. There are many possibilities, called *topologies*, for the choice of $n_i[k]$. In the simplest, called the global best topology, every particle is the neighbor of every other particle: $n_i[k] = \{1, 2, \ldots, N_p\} \setminus \{i\}$. The topology used for lbest PSO in this paper is described later.
- $l_i[k] \in \mathbb{R}^d$: the best location among the particles in $n_i[k]$ over all iterations up to and including the kth. $l_i[k]$ is called the *local best* for the ith particle.

$$F(l_i[k]) = \min_{j \in \{i\} \cup n_i[k]} F(p_j[k]). \tag{18}$$

- $p_g[k] \in \mathbb{R}^d$: The best location among all the particles in the swarm, $p_g[k]$ is called the *global best*.

$$F(p_g[k]) = \min_{1 \le i \le N_p} F(p_i[k]). \tag{19}$$

The dynamical equations for *lbest* PSO are as follows.

$$\begin{aligned} v_i[k+1] &= w[k]v_i[k] + c_1(p_i[k] - x_i[k])\mathbf{r}_1 \\ &\quad + c_2(l_i[k] - x_i[k])\mathbf{r}_2, \end{aligned} \tag{20}$$

$$x_i[k+1] = x_i[k] + z_i[k+1], \tag{21}$$

$$z_i^j[k] = \begin{cases} v_i^j[k], & -v_{\max}^j \le v_i^j[k] \le v_{\max}^j \\ -v_{\max}^j, & v_i^j[k] < -v_{\max}^j \\ v_{\max}^j, & v_i^j[k] > v_{\max}^j \end{cases}. \tag{22}$$

Here, $w[k]$ is a deterministic function known as the inertia weight, c_1 and c_2 are constants, and $\mathbf{r}_i$ is a diagonal matrix with iid random variables having a uniform distribution over [0, 1]. Limiting the velocity as shown in Eq. (22) is called *velocity clamping*.

The iterations are initialized at $k = 1$ by independently drawing (i) $x_i^j[1]$ from a uniform distribution over $[a^j, b^j]$, and (ii) $v_i^j[1]$ from a uniform distribution over $[a^j - x_i^j[1], b^j - x_i^j[1]]$. For termination of the iterations, we use the simplest condition: terminate when a prescribed number N_{iter} of iterations are completed. The solutions found by PSO for the minimizer and the minimum value of the fitness are $p_g[N_{\text{iter}}]$ and $F(p_g[N_{\text{iter}}])$ respectively. Other, more sophisticated, termination conditions are available (Engelbrecht 2005), but the simplest one has served well across a variety of regression problems in our experience.

The second and third terms on the RHS of Eq. (20) are the cognitive and social forces respectively. On average they attract a particle towards its personal and local bests, promoting the exploitation of an already good solution to find better ones nearby. The term containing the inertia weight, on the other hand, promotes motion along the same direction and allows a particle to resist the cognitive and social forces. Taken together, the terms control the exploratory and exploitative behaviour of the algorithm. We allow the inertia weight $w[k]$ to decrease linearly with k from an initial value $w_{\max}$ to a final value $w_{\min}$ in order to transition PSO from an initial exploratory to a final exploitative phase.

For the topology, we use the *ring topology* with 2 neighbors in which

$$n_i[k] = \begin{cases} \{i-1, i+1\}, & i \notin \{1, N_p\} \\ \{N_p, i+1\}, & i = 1 \\ \{i-1, 1\}, & i = N_p \end{cases}. \tag{23}$$

The local best, $l_i[k]$, in the kth iteration is updated after evaluating the fitnesses of all the particles. The velocity and position updates given by Eqs. (20) and (21) respectively form the last set of operations in the kth iteration.

To handle particles that exit the search space, we use the "let them fly" boundary condition under which a particle outside the search space is assigned a fitness value of ∞. Since both $p_i[k]$ and $l_i[k]$ are always within the search space, such a particle is eventually pulled back into the search space by the cognitive and social forces.

3.1 PSO tuning

Stochastic global optimizers, including PSO, that terminate in a finite number of iterations do not satisfy the conditions laid out in Solis and Wets (1981) for convergence to the global optimum. Only the probability of convergence can be improved by tuning the parameters of the algorithm for a given optimization problem.

In this sense, most of the parameters involved in PSO are found to have fairly robust values when tested across an extensive suite of benchmark fitness functions (Bratton and Kennedy 2007). Based on widely prevalent values in the literature, these are: $N_p = 40$, $c_1 = c_2 = 2.0$, $w_{\max} = 0.9$, $w_{\min} = 0.4$, and $v^j_{\max} = 0.5[b^j - a^j]$.

Typically, this leaves the maximum number of iterations, N_{iter}, as the principal parameter that needs to be tuned. However, for a given N_{iter}, the probability of convergence can be increased by the simple strategy of running multiple, independently initialized runs of PSO on the same fitness function and choosing the best fitness value found across the runs. The probability of missing the global optimum decreases exponentially as $(1 - P_{\text{conv}})^{N_{\text{runs}}}$, where P_{conv} is the probability of successful convergence in any one run and N_{runs} is the number of independent runs.

Besides N_{iter}, therefore, N_{runs} is the remaining parameter that should be tuned. If the independent runs can be parallelized, N_{runs} is essentially fixed by the available number of parallel workers although this should not be stretched to the extreme. If too high a value of N_{runs} is needed in an application (say $N_{\text{runs}} \geq 8$), it is usually an indicator that P_{conv} should be increased by tuning the other PSO parameters or by

Input:
• $\overline{y} \leftarrow$ Data
• $N_{\text{runs}} \leftarrow$ Number of PSO runs
• $N_{\text{iter}} \leftarrow$ Maximum number of iterations
• $\mathbb{N}_{\text{knots}} \leftarrow \{M_1, M_2, \ldots, M_{\max}\}$; Number of knots (not counting repetitions)
• $\lambda \leftarrow$ Regulator gain
Execute:
for $M \in \mathbb{N}_{\text{knots}}$ **do** ▷ Loop over models
  for $r \in \{1, 2, \ldots, N_{\text{runs}}\}$ **do** ▷ (Parallel) loop over PSO runs
    $\widehat{\tau}(r) \leftarrow \arg\min_{\overline{\tau}} F_\lambda(\overline{\tau})$ using PSO ▷ Best location
    $\widehat{\alpha}(r) \leftarrow$ B-spline coefficients corresponding to $\widehat{\tau}(r)$
    $F(M, r) \leftarrow F_\lambda(\widehat{\tau}(r))$ ▷ Best fitness value
  end for
  $r_M \leftarrow \arg\min_r F(M, r)$ ▷ Best PSO run
  $\text{AIC}(M) \leftarrow$ AIC for $F(M, r_M)$ (c.f., Eq. 24)
  $\widehat{f}(M) \leftarrow$ Estimated function corresponding to $\widehat{\tau}(r_M)$ and $\widehat{\alpha}(r_M)$
end for
$M_{\text{best}} \leftarrow \arg\min_M \text{AIC}(M)$ ▷ Model with lowest AIC
$\widehat{f} \leftarrow \widehat{f}(M_{\text{best}})$
$\widehat{f} \leftarrow$ Bias corrected $\widehat{f}$ (c.f., Sec. 4.1)
Output:
• M_{best} ▷ Best model
• Estimated, bias-corrected $\widehat{f}$ ▷ Estimated function from best model
• $F(M_{\text{best}}, r_{M_{\text{best}}})$ ▷ Fitness of best model

Fig. 2 Pseudo-code for the SHAPES algorithm. All quantities with parenthesized integer arguments stand for arrays, with the argument as the array index

exploring a different PSO variant. In this paper, we follow the simpler way of tuning N_{runs} by setting it to $N_{\text{runs}} = 4$, the typical number of processing cores available in a high-end desktop.

4 SHAPES algorithm

The SHAPES algorithm is summarized in the pseudo-code given in Fig. 2. The user specified parameters of the algorithm are (i) the number, N_{runs}, of PSO to use per data realization; (ii) the number of iterations, N_{iter}, to termination of PSO; (iii) the set of models, $\mathbb{N}_{\text{knots}}$, over which AIC based model selection (see below) is used; (iv) the regulator gain λ. Following the standard initialization condition for PSO (c.f., Sect. 3), the initial knots for each run of PSO are drawn independently from a uniform distribution over [0, 1].

A model in SHAPES is specified by the number of non-repeating knots. For each model $M \in \mathbb{N}_{\text{knots}}$, $F(M, r_M)$ denotes the fitness value, where $1 \leq r_M \leq N_{\text{runs}}$ is the best PSO run. The AIC value for the model is given by

$$\text{AIC} = 4M + F(M, r_M), \tag{24}$$

which follows from the number of optimized parameters being $2M$ (accounting for both knots and B-spline coefficients) and the log-likelihood being proportional to the

least squares function for the noise model used here. (Additive constants that do not affect the minimization of AIC have been dropped.)

The algorithm acts on given data $\overline{y}$ to produce (i) the best fit model $M_{\text{best}} \in \mathbb{N}_{\text{knots}}$; (ii) the fitness value associated with the best fit model; (iii) the estimated function $\widehat{f}$ from the best fit model. The generation of $\widehat{f}$ includes a bias correction step described next.

4.1 Bias correction

The use of a non-zero regulator gain leads to shrinkage in the estimated B-spline coefficients. As a result, the corresponding estimate, $\widehat{f}$, has a systematic point-wise bias towards zero. A bias correction transformation is applied to $\widehat{f}$ as follows.

First, the unit norm estimated function $\widehat{u}$ is obtained,

$$\widehat{u} = \frac{\widehat{f}}{\|\widehat{f}\|}, \tag{25}$$

where $\|\widehat{f}\| = [\sum \widehat{f}_i^2]^{1/2}$ is the L_2 norm.

Next, a scaling factor A is estimated as

$$A = \arg\min_a \sum_{i=0}^{N-1} (y_i - a\widehat{u})^2 . \tag{26}$$

The final estimate is given by $\widehat{f} = A\widehat{u}$.

As discussed earlier in Sect. 2.2 (c.f. Eqs. 9 and 10), the penalty used in this paper is one among several alternatives available in the literature. For some forms of the penalty, there need not be any shrinkage in the B-spline coefficients and the bias correction step above would be unnecessary.

4.2 Knot merging and dispersion

In both of the mappings described in Sect. 4.3, it is possible to get knot sequences in which a subset $(\tau_i, \tau_{i+1}, \ldots, \tau_{i+m-1})$ of $1 < m \leq M - 2$ of interior knots falls within an interval (x_j, x_{j+1}) between two consecutive predictor values. There are two possible options to handle such a situation.

- Heal: Overcrowded knots are dispersed such that there is only one knot between any two consecutive predictor values. This can be done iteratively by moving a knot to the right or left depending on the difference in distance to the corresponding neighbors.
- Merge: All the knots in an overcrowded set are made equal to the rightmost knot τ_{i+m-1} until its multiplicity saturates at k. The remaining knots, τ_i to $\tau_{i+m-1-k}$, are equalized to the remaining rightmost knot $\tau_{i+m-1-k}$ until its multiplicity saturates to k, and so on. (Replacing rightmost by leftmost when merging is an equally valid

alternative.) Finally, if more than one set of merged knots remain within an interval (x_j, x_{j+1}), they are dispersed by healing.

If only healing is used, SHAPES cannot fit curves that have jump discontinuities in value or derivatives. Therefore, if it is known that the unknown curve in the data is free of jump discontinuities, healing acts as an implicit regularization to enforce this condition. Conversely, merging should be used when jump discontinuities cannot be discounted.

It is important to note that in both healing and merging, the number of knots stays fixed at $M + 2(k-1)$ where $M \in \mathbb{N}_{\text{knots}}$.

4.3 Mapping particle location to knots

For a given model $M \in \mathbb{N}_{\text{knots}}$, the search space for PSO is M dimensional. Every particle location, $\overline{z} = (z_0, z_1, \ldots, z_{M-1})$, in this space has to be mapped to an $M + 2(k-1)$ element knot sequence $\overline{\tau}$ before evaluating its fitness $F_\lambda(\overline{\tau})$.

We consider two alternatives for the map from $\overline{z}$ to $\overline{\tau}$.

- Plain: $\overline{z}$ is sorted in ascending order. After sorting, $k-1$ copies of z_0 and z_{M-1} are prepended and appended respectively to $\overline{z}$. These are the repeated end knots as described in Sect. 2.1.
- Centered-monotonic: In this scheme (Leung 2015), the search space is the unit hypercube: $z_i \in [0, 1], \forall i$. First, an initial set of M knots is obtained from

$$z_0 = \tau_0, \tag{27}$$

$$z_{1 \le i \le M-2} = \frac{\tau_i - \tau_{i-1}}{\tau_{i+1} - \tau_{i-1}}, \tag{28}$$

$$z_{M-1} = \frac{\tau_{M-1} - \tau_0}{1 - \tau_0}. \tag{29}$$

This is followed by prepending and appending $k-1$ copies of τ_0 and τ_{M-1} respectively to the initial knot sequence.

In the plain map, any permutation of $\overline{z}$ maps into the same knot sequence due to sorting. This creates degeneracy in F_λ, which may be expected to make the task of global minimization harder for PSO. The centered-monotonic map is designed to overcome this problem: by construction, it assigns a unique $\overline{\tau}$ to a given $\overline{z}$. Moreover, $\overline{\tau}$ is always a monotonic sequence, removing the need for a sorting operation. This map also has the nice normalization that the center of the search space at $z_i = 0.5$, $1 \le i \le M-2$, corresponds to uniform spacing of the interior knots.

It should be noted here that the above two maps are not the only possible ones. The importance of the "lethargy theorem" (degeneracy of the fitness function) and using a good parametrization for the knots in regression spline was pointed out by Jupp (1978) back in 1978. A logarithmic map for knots was proposed in Jupp (1978) that, while not implemented in this paper, should be examined in future work.

4.4 Optimization of end knots

When fitting curves to noisy one-dimensional data in a signal processing context, a common situation is that the signal is transient and localized well away from the end points x_0 and x_{N-1} of the predictor. However, the location of the signal in the data—its time of arrival in other words—may be unknown. In such a case, it makes sense to keep the end knots free and subject to optimization.

On the other hand, if it is known that the curve occupies the entire predictor range, it is best to fix the end knots by keeping z_0 and z_{M-1} fixed. (This reduces the dimensionality of the search space for PSO by 2.)

4.5 Retention of end B-splines

The same signal processing scenario considered above suggests that, for signals that decay smoothly to zero at their start and end, it is best to drop the end B-spline functions because they have a jump discontinuity in value (c.f., Fig. 1). In the contrary case, the end B-splines may be retained so that the estimated signal can start or end at non-zero values.

5 Simulation study setup

We examine the performance of `SHAPES` on simulated data with a wide range of benchmark functions. In this section, we present these functions, the simulation protocol used, the metrics for quantifying performance, and a scheme for labeling test cases that is used in Sect. 7. (In the following, the terms "benchmark function" and "benchmark signal" are used interchangeably.)

5.1 Benchmark functions

The benchmark functions used in this study are listed in Table 1 and plotted in Fig. 3.

Function f_1 has a sharp change but is differentiable everywhere. Functions f_2 and f_6 have jump discontinuities, and f_3 has a jump discontinuity in its slope. Functions f_4 and f_5 are smooth but sharply peaked. Functions f_7 to f_{10} all decay to zero at both ends and serve to model smooth but transient signals; f_7 to f_9 are designed to require progressively higher number of knots for fitting; f_{10} is an oscillatory signal that is typical for signal processing applications and expected to require the highest number of knots. In addition, f_7 and f_8 test the ability of `SHAPES` to localize time of arrival.

5.2 Data simulation

Following the regression model in Eq. (1), a simulated data realization consists of pseudorandom iid noise drawn from $N(0, 1)$ added to a given benchmark function that is sampled uniformly at 256 points in $[0, 1]$.

Table 1 The benchmark functions used in this paper

Expression	Domain
$f_1(x) = 90(1 + e^{-100(x-0.4)})^{-1}$	$x \in [0, 1]$
$f_2(x) = \begin{cases} (0.01 + (x - 0.3)^2)^{-1} \\ (0.015 + (x - 0.65)^2)^{-1} \end{cases}$	$0 \le x < 0.6$ $0.6 \le x \le 1$
$f_3(x) = 100e^{-\lvert 10x-5\rvert} + (10x - 5)^5/500$	$x \in [0, 1]$
$f_4(x) = \sin(x) + 2e^{-30x^2}$	$x \in [-2, 2]$
$f_5(x) = \sin(2x) + 2e^{-16x^2} + 2$	$x \in [-2, 2]$
$f_6(x) = \begin{cases} 4x^2(3 - 4x) \\ \frac{4}{3}x(4x^2 - 10x + 7) - \frac{3}{2} \\ \frac{16}{3}x(x - 1)^2 \end{cases}$	$0 \le x < 0.5$ $0.5 \le x < 0.75$ $0.75 \le x \le 1$
$f_7(x) = B_{3,4}(x; \overline{\tau})$; $\overline{\tau} = (\tau_0, \tau_1, \ldots, \tau_{11})$ $\tau_i = \begin{cases} 0.3, & 0 \le i \le 2 \\ 0.55, & 8 \le i \le 10 \end{cases}$ $(\tau_3, \ldots, \tau_7) = (0.3, 0.4, .45, 0.5, 0.55)$	$x \in [0, 1]$
$f_8(x) = B_{3,4}(x; \overline{\tau}) + B_{3,4}(x - 0.125; \overline{\tau})$	$x \in [0, 1]$
$f_9(x) = B_{3,4}(x - 0.25, \overline{\tau}) + B_{3,4}(x - 0.125; \overline{\tau})$	$x \in [0, 1]$
$f_{10}(x) = e^{-\frac{(x-0.5)^2}{0.125}} \sin(10.24\pi(x - 0.5))$	$x \in [0, 1]$

The sources from which the functions have been obtained are: f_1 to f_3 (Yoshimoto et al. 2003); f_4 (DiMatteo et al. 2001); f_5 (Denison et al. 1998; Li and Yan 2006); f_6 (Lee 2002); f_7 (Mohanty 2018). Functions f_8 to f_{10} are introduced here

We consider the performance of SHAPES across a range of signal to noise ratio (SNR) defined as,

$$\text{SNR} = \frac{\|f\|}{\sigma}, \tag{30}$$

where f is a benchmark function and σ is the standard deviation—set to unity in this paper—of the noise. For each combination of benchmark function and SNR, SHAPES is applied to $N_R = 1000$ independent data realizations. This results in 1000 corresponding estimated functions. Statistical summaries, such as the point-wise mean and standard deviation of the estimate, are computed from this set of estimated functions.

5.3 Metrics

The principal performance metric used in this paper is the sample root mean squared error (RMSE):

$$\text{RMSE} = \left[\frac{1}{N_R} \sum_{j=1}^{N_R} \|f - \widehat{f}_j\|^2 \right]^{1/2}, \tag{31}$$

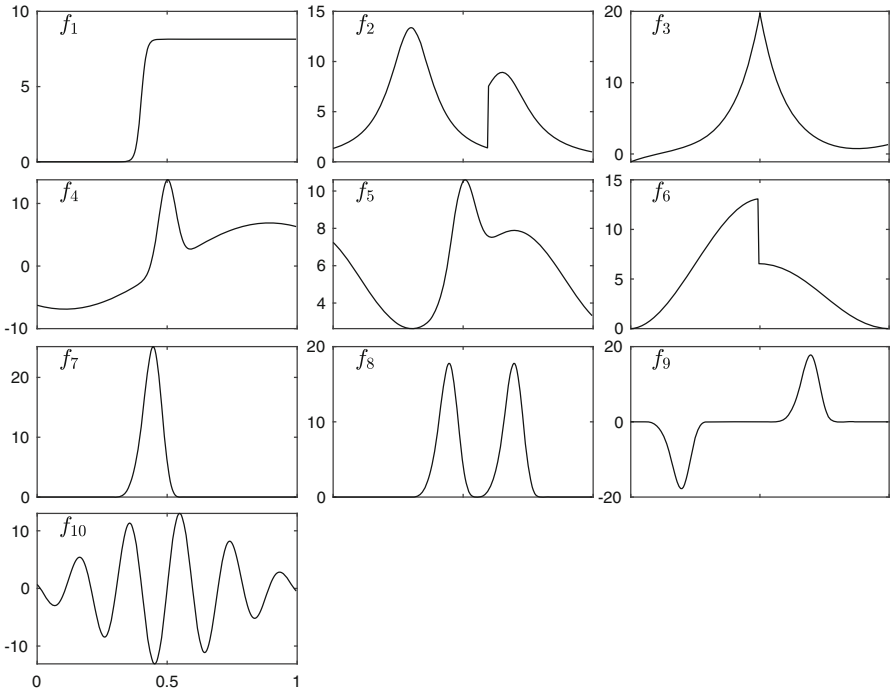

Fig. 3 Benchmark functions normalized to have SNR $= 100$. The function name is indicated in the upper left corner of each panel. The abscissa in each panel is identical to the one showing f_{10}

where f is the true function in the data and $\widehat{f_j}$ its estimate from the jth data realization. We use bootstrap with 10^4 independently drawn samples with replacement from the set $\{\|f - \widehat{f_j}\|^2\}$ to obtain the sampling error in RMSE.

A secondary metric that is useful is the sample mean of the number of knots in the best fit model. To recall, this is the average of $M_{\text{best}} \in \mathbb{N}_{\text{knots}}$ over the N_R data realizations, where M_{best} and $\mathbb{N}_{\text{knots}}$ were defined in Sect. 4. The error in M_{best} is estimated by its sample standard deviation.

5.4 Labeling scheme

Several design choices in `SHAPES` were described in Sect. 4. A useful bookkeeping device for keeping track of the many possible combinations of these choices is the labeling scheme presented in Table 2.

Following this labeling scheme, a string such as LP_100_0.1_50_FKM refers to the combination: lbest PSO; plain map from PSO search space to knots; SNR $= 100$ for the true function in the data; regulator gain $\lambda = 0.1$; maximum number of PSO iterations set to 50; end knots fixed; end B-splines retained; merging of knots allowed.

Table 2 Labeling scheme for a combination of design choices in SHAPES

PSO algorithm (Sect. 3)	L: lbest PSO	$*$
Knot map (Sect. 4.3)	P: Plain	C: Centered-monotonic
SNR (Eq. 30)	(Numerical)	
λ (Eq. 9)	(Numerical)	
N_{iter} (Number of PSO iterations)	(Numerical)	
End knots (Sect. 4.4)	F: Fixed	V: Variable
End B-splines (Sect. 4.5)	K: Keep	D: Drop
Knot merging (Sect. 4.2)	M: Merge	H: Heal

The string labeling a combination is formed by going down the rows of the table and (a) picking one letter from the last two columns of each row, or (b) inserting the value of a numerical quantity. Numerical values in the key string are demarcated by underscores on both sides. Thus, a key string looks like $Y_1Y_2\_X_3\_X_4\_X_5\_Y_6Y_7Y_8$ where Y_i and X_i stand for letter and numerical entries respectively, and i is the row number of the table starting from the top. We have left the possibility open for replacing lbest PSO with some other variant in the future. This is indicated by the '$*$' symbol in the top row

6 Computational considerations

The results in this paper were obtained with a code implemented entirely in MATLAB (Matlab Release 2018b). Some salient points about the code are described below.

The evaluation of B-splines uses the efficient algorithm given in de Boor (2001). Since our current B-spline code is not vectorized, it suffers a performance penalty in MATLAB. (We estimate that it is $\approx 50\%$ slower as a result.) Nonetheless, the code is reasonably fast: A single PSO run on a single data realization, for the more expensive case of SNR $= 100$, takes about 11 sec on an Intel Xeon (3.0 GHz) class processor. It is important to note that the run-time above is specific to the set, $\mathbb{N}_{\text{knots}}$, of models used. In addition, due to the fact that the number of particles breaching the search space boundary in a given PSO iteration is a random variable and that the fitness of such a particle is not computed, the actual run times vary slightly for different PSO runs and data realizations.

The only parallelization used in the current code is over the independent PSO runs. Profiling shows that $\approx 60\%$ of the run-time in a single PSO run is consumed by the evaluation of particle fitnesses, out of which $\approx 45\%$ is spent in evaluating B-splines. Further substantial saving in run-time is, therefore, possible if particle fitness evaluations are also parallelized. This dual parallelization is currently not possible in the MATLAB code but, given that we use $N_p = 40$ particles, parallelizing all N_p fitness evaluations can be expected to reduce the run-time by about an order of magnitude. However, realizing such a large number of parallel processes needs hardware acceleration using, for example, graphics Processing Units.

The operations count in the most time-consuming parts of the code (e.g., evaluating B-splines) scales linearly with the length of the data. Hence, the projected ratios above in run-time speed-up are not expected to change much with data length although the overall run-time will grow linearly.

The pseudorandom number streams used for the simulated noise realizations and in the PSO dynamical equations utilized built-in and well-tested default generators. The

PSO runs were assigned independent pesudorandom streams that were initialized, at the start of processing any data realization, with the respective run number as the seed. This (a) allows complete reproducibility of results for a given data realization, and (b) does not breach the cycle lengths of the pseudorandom number generators when processing a large number of data realizations.

7 Results

The presentation of results is organized as follows. Section 7.1 shows single data realizations and estimates for a subset of the benchmark functions. Section 7.2 analyzes the impact of the regulator gain λ on estimation. Sections 7.3 and 7.4 contain results for SNR $= 100$ and SNR $= 10$ respectively. Section 7.5 shows the effect of the bias correction step described in Sect. 4.1 on the performance of SHAPES for both SNR values. In Sect. 7.6, we compare the performance of SHAPES with two well-established smoothing methods, namely, wavelet-based thresholding and shrinkage (Donoho and Johnstone 1995), and smoothing spline with adaptive selection of the regulator gain (Craven and Wahba 1978). The former follows an approach that does not use splines at all, while the latter uses splines but avoids free knot placement. As such, they provide a good contrast to the approach followed in SHAPES.

In all applications of SHAPES, the set of models used was

$$\mathbb{N}_{\text{knots}} = \{5, 6, 7, 8, 9, 10, 12, 14, 16, 18\}.$$

The spacing between the models is set wider for higher knot numbers in order to reduce the computational burden involved in processing a large number of data realizations. In an application involving just a few realizations, a denser spacing may be used.

Figure 4 shows the performance of lbest PSO across the set of benchmark functions as a function of the parameter N_{iter}. Given that the fitness values do not change in a statistically significant way when going from $N_{\text{iter}} = 100$ to $N_{\text{iter}} = 200$ in the SNR $= 100$ case, we set it to the former as it saves computational cost. A similar plot of fitness values (not shown) for SNR $= 10$ is used to set $N_{\text{iter}} = 50$ for the SNR $= 10$ case.

7.1 Sample estimates

In Fig. 5, we show function estimates obtained with SHAPES for arbitrary single data realizations. While not statistically rigorous, this allows an initial assessment of performance when the SNR is sufficiently high. Also shown with each estimate is the location of the knots found by SHAPES.

For ease of comparison, we have picked only the benchmark functions (f_1 to f_6) used in Gálvez and Iglesias (2011). The SNR of each function matches the value one would obtain using the noise standard deviation tabulated in Gálvez and Iglesias (2011). Finally, the algorithm settings were brought as close as possible by (a) setting the regulator gain $\lambda = 0$, (b) using the plain map (c.f., Sect. 4.3), (c) keeping the end

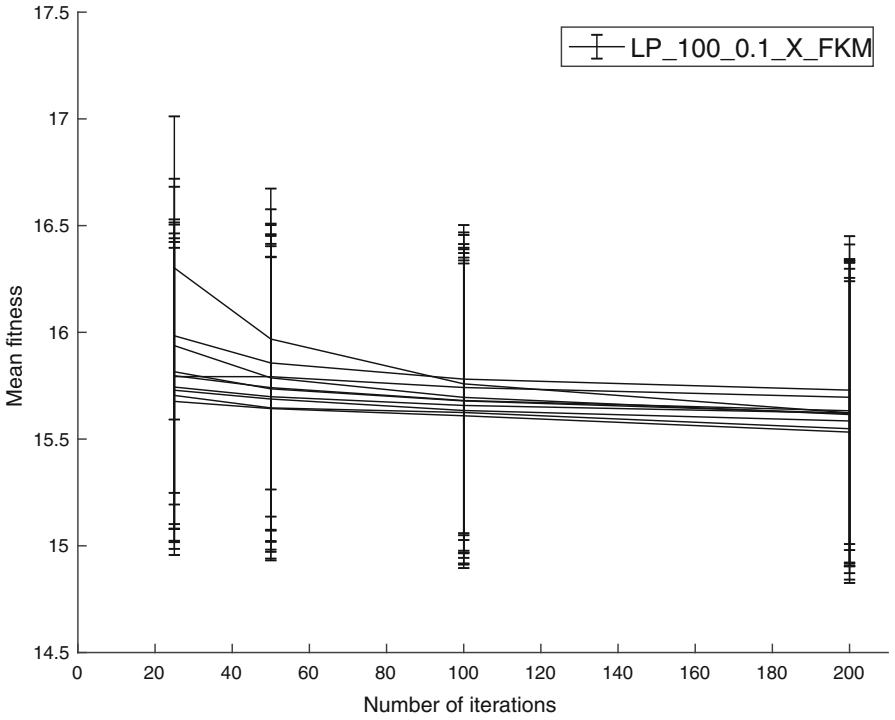

Fig. 4 Performance of *lbest* PSO as a function of the number of iterations, N_{iter}, to termination. Each curve corresponds to one of the benchmark functions at SNR $= 100$ and shows the mean fitness value as a function of N_{iter}. The mean fitness value is an average over $N_R = 1000$ data realizations of the fitness value corresponding to the best model (i.e., $F(M_{\text{best}}, r_{M_{\text{best}}})$ defined in Fig. 2). The error bars represent $\pm 1\sigma$ deviations where σ is the sample standard deviation. The other algorithm settings used for this plot can be read off from the key string shown in the legend using Table 2

knots fixed, and (d) allowing knots to merge. Differences remain in the PSO variant (and associated parameters) used and, possibly, the criterion used for merging knots.

We find that `SHAPES` has excellent performance at high SNR values: without any change in settings, it can fit benchmark functions ranging from spatially inhomogeneous but smooth to ones that have discontinuities. For the latter, `SHAPES` allows knots to coalesce into repeated knots in order to improve the fit at the location of the discontinuities. The sample estimates in Fig. 5 are visually indistinguishable from the ones given in Gálvez and Iglesias (2011). The same holds for the sample estimates given in Yoshimoto et al. (2003), which uses benchmark functions f_1 to f_3 and SNRs similar to Gálvez and Iglesias (2011).

7.2 Regulator gain

While the aim of restricting the number of knots in regression spline is to promote a smoother estimate, it is an implicit regularization that does not guarantee smoothness. In the absence of an explicit regularization, a fitting method based on free knot

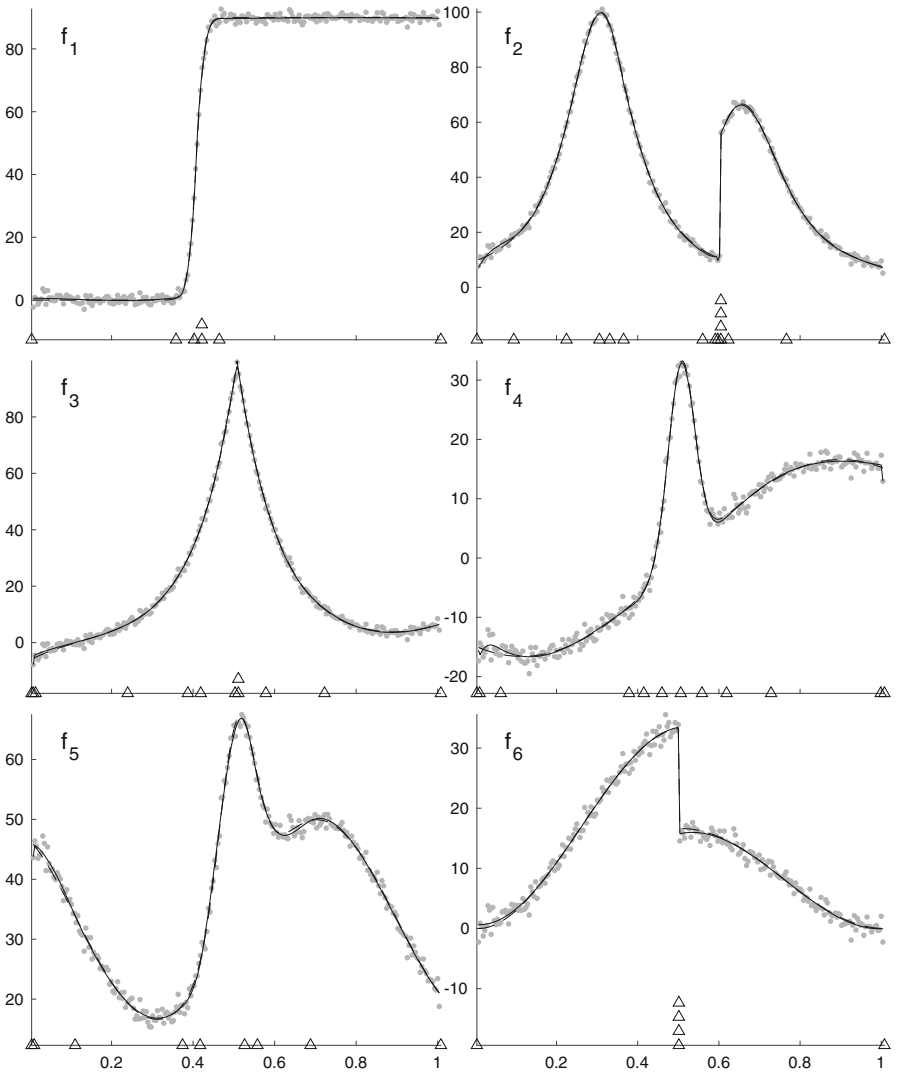

Fig. 5 Sample estimated functions for the benchmark functions f_1 to f_6. In each panel: the solid black curve is the estimated function; triangles show the locations of its knots (vertically stacked triangles denote repeated knots); the dashed black curve is the true function; gray dots represent the data realization. (In most cases, the solid and dashed curves are visually indistinguishable.) The SNRs (rounded to integer values) of the functions in order from f_1 to f_6 are 1104, 747, 506, 241, 633, and 254, respectively. The algorithm settings were LP_SNR_0_100_FKM (c.f., Table 2). Note that under these settings, the end B-splines are retained, which requires end knots to have the maximum allowed multiplicity. But this is a fixed multiplicity in each plot and not shown for clarity

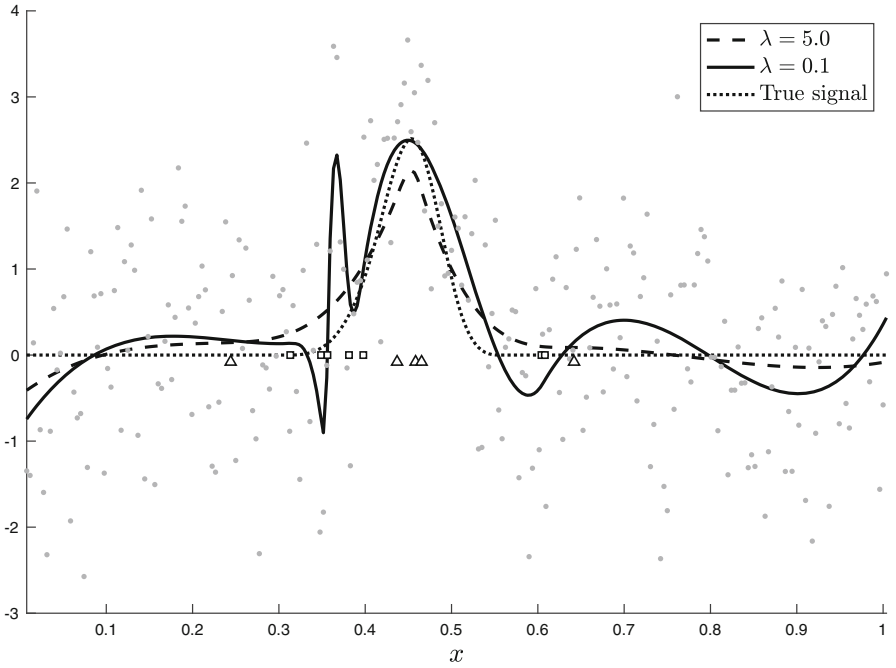

Fig. 6 Effect of regulator gain, λ, on estimation. The solid and dashed curves are the estimates obtained with $\lambda = 0.1$ and $\lambda = 5.0$ respectively, where λ is the regulator gain for the penalty term in Eq. (9). The true curve—benchmark function f_7 with SNR = 10—is shown with a dotted line and the gray dots show the data realization. The interior knots in the best model for $\lambda = 0.1$ and $\lambda = 5.0$ are shown as squares and triangles respectively. (Not shown here is an extra repeated knot for $\lambda = 0.1$.) Besides the difference in λ, the algorithm settings – LP_10_λ_50_FKM (see Table 2)—were identical for the two estimated curves

placement will exploit this loophole to form spurious clusters of knots that fit outliers arising from noise and overfit the data. This issue becomes increasingly important as the level of noise in the data increases.

Figure 6 illustrates how adding the penalized spline regulator helps mitigate this problem of knot clustering. Shown in the figure is one data realization and the corresponding estimates obtained with high and low values of the regulator gain λ. For the latter, sharp spikes appear in the estimate where the function value is not high but the noise values are. The method tries to fit out these values by putting more knots in the model and clustering them to form the spikes. Since knot clustering also needs large B-spline coefficients in order to build a spike, a larger penalty on the coefficients suppresses spurious spikes.

Figures 7 and 8 present a statistically more rigorous study of the effect of λ by examining the RMSE attained across the whole set of benchmark functions at different SNR values. In both figures, the RMSE is shown for identical algorithm settings except for λ, and in both we observe that increasing the regulator gain improves the RMSE. (The lone case where this is not true is addressed in more detail in Sect. 7.4.) The improvement becomes more pronounced as SNR is lowered. (The effect of λ on the

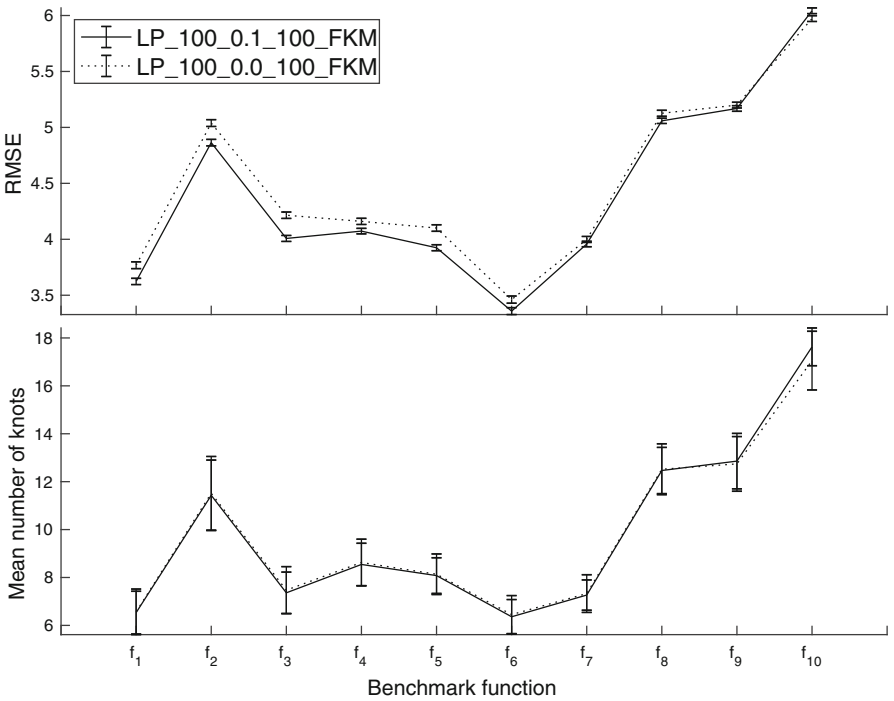

Fig. 7 Effect of regulator gain on (top panel) the root mean squared error (RMSE), and (bottom panel) the mean number of knots in the best model for SNR $= 100$ benchmark functions. In both panels, the solid and dotted curves correspond to $\lambda = 0.1$ and $\lambda = 0.0$ respectively. The other algorithm settings used for this plot can be read off from the key strings shown in the legend using Table 2. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where σ is the estimated standard deviation

number of knots in the best fit model at either SNR is within the sampling error of the simulation.)

The higher values of the regulator gains in Figs. 7 and 8—$\lambda = 0.1$ and $\lambda = 5.0$ for SNR $= 100$ and SNR $= 10$ respectively—were chosen according to the SNR. These pairings were chosen empirically keeping in mind that there is an optimum regulator gain for a given noise level. Too high a gain becomes counterproductive as it simply shrinks the estimate towards zero. Too low a value, as we have seen, brings forth the issue of knot clustering and spike formation. Since the latter is a more serious issue for a higher noise level, the optimum regulator gain shifts towards a correspondingly higher value.

7.3 Results for SNR = 100

We have already selected some of the algorithm settings in the preceding sections, namely, the number of iterations to use and the regulator gain for a given SNR. Before proceeding further, we need to decide on the remaining ones.

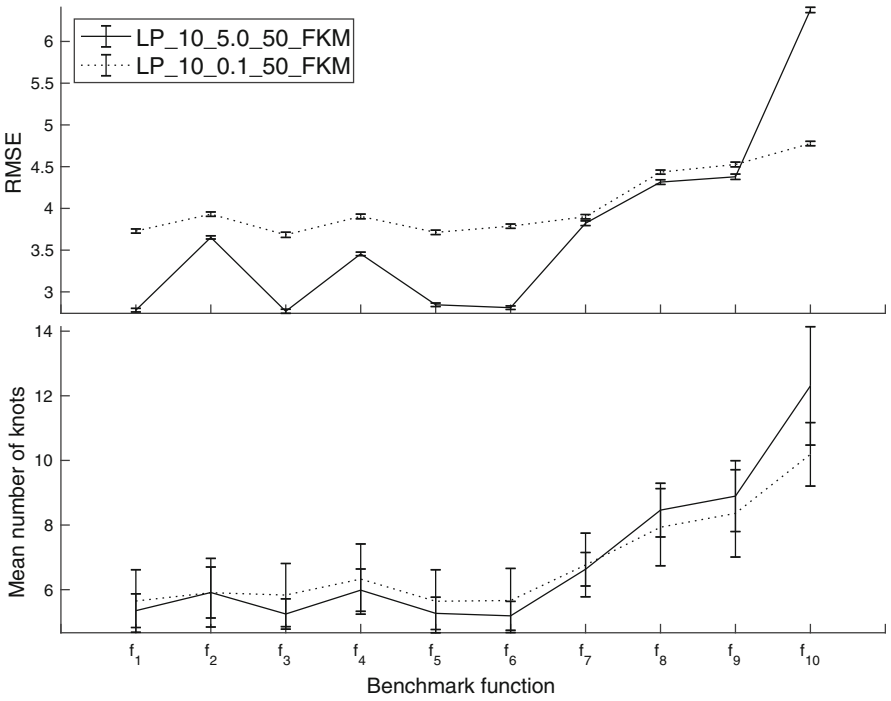

Fig. 8 Effect of regulator gain on (top panel) the root mean squared error (RMSE), and (bottom panel) the mean number of knots in the best model for SNR $= 10$ benchmark functions. In both panels, the solid and dotted curves correspond to $\lambda = 5.0$ and $\lambda = 0.1$ respectively. The other algorithm settings used for this plot can be read off from the key strings shown in the legend using Table 2. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where σ is the estimated standard deviation

For the SNR $= 100$ case, it is clear that the end knots and end B-splines must be retained because benchmark functions f_1 to f_6 do not all decay to zero and the noise level is not high enough to mask this behavior. Similarly, knot merging is an obvious choice because discontinuities in some of the benchmark functions are obvious at this SNR and they cannot be modeled under the alternative option of healing. The remaining choice is between the two knot maps: plain or centered-monotonic.

As shown in Fig. 9, the RMSE is distinctly worsened by the centered-monotonic map across all the benchmark functions. This map also leads to a higher number of knots in the best fit model although the difference is not as significant statistically. Thus, the clear winner here is the map in which knots are merged.

With all the design choices fixed, the performance of `SHAPES` can be examined. This is done in Figs. 10 and 11 where the point-wise sample mean and $\pm 2\sigma$ deviation, σ being the sample standard deviation, are shown for all the benchmark functions. Note that the level of noise now is much higher than the examples studied in Sect. 7.1.

It is evident from these figures that `SHAPES` is able to resolve different types of discontinuities as well as the locations of features such as peaks and sharp changes. In interpreting the error envelope, it should be noted that the errors at different points

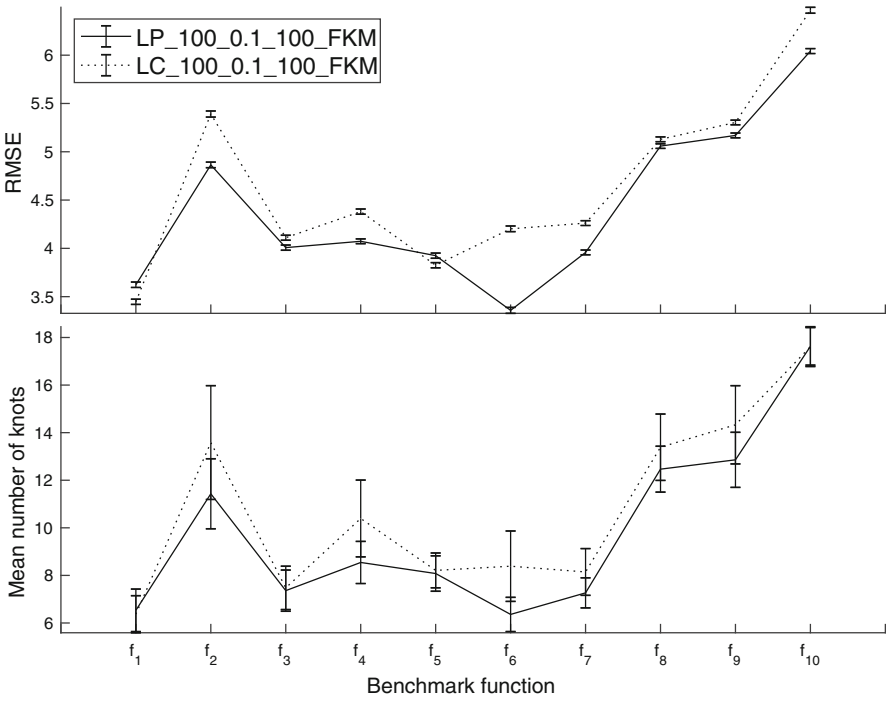

Fig. 9 Effect of the map used for transforming PSO search space coordinates to knots on (top panel) the root mean squared error (RMSE), and (bottom panel) the mean number of knots in the best model for SNR $= 100$ benchmark functions. The solid and dotted curves correspond to the plain and centered-monotonic maps respectively. The other algorithm settings used for this plot can be read off from the key strings shown in the legend using Table 2. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where σ is the estimated standard deviation

are strongly correlated, a fact not reflected in the point-wise standard deviation. Thus, a typical single estimate is not an irregular curve bounded by the error envelopes, as would be the case for statistically independent point-wise errors, but a smooth function. Nonetheless, the error envelopes serve to indicate the extent to which an estimate can deviate from the true function.

7.4 Results for SNR = 10

Here, we examine the case of high noise level at SNR $= 10$. Figures 12 and 13 show the point-wise sample mean and $\pm 2\sigma$ deviation, σ being the sample standard deviation, for all the benchmark functions. The algorithm settings used are the same as in Sect. 7.3 for the SNR $= 100$ case except for the regulator gain and the number of PSO iterations: $\lambda = 5.0$ and $N_{\text{iter}} = 50$ respectively.

Unlike the SNR $= 100$ case, the high noise level masks many of the features of the functions. For example, the discontinuities and the non-zero end values for f_1 to f_6 are washed out. Thus, the algorithm settings to use are not at all as clear cut as before.

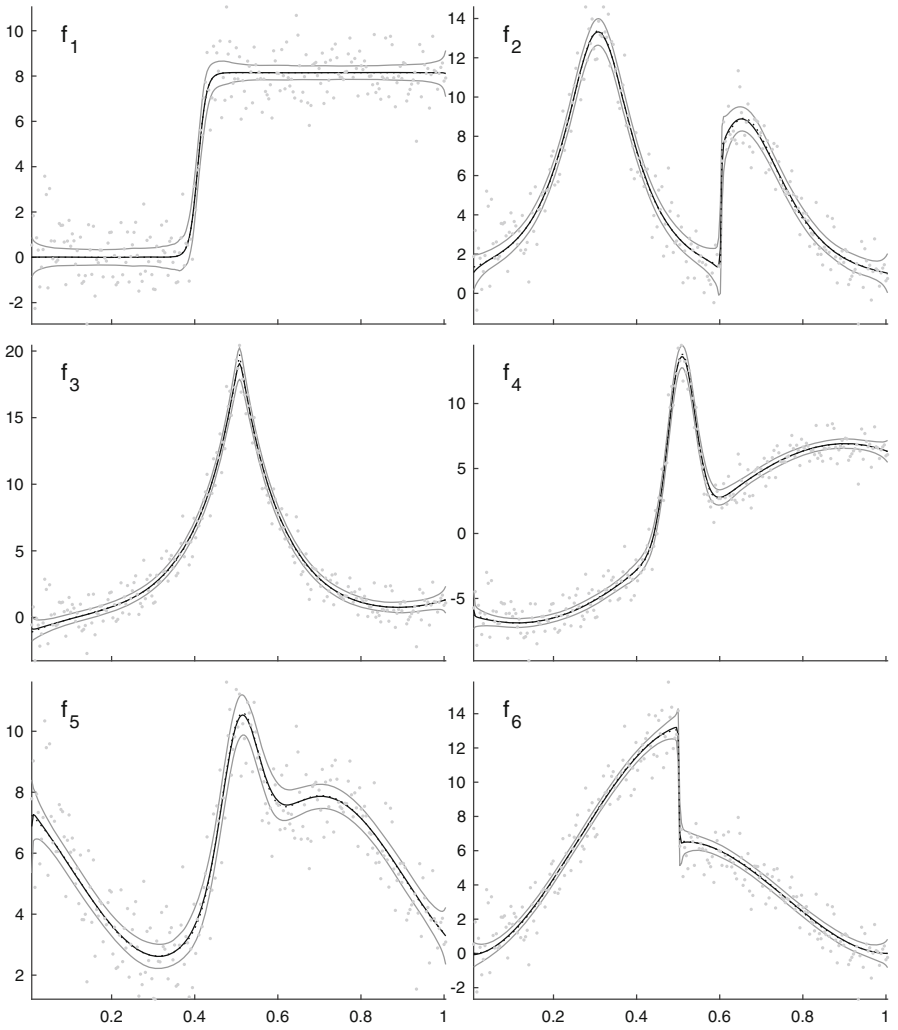

Fig. 10 Mean estimated functions (black curve) for benchmark functions f_1 to f_6 at SNR $= 100$. The true functions are shown as dotted curves but they are practically indistinguishable from the mean estimated functions. The gray curves show $\pm 2\sigma$ deviation from the mean function, where σ is the estimated standard deviation. The gray dots show an arbitrary data realization for the purpose of visualizing the noise level. The abscissa has the same range for each panel. The algorithm settings used are given by the key string LP_100_0.1_100_FKM, which can be expanded using Table 2

In fact, the results presented next show that alternative settings can show substantial improvements in some cases.

First, as shown in Fig. 14, the estimation of f_{10} actually improves significantly when the regulator gain is turned down to $\lambda = 0.1$. While this is the lone outlier in the general trend between regulator gain and RMSE (c.f., Fig. 8), it points to the importance of choosing the regulator gain adaptively rather than empirically as done in this paper.

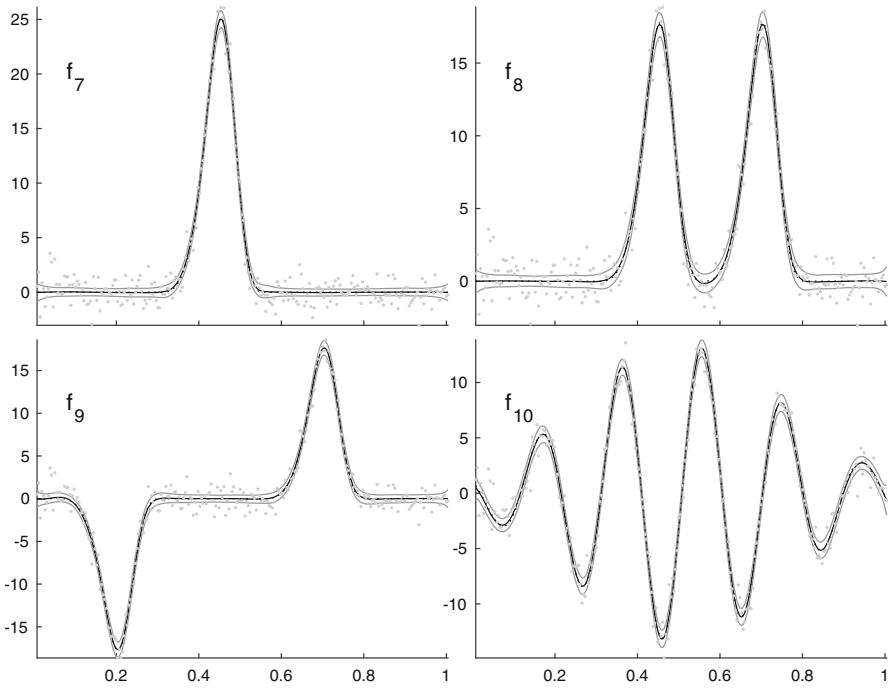

Fig. 11 Mean estimated functions (black curve) for benchmark functions f_7 to f_{10} at SNR $= 100$. The true functions are shown as dotted curves but they are practically indistinguishable from the mean estimated functions. The gray curves show $\pm 2\sigma$ deviation from the mean function, where σ is the estimated standard deviation. The gray dots show an arbitrary data realization for the purpose of visualizing the noise level. The abscissa has the same range for each panel. The algorithm settings used are given by the key string LP_100_0.1_100_FKM, which can be expanded using Table 2

Next, Fig. 15 examines the effect of the knot map and its interplay with fixing the end knots or allowing them to vary. Allowing the end knots to vary under either knot map leads to a worse RMSE but the number of knots required in the best fit model is reduced, significantly so for f_7 to f_{10}. A plausible explanation for this is that the high noise level masks the behavior of the functions at their end points, and freeing up the end knots allows `SHAPES` to ignore those regions and focus more on the ones where the function value is higher relative to noise.

Under a given end knot condition, the centered-monotonic map always performs worse in Fig. 15 albeit the difference is statistically significant for only a small subset of the benchmark functions. Remarkably, this behavior is reversed for some of the benchmark functions when additional changes are made to the design choices. Figure 16 shows the RMSE when the centered-monotonic map and variable end knots are coupled with the dropping of end B-splines and healing of knots. Now, the performance is better for functions f_7 to f_{10} relative to the best algorithm settings found from Fig. 15: not only is there a statistically significant improvement in the RMSE for these functions but this is achieved with a substantially smaller number of knots.

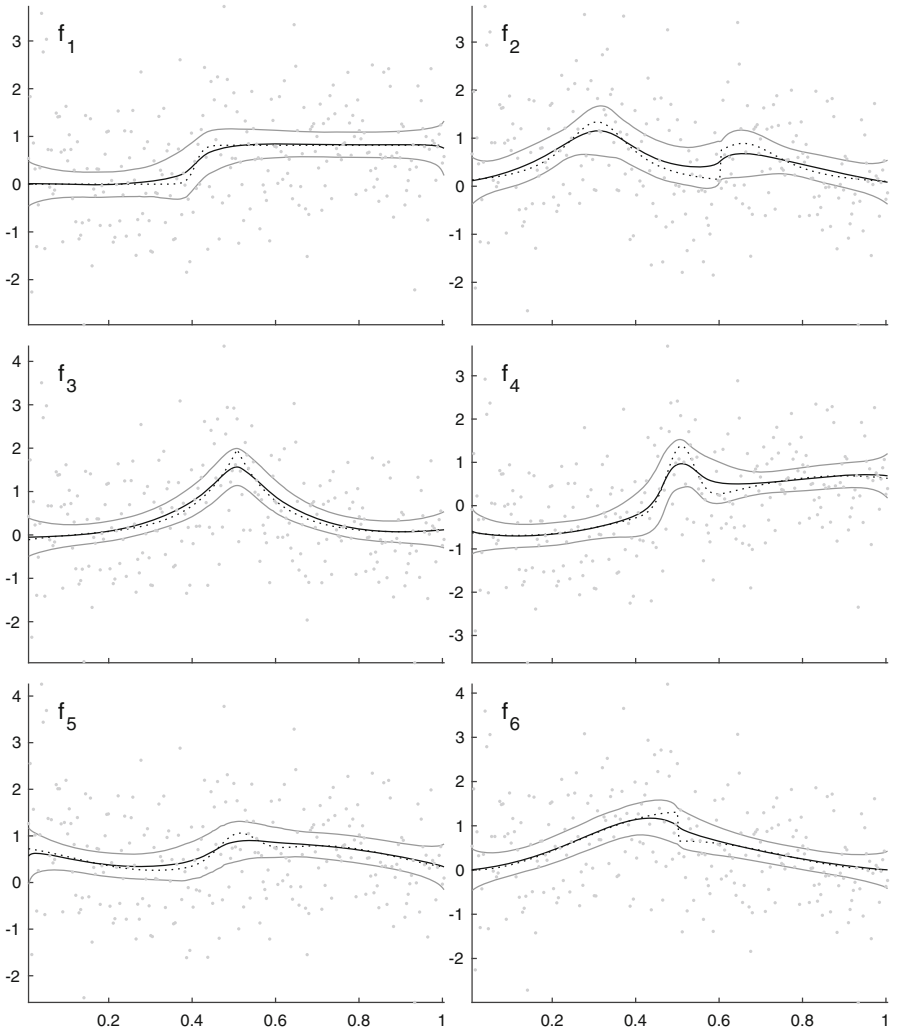

Fig. 12 Mean estimated functions (black curve) for benchmark functions f_1 to f_6 at SNR $= 10$. The true functions are shown as dotted curves. The gray curves show $\pm 2\sigma$ deviation from the mean function, where σ is the estimated standard deviation. The gray dots show an arbitrary data realization for the purpose of visualizing the noise level. The abscissa has the same range for each panel. The algorithm settings used are given by the key string LP_10_5_50_FKM, which can be expanded using Table 2

This improvement comes at the cost, however, of significantly worsening the RMSE for the remaining benchmark functions.

7.5 Effect of bias correction

Figure 17 shows the effect of using the bias correction step described in Sect. 4.1 on RMSE. We see that bias correction reduces the RMSE for some of the benchmark

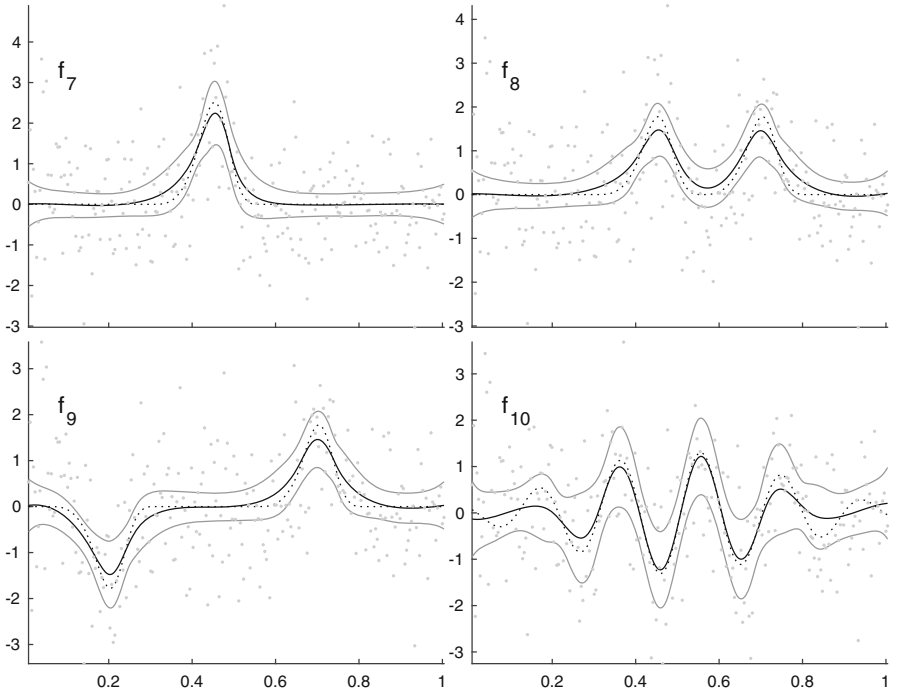

Fig. 13 Mean estimated functions (black curve) for benchmark functions f_7 to f_{10} at SNR $= 10$. The true functions are shown as dotted curves. The gray curves show $\pm 2\sigma$ deviation from the mean function, where σ is the estimated standard deviation. The gray dots show an arbitrary data realization for the purpose of visualizing the noise level. The abscissa has the same range for each panel. The algorithm settings used are given by the key string LP_10_5_50_FKM, which can be expanded using Table 2

functions, namely f_7 to f_{10}, and that the reduction is more at higher SNR for f_{10}. For the remaining benchmark functions, bias correction makes no difference to the RMSE.

7.6 Comparison with other methods

In this section, we compare the performance of `SHAPES` with `WaveShrink` (Donoho and Johnstone 1995) and smoothing spline (Reinsch 1967; Craven and Wahba 1978) as implemented in R Core Team (2019) (called `R:sm.spl` in this paper). The former is taken from the Matlab-based package `WaveLab` (Huo 2000) and it performs smoothing by thresholding the wavelet coefficients of the given data and applying nonlinear shrinkage to threshold-crossing coefficients. For both of these methods, we use default values of their parameters with the following exceptions: for `WaveShrink` we used the "Hybrid" shrinkage method, while for `R:sm.spl` we use GCV to determine the regulator gain. For reproducibility, we list the exact commands used to call these methods:

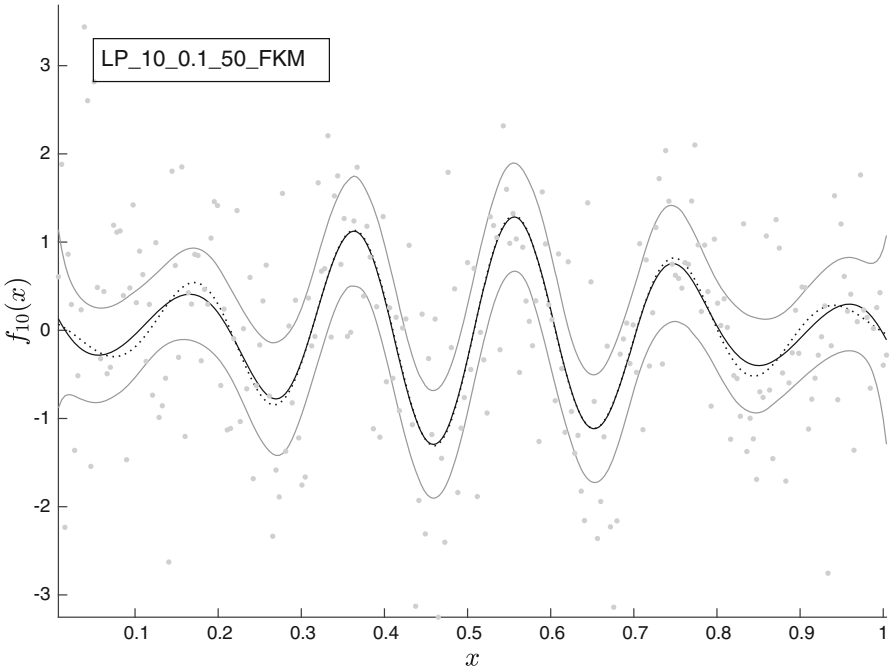

Fig. 14 The mean estimated function (black) for benchmark function f_{10} at SNR $= 10$ with regulator gain $\lambda = 0.1$. (The dotted curve shows the true function.) The gray curves show $\pm 2\sigma$ deviation from the mean function, where σ is the estimated standard deviation. The gray dots show an arbitrary data realization for the purpose of visualizing the noise level. The other algorithm settings can be read off from the key shown in the plot legend using Table 2

– `WaveShrink(Y,'Hybrid',L)`: `Y` is the data to be smoothed and `L` is the coarsest resolution level of the discrete wavelet transform of `Y`.
– `smooth.spline(X,Y,cv=FALSE)`: `X` is the set of predictor values, `Y` is the data to be smoothed, and `cv=FALSE` directs the code to use GCV for regulator gain determination.

Each method above was applied to the same dataset as used for producing Fig. 10. Statistical summaries, namely, the mean estimated function, the $\pm 2\sigma$ deviation from the mean, and RMSE were obtained following the same procedure as described for `SHAPES`. A crucial detail: when applying `WaveShrink`, we use `L` $\in \{1, 2, \ldots, 6\}$ and pick the one that gives the lowest RMSE.

The results of the comparison are shown in Table 3, Figs. 18 and 19. Table 3 shows the RMSE values attained by the methods for the benchmark functions f_1 to f_6, all normalized to have SNR $= 100$. Figure 18 shows more details for the benchmark functions that produce the best and worst RMSE values for `WaveShrink`. Similarly, Fig. 19 corresponds to the best and worst benchmark functions for `R:sm.spl`.

While noted in the caption of Fig. 18, it is worth re-emphasizing here that the error envelopes of `SHAPES` shown in Figs. 18 and 19 are computed relative to the mean estimated function of the method being compared to, not to that of `SHAPES` itself.

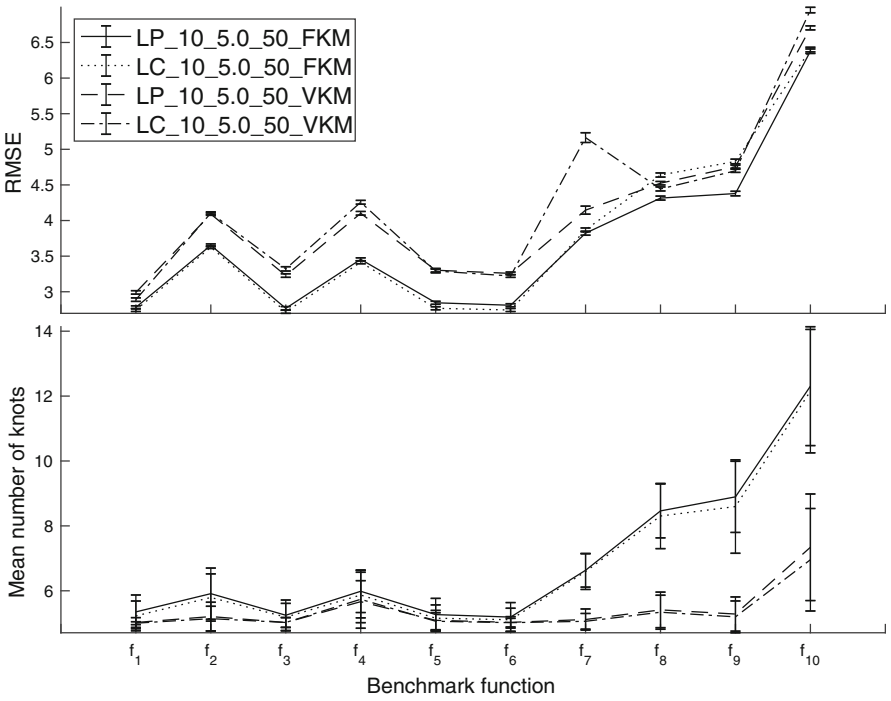

Fig. 15 Comparison of plain and centered-monotonic maps under fixed (F) and variable (V) end knot conditions at SNR $= 10$. The top and bottom panels respectively show RMSE and mean number of knots in the best model. The other algorithm settings used for this plot can be read off from the key strings shown in the legend using Table 2. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where σ is the estimated standard deviation

This modification eliminates visual confusion caused by the different biases (i.e., mean estimated functions) of the methods. However, the bias curves shown separately in these figures do use the respective mean estimated function for each method. The actual mean estimated functions and corresponding error envelopes of `SHAPES` can be seen in Fig. 10.

From Table 3, we see that `SHAPES` has the lowest RMSE in all cases. Figures 18 and 19 show that this arises from `SHAPES` generally having both a lower estimation variance (where its error envelope nests within that of the other methods) as well as a lower bias. Typically, `R:sm.spl` has a lower variance than `SHAPES` around stationary points of the true function but the difference is marginal. In some cases, such as f_3 and f_6, the bias in the `SHAPES` estimate is significantly lower than that of either `WaveShrink` or `R:sm.spl`. In general, `WaveShrink` estimates are less smooth than those from either `SHAPES` or `R:sm.spl`. This is manifested, for example, in the rougher behavior of the mean estimated function from `WaveShrink`. Both `WaveShrink` and `R:sm.spl` have a much poorer resolution of the jump discontinuity in f_6 compared to `SHAPES` (c.f., Fig. 10).

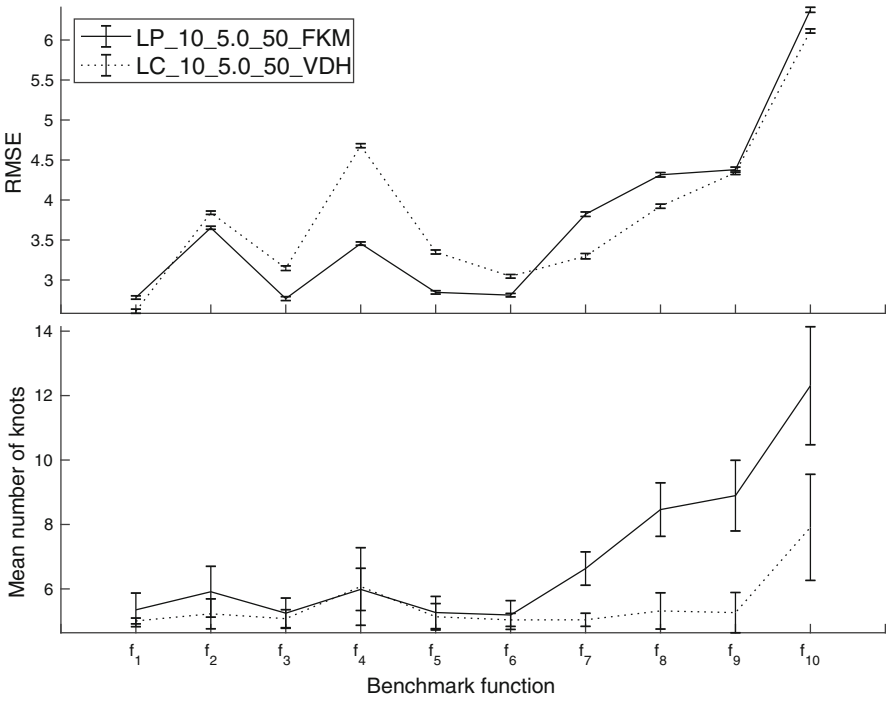

Fig. 16 Comparison of plain and centered-monotonic maps under the FKM and VDH algorithm settings respectively at SNR = 10. See Table 2 for the meaning of these and other algorithm settings given by the key strings in the legend. The top and bottom panels respectively show RMSE and mean number of knots in the best model. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where σ is the estimated standard deviation

8 Conclusions

Our results show that the challenge of free knot placement in adaptive spline fitting is solvable. The most important element of the solution is the use of an effective metaheuristic for knot optimization. We have shown that lbest PSO is effective in this task. Considering the f_{10} benchmark function for example, the best model found by `SHAPES` reaches the vicinity of the highest number (= 18) of non-repeating knots considered in this paper. The good quality of the fit obtained for f_{10} shows that PSO was able to handle this high-dimensional optimization well.

Relative to the SNRs used commonly in the literature on adaptive spline fitting, the values of SNR used in this paper, namely SNR = 100 and SNR = 10, can be ranked respectively as being moderate to low. For the former, discontinuities in function values or derivatives were well localized by `SHAPES` in all the cases. At the same time, the smooth parts of the benchmark functions were also well estimated. The estimates from `SHAPES` for low SNR (= 10) had, naturally, more error. In particular, the noise level in all the data realizations was high enough to completely mask the presence of discontinuities and, thus, they were not well localized. Nonetheless, even with a conservative error envelope of $\pm 2\sigma$ around the mean estimated signal, the

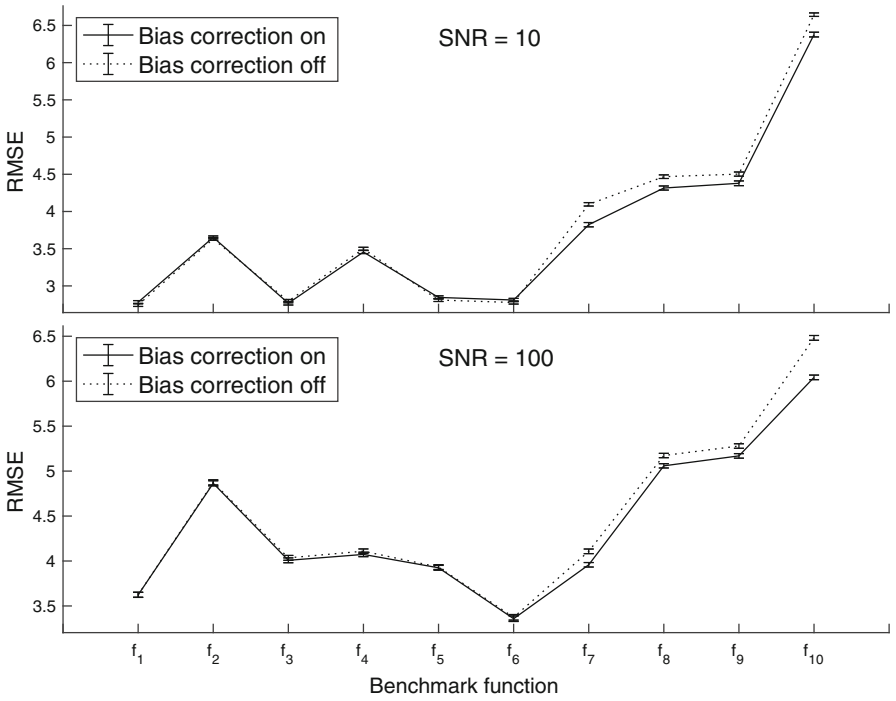

Fig. 17 The effect of bias correction on root mean squared error (RMSE) for SNR $= 10$ (top) and SNR $=$ 100 (bottom) benchmark functions. Solid and dotted curves in each panel correspond to bias correction switched on and off, respectively. The algorithm settings used were LP_SNR_λ_N_{iter}_FKM in all the cases with (top) $\lambda = 5.0$, $N_{\text{iter}} = 50$, and (bottom) $\lambda = 0.1$, $N_{\text{iter}} = 100$. These are the fiducial settings used for SNR $= 10$ and SNR $= 100$ in Sects. 7.4 and 7.3, respectively. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where σ is the estimated standard deviation

overall shape of the true function is visually clear in all the examples. This shows that the estimated functions are responding to the presence of the true function in the data.

While we have characterized the performance of SHAPES as an estimator in this paper, the observation made above for the low SNR case suggests that SHAPES may also be used to set up a hypotheses test. This could be based, say, on the fitness value returned by SHAPES. Note that, being a non-parametric method, SHAPES can handle functions with qualitatively disparate behaviors—from a simple change between two levels to oscillatory—without requiring any special tuning. Thus, such a hypotheses test would allow the detection of signals with a wide morphological range. This investigation is in progress.

The dependence of design choices on SNR, as elucidated in this paper, does not seem to have been fully appreciated in the literature on adaptive spline fitting, probably because the typical scenario considered is that of high SNR. While performance of SHAPES for SNR $= 100$ is found to be fairly robust to the design choices made, they have a non-negligible affect at SNR $= 10$. The nature of the true function also influences the appropriate algorithm settings for the latter case. Fortunately, the set-

Table 3 RMSE values for `SHAPES`, `WaveShrink`, and `R:sm.spl` obtained with the same dataset as used for Fig. 10

	`SHAPES`	`WaveShrink`	`R:sm.spl`
f_1	3.62	7.96 (4)	4.91
f_2	4.86	7.47 (4)	7.39
f_3	4.01	5.82 (3)	5.52
f_4	4.07	8.11 (4)	5.24
f_5	3.92	6.21 (3)	4.19
f_6	3.36	6.93 (3)	7.62

The benchmark functions used are f_1 to f_6 at SNR $= 100$. In the case of `WaveShrink`, the RMSE is the lowest attained over different values of the parameter `L`. The best value of `L` is shown parenthetically

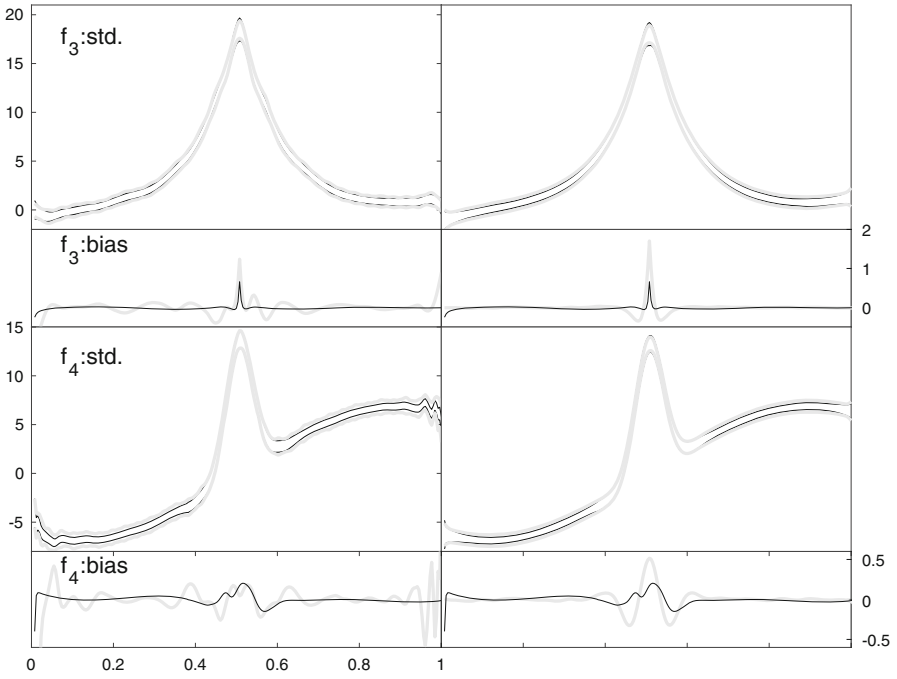

Fig. 18 Comparison of `SHAPES` with alternate methods, `WaveShrink` and `R:sm.spl`, for benchmark functions f_3 and f_4 at SNR $= 100$. The left column of figure panels is associated with `WaveShrink` and the right with `R:sm.spl`. In each column, (i) the benchmark function used is indicated in each panel, (ii) black curves correspond to `SHAPES`, and (iii) gray curves to the alternate algorithm. For each benchmark function, there are two panels: (i) The one labeled "std" shows the $\pm 2\sigma$ error envelopes relative to the estimated mean signal from the alternate method; (ii) the one labeled "bias" shows the difference between the true function and the estimated mean signal from each method. The abscissa has the same range for each panel. The ordinate values are identical across the panels in a given row. The dataset used and the algorithm settings for `SHAPES` are the same as in Fig. 10

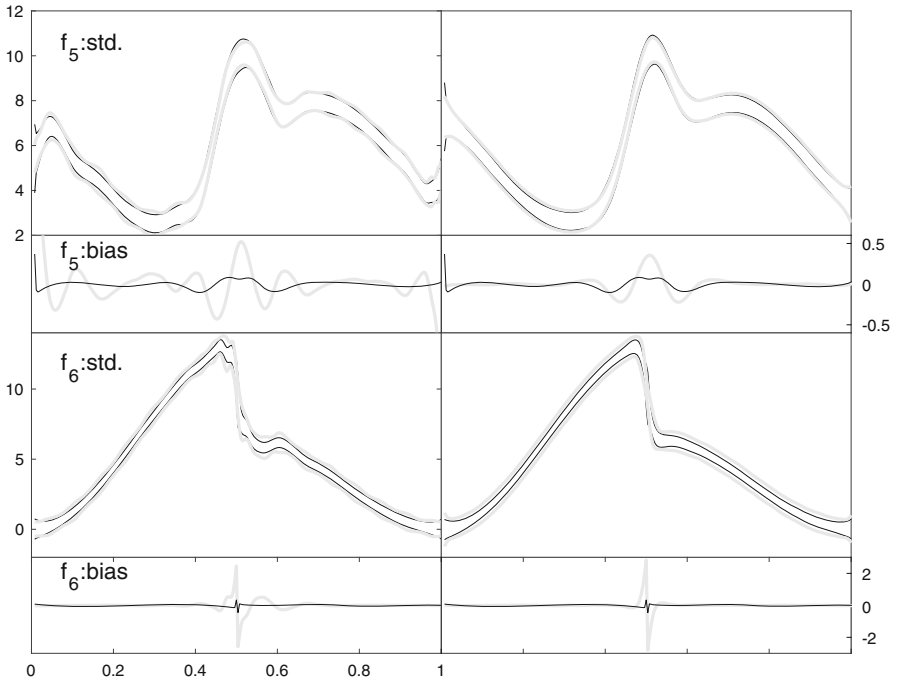

Fig. 19 Same as Fig. 18 except for the change in benchmark functions to f_5 and f_6

tings were found to depend on only some coarse features of a function, such as its behavior at data boundaries (f_1 to f_6), whether it is transient (f_7 to f_9), or whether it is oscillatory (f_{10}). Such features are often well-known in a real-world application domain: it is unusual to deal with signals that have discontinuities as well as signals that are smooth and transient in the same application. Hence, in most such cases, it should be straightforward to pick the best settings for SHAPES.

The inclusion of a penalized spline regulator was critical in SHAPES for mitigating the problem of knot clustering. For all except one (f_{10}) benchmark functions considered here, the regulator gain was determined empirically by simply examining a few realizations at each SNR with different values of the regulator gain λ. Ideally, however, λ should be determined adaptively from given data using a method such as GCV. The case of f_{10} at SNR $= 10$ provides a particularly good test bed in this regard: while $\lambda = 5.0$ worked well for the other benchmark functions at SNR $= 10$, the RMSE for f_{10} improved significantly when the gain was lowered to $\lambda = 0.1$. Thus, any method for determining λ adaptively must be able to handle this extreme variation. We leave the additional refinement of using an adaptive regulator gain in SHAPES to future work.

The extension of SHAPES to multi-dimensional splines and longer data lengths is the next logical step in its development. It is likely that extending SHAPES to these higher complexity problems will require different PSO variants than the one used here.

The codes used in this paper for the FKM option (c.f., Sect. 7.3) are available in a `GitHub` repository at the https://github.com/mohanty-sd/SHAPES.git

Acknowledgements The contribution of S.D.M. to this paper was supported by National Science Foundation (NSF) Grant PHY-1505861. The contribution of E.F. to this paper was supported by NSF Grant PHY-1757830. We acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin (www.tacc.utexas.edu) for providing HPC resources that have contributed to the research results reported within this paper.

References

Akaike H (1998) Information theory and an extension of the maximum likelihood principle. In: Selected papers of Hirotugu Akaike. Springer, Berlin, pp 199–213

Bratton D, Kennedy J, (2007) Defining a standard for particle swarm optimization. In: Swarm intelligence symposium (SIS). IEEE, pp 120–127

Burchard HG (1974) Splines (with optimal knots) are better. Appl Anal 3(4):309–319

Claeskens G, Krivobokova T, Opsomer JD (2009) Asymptotic properties of penalized spline estimators. Biometrika 96(3):529–544

Craven P, Wahba G (1978) Smoothing noisy data with spline functions. Numer Math 31(4):377–403

Curry HB, Schoenberg IJ (1947) On spline distributions and their limits—the polya distribution functions. Bull Am Math Soc 53(11):1114–1114

de Boor C (1972) On calculating with b-splines. J Approx Theory 6(1):50–62

de Boor C (2001) A practical guide to splines (applied mathematical sciences). Springer, Berlin

Denison D, Mallick B, Smith A (1998) Automatic Bayesian curve fitting. J R Stat Soc Ser B (Stat Methodol) 60(2):333–350

DiMatteo I, Genovese CR, Kass RE (2001) Bayesian curve-fitting with free-knot splines. Biometrika 88(4):1055–1071

Donoho DL, Johnstone IM (1995) Adapting to unknown smoothness via wavelet shrinkage. J Am Stat Assoc 90(432):1200–1224

Eilers PH, Marx BD (1996) Flexible smoothing with b-splines and penalties. Stat Sci 11:89–102

Eilers PH, Marx BD, Durbán M (2015) Twenty years of p-splines. SORT: Stat Oper Res Trans 39(2):0149–186

Engelbrecht AP (2005) Fundamentals of computational swarm intelligence, vol 1. Wiley, Chichester

Friedman JH (1991) Multivariate adaptive regression splines. Ann Stat 19(1):1–67

Friedman JH, Silverman BW (1989) Flexible parsimonious smoothing and additive modeling. Technometrics 31(1):3–21

Gálvez A, Iglesias A (2011) Efficient particle swarm optimization approach for data fitting with free knot b-splines. Comput Aided Des 43(12):1683–1692

Goepp V, Bouaziz O, Nuel G (2018) Spline regression with automatic knot selection. arXiv preprint arXiv:1808.01770

Golub GH, Heath M, Wahba G (1979) Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics 21(2):215–223

Green PJ (1995) Reversible jump markov chain monte carlo computation and bayesian model determination. Biometrika 82(4):711–732

Härdle W (1990) Applied nonparametric regression, no. 19. Cambridge University Press, Cambridge

Huo X, Duncan M, Levi O, Buckheit J, Chen S, Donoho D, Johnstone I, (2000). http://statweb.stanford.edu/~wavelab/Wavelab_850/AboutWaveLab.pdf

Jupp DL (1978) Approximation to data by splines with free knots. SIAM J Numer Anal 15(2):328–343

Kang H, Chen F, Li Y, Deng J, Yang Z (2015) Knot calculation for spline fitting via sparse optimization. Comput Aided Des 58:179–188

Kennedy J (1999) Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), vol 3. IEEE, pp 1931–1938

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, Perth, WA, Australia, vol 4. IEEE, p 1942

Krivobokova T (2013) Smoothing parameter selection in two frameworks for penalized splines. J R Stat Soc Ser B (Stat Methodol) 75(4):725–741
Krivobokova T, Crainiceanu CM, Kauermann G (2008) Fast adaptive penalized splines. J Comput Graph Stat 17(1):1–20
Lee T (2002) On algorithms for ordinary least squares regression spline fitting: a comparative study. J Stat Comput Simul 72(8):647–663
Leung C (2015) Estimation of unmodeled gravitational wave transients with spline regression and particle swarm optimization. SIAM Undergraduate Research Online (SIURO) 8
Li M, Yan Y (2006) Bayesian adaptive penalized splines. J Acad Bus Econ 2:129–141
Lindstrom MJ (1999) Penalized estimation of free-knot splines. J Comput Graph Stat 8(2):333–352
Liu Z, Guo W (2010) Data driven adaptive spline smoothing. Stat Sin 20:1143–1163
Luo Z, Wahba G (1997) Hybrid adaptive splines. J Am Stat Assoc 92(437):107–116
Luo J, Kang H, Yang Z (2019) Knot calculation for spline fitting based on the unimodality property. Comput Aided Geom Des 73:54–69
Lyche T, Mørken K (1988) A data-reduction strategy for splines with applications to the approximation of functions and data. IMA J Numer Anal 8(2):185–208
Matlab Release (2018b) The MathWorks, Inc., Natick, MA, United States. http://www.mathworks.com/
Mitchell M (1998) An introduction to genetic algorithms by Melanie Mitchell. MIT Press, Cambridge
Miyata S, Shen X (2003) Adaptive free-knot splines. J Comput Graph Stat 12(1):197–213
Mohanty SD (2012) Particle swarm optimization and regression analysis I. Astron Rev 7(2):29–35
Mohanty SD (2017) Spline based search method for unmodeled transient gravitational wave chirps. Phys Rev D 96:102008. https://doi.org/10.1103/PhysRevD.96.102008
Mohanty SD (2018) Swarm intelligence methods for statistical regression. Chapman and Hall/CRC, London
Normandin ME, Mohanty SD, Weerathunga TS (2018) Particle swarm optimization based search for gravitational waves from compact binary coalescences: performance improvements. Phys Rev D 98(4):044029
Park H, Lee J-H (2007) B-spline curve fitting based on adaptive curve refinement using dominant points. Comput Aided Des 39(6):439–451
Pittman J (2002) Adaptive splines and genetic algorithms. J Comput Graph Stat 11(3):615–638
R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2019). https://www.R-project.org/
Ramsay J, Silverman BW (1997) Functional data analysis. Springer series in statistics. Springer, New York
Reinsch CH (1967) Smoothing by spline functions. Numer Math 10(3):177–183
Ruppert D (2002) Selecting the number of knots for penalized splines. J Comput Graph Stat 11(4):735–757
Ruppert D, Carroll RJ (2000) Theory and methods: spatially-adaptive penalties for spline fitting. Austr N Z J Stat 42(2):205–223
Ruppert D, Wand MP, Carroll RJ (2003) Semiparametric regression, vol 12. Cambridge University Press, Cambridge
Scheipl F, Kneib T (2009) Locally adaptive bayesian p-splines with a normal-exponential-gamma prior. Comput Stat Data Anal 53(10):3533–3552
Smith PL (1982) Curve fitting and modeling with splines using statistical variable selection techniques. Technical report, NASA, Langley Research Center, Hampton, VA, report NASA, 166034
Solis FJ, Wets RJ-B (1981) Minimization by random search techniques. Math Oper Res 6(1):19–30
Stone CJ, Hansen MH, Kooperberg C, Truong YK et al (1997) Polynomial splines and their tensor products in extended linear modeling: 1994 Wald memorial lecture. Ann Stat 25(4):1371–1470
Storlie CB, Bondell HD, Reich BJ (2010) A locally adaptive penalty for estimation of functions with varying roughness. J Comput Graph Stat 19(3):569–589
Ülker E, Arslan A (2009) Automatic knot adjustment using an artificial immune system for b-spline curve approximation. Inf Sci 179(10):1483–1494
Wahba G (2002) In discussion Wavelet shrinkage: Asymptopia? with discussion and a reply by Donoho DL, Johnstone IM, Kerkyacharian G, Picard D, J R Stat Soc Ser B 57: 545–564
Wahba G (1990) Spline models for observational data, vol 59. SIAM, Philadelphia
Wand MP (2000) A comparison of regression spline smoothing procedures. Comput Stat 15(4):443–462
Wang X, Du P, Shen J (2013) Smoothing splines with varying smoothing parameter. Biometrika 100(4):955–970
Wegman EJ, Wright IW (1983) Splines in statistics. J Am Stat Assoc 78(382):351–365
Wold S (1974) Spline functions in data analysis. Technometrics 16(1):1–11

Yang L, Hong Y (2017) Adaptive penalized splines for data smoothing. Comput Stat Data Anal 108:70–83
Yoshimoto F, Harada T, Yoshimoto Y (2003) Data fitting with a spline using a real-coded genetic algorithm. Comput Aided Des 35(8):751–760
Zhou S, Shen X (2001) Spatially adaptive regression splines and accurate knot selection schemes. J Am Stat Assoc 96(453):247–259