Adaptive spline fitting with particle swarm optimization

Soumya D. Mohanty†
Department of Physics and Astronomy, The University of Texas Rio Grande Valley,
One West University Blvd., Brownsville, TX 78520, USA

Ethan Fahnestool†
Department of Physics and Astronomy, The University of Rochester, 500 Wilson Blvd., Rochester, NY 14627, USA
(Dated: March 2019)

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.

I. 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. Splines play an important role in non-parametric regression, 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 that the quality of a spline fit depends significantly on the locations of 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 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). 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.

In attempts at solving the optimization challenge directly, general purpose stochastic optimization algorithms (metaheuristics) such as Genetic Algorithm (GA) or those based on Markov Chain Monte Carlo (MCMC), have been studied. 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 is 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 [17], and independently in [18], that Particle Swarm Optimization (PSO), 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 [17] albeit for examples with very low noise level.

A question that has not been addressed adequately so far, either with addition-deletion schemes or with metaheuristics, is that of performance of adaptive spline fitting on data with a high noise level. There is a growing need for extending non-parametric regression to highly
noisy data in many scientific fields. This is especially acute in the case of gravitational wave (GW) searches that have recently achieved success [20] (and the 2017 Nobel prize in Physics). The most important target for GW searches, where the data is a noisy time series, are transient chirps: signals whose instantaneous frequency, \( \omega(t) \), and amplitude, \( a(t) \), evolve adiabatically on the timescale of the instantaneous period, \( 1/\omega(t) \). For GW sources having a sufficiently accurate theoretical model for predicting \( \omega(t) \) and \( a(t) \), parametric regression methods such as matched filtering [21] can be used and the associated data analysis problem is well under control (except for computationally issues). However, if either \( \omega(t) \) or \( a(t) \), or both, are not known a priori, non-parametric regression methods must be used and the high level of noise in GW data makes this an especially challenging task. In this context, the use of a spline-based non-parametric regression method (SEECCR) was shown in [22] to be much more effective than searches based on time-frequency analysis. In SEECCR, both \( \omega(t) \) and \( a(t) \) are modeled by splines, and knot optimization using PSO is used for fitting \( \omega(t) \).

It was found in [22], and later in a simplified model problem [23], 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 [23] by introducing a penalized spline regularization [24] of the regression problem. Penalized spline regression has also been used in combination with knot addition [4] but its role there – suppression of numerical instability arising from a large numbers of knots – is very different.

The progress described above in adaptive spline fitting using free knot placement 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 penalized spline 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 penalized spline regulator using a least squares derived rescaling. The elements of SHAPES outlined above match most closely the method that was used in [23] for the analysis of a single example. However, the crucial step of bias reduction was missing there. (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 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. Sec. III provides a brief review of pertinent topics in spline fitting. The PSO metaheuristic and the particular variant used in this paper are reviewed in Sec. IV. Details of SHAPES are described in Sec. V along with the principal design choices. The setup used for our simulations is described in Sec. VI. Computational aspects of SHAPES are addressed in Sec. VII. This is followed by the presentation of results in Sec. VIII. Our conclusions are summarized in Sec. IX.

II. FITTING SPLINES TO NOISY DATA

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

\[
\hat{f} = \arg \min_f \sum_{i=0}^{N-1} (y_i - f(x_i))^2 + \lambda \int_0^1 dx (f''(x))^2
\]

where \( \lambda \geq 0 \), the regulator gain, determines the strength of the smoothness requirement. It can be shown that \( \hat{f} \) belongs to the family of cubic splines defined by \( \{x_i\} \) as the set of knots. Consequently, \( \hat{f} \) is known as the smoothing spline estimate [25].

It is empirically observed that the large number of knots – all the values in \( \{x_i\} \) – in spline smoothing results in a high degree of freedom and overfitting. Restricting the number of knots to be \( \ll N \) is one way to control overfitting and this leads to the regression spline [3] estimate. Analysis of the asymptotic properties of smoothing and regression spline estimates also lends support to this modification [20].

A. B-spline functions

For a given set of knots \( \bar{b} = (b_0, b_1, \ldots, b_{M-1}) \), \( b_i \in [0, 1] \), \( b_{i+1} > b_i \), and given order \( k \) of the spline polynomi-
als, 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 \([27]\).

In this paper, we need B-spline functions for the more general case of a knot sequence \(\tau = (\tau_0, \tau_1, \ldots, \tau_{P-1})\), \(\tau_{i+1} \geq \tau_i\), 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 \([28]\) given below provide an efficient way to compute the set of B-spline functions, \(\{B_{i,k}(x; \tau)\}\), for any given order. The recursions start with B-splines of order 1, which are piecewise constant functions

\[
B_{j,1}(x; \tau) = \begin{cases} 
1, & \tau_j \leq x < \tau_{j+1} \\
0, & \text{else}
\end{cases}
\]

For \(2 \leq k' \leq k\),

\[
\begin{align*}
B_{j,k'}(x) &= \omega_{j,k'}(x)B_{j,k'-1}(x) + \gamma_{j+1,k'}(x)B_{j+1,k'-1}(x); \\
\omega_{j,k'}(x) &= \begin{cases} 
\frac{x-\tau_j}{\tau_{j+k'-1} - \tau_j}, & \tau_{j+k'-1} \neq \tau_j \\
0, & \tau_{j+k'-1} = \tau_j
\end{cases}, \\
\gamma_{j,k'}(x) &= \begin{cases} 
1 - \omega_{j,k'}(x), & \tau_{j+k'-1} \neq \tau_j \\
0, & \tau_{j+k'-1} = \tau_j
\end{cases}.
\end{align*}
\]

In the recursion above, \(0 \leq j \leq P-k'-1\). Fig. 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,

\[
f(x; \alpha, \tau) = \sum_{j=0}^{P-k-1} \alpha_j B_{j,k}(x; \tau),
\]

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

\[
L(\alpha, \tau) = \sum_{i=0}^{N-1} (y_i - f(x_i; \alpha, \tau))^2.
\]

**B. Regression and penalized spline with free knot placement**

The regression problem is implicitly regularized by the restriction \(P \ll N\). Explicit regularization can be restored by minimizing

\[
L_\lambda(\alpha, \tau) = L(\alpha, \tau) + \lambda \sum_{j=0}^{P-k-1} \alpha_j^2,
\]

thereby enforcing more smoothness on the solution. When \(L_\lambda(\alpha, \tau)\) is minimized keeping \(\tau\) fixed, the method is called penalized spline \([24]\).

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 \(L_\lambda(\alpha, \tau)\) (c.f., Eq. 1) over both \(\alpha\) and \(\tau\). (For lack of a better term, the method of regression spline with knot optimization and explicit regularization will also be referred to as regression spline in the following.)

Minimization of \(L_\lambda\) over \(\alpha\) and \(\tau\) can be nested as follows.

\[
\min_{\alpha, \tau} L_\lambda(\alpha, \tau) = \min_{\alpha} \left( \min_{\tau} L_\lambda(\alpha, \tau) \right).
\]

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

\[
B_{m,n}(\tau) = B_{m,k}(x_n; \tau),
\]

as

\[
\hat{\alpha}(\tau) = \frac{B^T}{\tau} \left( \frac{GG^T}{\tau} \right)^{-1},
\]

\[
G = BB^T + \lambda I,
\]

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

\[
F_\lambda(\tau) = L_\lambda(\hat{\alpha}(\tau), \tau),
\]

needs to be performed numerically.
Due to the fact that freely moveable knots can coin-
cide, and that this produces discontinuities in B-spline
functions as outlined earlier, curve fitting by regression
spline can accommodate a broader class of functions –
smooth with localized discontinuities – than smoothing
or penalized spline.

The main bottleneck in implementing the full regres-
sion spline method is the global minimization of $F_\lambda(\tau)$
since it is a high-dimensional non-convex function hav-
ing multiple local minima. Trapping by local minima
renders greedy methods ineffective and high-dimension-
ality 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.

C. Model selection

In addition to the parameters $\Theta$ and $\tau$, regression spline
with explicit regularization has two hyper-parameters,
namely the regulator gain $\lambda$ and the number of interior
knots $P - 2(k - 1)$, that affect the outcome of fitting.
Model selection methods can be employed to adaptively
fix these hyper-parameters based on the data.

In this paper, we restrict ourselves to the adap-
tive selection of only the number of knots. This is
done by minimizing the Akaike Information Criterion
\cite{Akaike1974}: For a regression model with $K$ parameters
$\Theta = (\theta_0, \theta_1, \ldots, \theta_{K-1})$,
\[
AIC = 2K - 2\max_{\Theta} \ln \left( \Lambda(\Theta) \right),
\]
where $\Lambda(\Theta)$ is the likelihood function. The specific
expression for AIC used in SHAPES is provided in Sec. IV.

III. PARTICLE SWARM OPTIMIZATION

Under the PSO metaheuristic, the function to be opti-
mized (called the fitness function) is sampled at a fixed
number of locations (called particles). The set of parti-
cles is called a swarm. The particles move in the search
space following stochastic iterative rules called dynami-
cal 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) respect-
ively.

Since its introduction by Kennedy and Eberhart \cite{Kennedy1995},
the PSO metaheuristic has expanded to include a large
diversity of algorithms \cite{Eberhart1996}. In this paper, we consider
the variant called local-best (or lbest) PSO \cite{Clerc2002}. We begin
with the notation \cite{Clerc2002} 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 $\tau$,
  $F$ is $F_\lambda(\tau)$ (c.f., Eq. 14), 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 $i$th particle at the
  $k$th iteration.

- $v_i[k] \in \mathbb{R}^d$: a vector called the velocity of the $i$th
  particle that is used for updating the position of a particle.

- $p_i[k] \in \mathbb{R}^d$: the best location found by the $i$th
  particle over all iterations up to and including the $k$th.
  $p_i[k]$ is called the personal best position of the $i$th
  particle.

\[
F(p_i[k]) = \min_{1 \leq j \leq k} F(x_i[j]).
\]

- $n_i[k]$: 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
  $k$th. $l_i[k]$ is called the local best for the $i$th
  particle.

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

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

\[
F(p_g[k]) = \min_{1 \leq i \leq N_p} F(p_i[k]).
\]

The dynamical equations for lbest PSO are as follows.

\[
v_{i}[k + 1] = w[k]v_{i}[k] + c_1(p_i[k] - x_i[k])r_1 +
\]
\[
c_2(l_i[k] - x_i[k])r_2,
\]
\[
x_{i}[k + 1] = x_i[k] + v_i[k + 1],
\]
\[
v^j_i[k] = \begin{cases} 
  v^j_i[k], & -v_{max}^j \leq v_i^j[k] \leq v_{max}^j \\
  -v_{max}^j, & v_i^j[k] < -v_{max}^j \\
  v_{max}^j, & v_i^j[k] > v_{max}^j
\end{cases}
\]

Here, $w[k]$ is a deterministic function known as the inert-
tia weight, $c_1$ and $c_2$ are constants, and $r_1$ is a diagonal
matrix with iid random variables having a uniform distri-
bution over $[0, 1]$. Limiting the velocity as shown in
Eq. 21 is called velocity clamping.

The iterations are initialized at $k = 1$ by indepen-
dently drawing (i) $x^j_1[1]$ from a uniform distribution over
$[a^j, b^j]$, and (ii) $v^j_1[1]$ from a uniform distribution over
The local best, and position updates given by Eq. 19 and Eq. 20 respectively. Other, more sophisticated, termination conditions are available [30], 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. 19 are the cognitive and social forces respectively. On the 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 \( w_{\text{max}} \) to a final value \( w_{\text{min}} \) in order to transition PSO from an initial exploratory to a final exploitative phase.

For the topology, we use the ring topology with 2 neighbours in which

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

The local best, \( l_i[k] \), in the \( k \)th iteration is updated after evaluating the fitnesses of all the particles. The velocity and position updates given by Eq. 19 and Eq. 20 respectively form the last set of operations in the \( k \)th 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 \( \infty \). 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.

A. PSO tuning

Stochastic global optimizers, including PSO, that terminate in a finite number of iterations do not satisfy the conditions laid out in [33] 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 [33]. Based on widely prevalent values in the literature, these are: \( N_p = 40, c_1 = c_2 = 2.0, \ w_{\text{max}} = 0.9, \ w_{\text{min}} = 0.4, \) and \( \alpha^0_{\text{max}} = 0.5|\beta - \alpha^0| \).

Typically, this leaves the maximum number of iterations, \( N_{\text{iter}} \), as the principal parameter that needs to be tuned. However, for a given \( N_{\text{iter}} \), 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 \( P_{\text{conv}} \) is the probability of successful convergence in any one run and \( N_{\text{runs}} \) is the number of independent runs.

Besides \( N_{\text{iter}} \), therefore, \( N_{\text{runs}} \) is the remaining parameter that should be tuned. If the independent runs can be parallelized, \( N_{\text{runs}} \) 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 \( N_{\text{runs}} \) is needed in an application (say \( N_{\text{runs}} \geq 8 \)), it is usually an indicator that \( P_{\text{conv}} \) should be increased by tuning the other PSO parameters or by exploring a different PSO variant. In this paper, we follow the simpler way of tuning \( N_{\text{runs}} \) by setting it to \( N_{\text{runs}} = 4 \), the typical number of processing cores available in a high-end desktop.

IV. SHAPES ALGORITHM

The SHAPES algorithm is summarized in the pseudocode given in Fig. 2. The user specified parameters of the algorithm are (i) the number, \( N_{\text{runs}} \), of PSO to use per data realization; (ii) the number of iterations, \( N_{\text{iter}} \), to termination of PSO; (iii) the set of models, \( N_{\text{knots}} \), over which AIC based model selection (see below) is used; (iv) the regulator gain \( \lambda \).

A model in SHAPES is specified by the number of non-repeating knots. For each model \( M \in N_{\text{knots}} \), \( F(M, r_M) \) 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

\[
\text{AIC} = 4M + F(M, r_M),
\]

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 \( \mathbf{y} \) to produce (i) the best fit model \( M_{\text{best}} \in N_{\text{knots}} \); (ii) the fitness value associated with the best fit model; (iii) the estimated function \( \hat{f} \) from the best fit model. The generation of \( \hat{f} \) includes a bias correction step described next.

A. 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, \( \hat{f} \), has a systematic point-wise bias towards zero. A bias correction transformation is applied to \( \hat{f} \) as follows.
The final estimate is given by
\[
\hat{f}(M, r) = F_M(\hat{\tau}(r)) \quad \text{Best fitness value}
\]

\[
\text{Loop over models for } M \in \mathbb{N}_{\text{knots}} \quad \text{Do}
\]
\[
\text{Best PSO run} \quad \text{AIC}(M) = \text{AIC} for F(M, r_M) \quad \text{c.f., Eq. 24}
\]
\[
\hat{M}_{\text{best}} \leftarrow \arg \min_M \text{AIC}(M) \quad \text{Model with lowest AIC}
\]
\[
\hat{f} \leftarrow \hat{f}(\hat{M}_{\text{best}}) \quad \text{Bias corrected } \hat{f} \quad \text{c.f., Sec. IV.A}
\]

\[
\text{Fitness of best model } \text{AIC}(\hat{M}_{\text{best}}) \quad \text{Estimated bias-corrected } \hat{f} \quad \text{Estimated function from best model}
\]

\[
\text{Fitness of best model } \hat{f}(\hat{M}_{\text{best}}, r_{\text{best}}) \quad \text{Fitness of best model}
\]

FIG. 2: Pseudo-code for the SHAPES algorithm. All quantities with parentheses and integer arguments stand for arrays, with the argument as the array index.

First, the unit norm estimated function \( \hat{u} \) is obtained,
\[
\hat{u} = \frac{\hat{f}}{\| \hat{f} \|},
\]
where \( \| \hat{f} \| = \left( \sum \hat{f}^2 \right)^{1/2} \) is the L₂ norm.

Next, a scaling factor \( A \) is estimated as
\[
A = \arg \min_a \left( \sum_{i=0}^{N-1} (y_i - a\hat{u})^2 \right).
\]

The final estimate is given by \( \hat{f} = A\hat{u} \).

B. Knot merging and dispersion

In both of the mappings described in Sec. IV.C it is possible to get knot sequences in which a subset \( (\tau_i, \tau_{i+1}, \ldots, \tau_{i+m-1}) \) of \( 1 < m < M-2 \) 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 \( \tau_{i+m-1} \) until its multiplicity saturates at \( k \). The remaining knots, \( \tau_i \) 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 \( M \in \mathbb{N}_{\text{knots}} \).

C. 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 \( \overline{\tau} \) to \( \overline{z} \).

- Plain: \( \overline{\tau} \) is sorted in ascending order. After sorting, \( k-1 \) copies of \( z_0 \) and \( z_{M-1} \) are prepended and appended respectively to \( \overline{\tau} \). These are the repeated end knots as described in Sec. I.A

- Centered-monotonic: In this scheme \( \overline{\tau} \), the search space is the unit hypercube: \( z_i \in [0, 1], \forall i \). First, an initial set of \( M \) knots is obtained from
\[
z_0 = \tau_0, \quad 2 \leq i \leq M-2 = \frac{\tau_i - \tau_{i-1}}{\tau_{i+1} - \tau_{i-1}}, \quad z_{M-1} = \frac{\tau_{M-1} - \tau_0}{1 - \tau_0}.
\]

This is followed by prepending and appending \( k-1 \) copies of \( \tau_0 \) and \( \tau_{M-1} \) respectively to the initial knot sequence.

In the plain map, any permutation of \( \overline{\tau} \) maps into the same knot sequence due to sorting. This creates degeneracy in \( F\lambda \), 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 $\tau$ to a given $z$. Moreover, $\tau$ 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 $z_i = 0.5$, $1 \leq i \leq M - 2$, 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 [5] back in 1978. A logarithmic map for knots was proposed in [5] that, while not implemented in this paper, should be examined in future work.

D. 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.)

E. 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.

V. 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 Sec. VII.

A. Benchmark functions

The benchmark functions used in this study are listed in Table I 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.

| Expression | Domain |
|------------|--------|
| $f_1(x) = 90(1 + e^{-100(x-0.4)})^{-1}$ | $x \in [0, 1]$ |
| $f_2(x) = \begin{cases} 0.01 + (x - 0.3)^2 & \text{if } 0 \leq x < 0.6 \\ 0.015 + (x - 0.65)^2 & \text{if } 0.6 \leq x \leq 1 \end{cases}$ | |
| $f_3(x) = 100e^{-100x} - 3 + (10x - 5)^2/500$ | $x \in [0, 1]$ |
| $f_4(x) = \sin(x) + 2e^{-100x}$ | $x \in [-2, 2]$ |
| $f_5(x) = \sin(2x) + 2e^{-16x^2} + 2$ | $x \in [-2, 2]$ |
| $f_6(x) = \begin{cases} 4x^2(3 - 4x) & \text{if } 0 \leq x < 0.5 \\ \frac{1}{2}x(4x^2 - 10x + 7) - \frac{1}{2} & \text{if } 0.5 \leq x < 0.75 \\ 0.75 \leq x \leq 1 \end{cases}$ | |
| $f_7(x) = B_{3,4}(x; \tau)$; $\tau = (\tau_0, \tau_1, \ldots, \tau_{11})$ | $x \in [0, 1]$ |
| $\tau_i = \begin{cases} 0.3 & \text{if } 0 \leq i \leq 2 \\ 0.55 & \text{if } 8 \leq i \leq 10 \\ 0.3, 0.4, 0.5, 0.55 & \text{if } 2 \leq i \leq 8 \end{cases}$ | |
| $f_8(x) = 0.025 + B_{3,4}(x - 0.125; \tau)$ | $x \in [0, 1]$ |
| $f_9(x) = \begin{cases} 10 + 15|x - 0.55| & \text{if } |x - 0.55| \leq 0.25 \\ 0 & \text{else} \end{cases}$ | $x \in [0, 1]$ |
| $f_{10}(x) = e^{-\frac{16}{0.425}(x - 0.5)} \sin(10.24\pi(x - 0.5))$ | $x \in [0, 1]$ |

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 $f_{10}$. 

TABLE I: The benchmark functions used in this paper. A ‘*’ in the reference column stands for a function that is introduced in this paper. The sources from which the functions have been obtained are: $f_1$ to $f_3$ [15]; $f_4$ [13]; $f_5$ [22, 67]; $f_6$ [39]; $f_7$ [24]. Functions $f_8$ to $f_{10}$ are introduced here.
B. 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 \([0, 1]\).

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

\[
SNR = \frac{\|f\|}{\sigma},
\]

where \( f \) is a benchmark function and \( \sigma \) 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.

C. Metrics

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

\[
RMSE = \left[ \frac{1}{N_R} \sum_{j=1}^{N_R} \|f - \hat{f}_j\|^2 \right]^{1/2},
\]

where \( f \) is the true function in the data and \( \hat{f}_j \) its estimate from the \( j \)th data realization. We use bootstrap with \( 10^4 \) independently drawn samples with replacement from the set \( \{\|f - \hat{f}_j\|^2\} \) 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 \( M_{\text{best}} \in N_{\text{knots}} \) over the \( N_R \) data realizations, where \( M_{\text{best}} \) and \( N_{\text{knots}} \) were defined in Sec. IV.

D. Labeling scheme

Several design choices in SHAPES were described in Sec. IV. A useful bookkeeping device for keeping track of the many possible combinations of these choices is the labeling scheme presented in Table II.

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.

| PSO algorithm (Sec. III) | L: lbest PSO | P: | C: |
|-------------------------|-------------|---|---|
| Knot Map (Sec. IV C)    |             |   |   |
| SNR (Eq. 29)           | P: Plain    |   |   |
| \( \lambda \) (Eq. 9)  | C: Centered-monotonic |
| \( N_{\text{iter}} \) (Number of PSO iterations) |   |   |
| End knots (Sec. IV D)  | F: Fixed    | V: Variable |
| End B-splines (Sec. IV E) | K: Keep | D: Drop |
| Knot merging (Sec. IV F) | M: Merge | H: Heal |

TABLE II: Labeling scheme for a combination of design choices in SHAPES. 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 \( Y_1Y_2X_3X_4X_5X_6Y_7Y_8 \) 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.

VI. COMPUTATIONAL CONSIDERATIONS

The results in this paper were obtained with a code implemented entirely in MATLAB [39]. Some salient points about the code are described below.

The evaluation of B-splines uses the efficient algorithm given in [1]. 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.) Nevertheless, 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, \( N_{\text{knots}} \), 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 pseudorandom 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.

VII. RESULTS

The presentation of results is organized as follows. Sec. VII A shows single data realizations and estimates for a subset of the benchmark functions. Sec. VII B analyzes the impact of the regulator gain $\lambda$ on estimation. Sec. VII C and Sec. VII D contain results for SNR $= 100$ and SNR $= 10$ respectively.

In all cases, the set of models used was $N_{knots} = \{5, 6, 7, 8, 9, 10, 12, 14, 16, 18\}$. 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.

Fig. 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 $N_{iter} = 100$ to $N_{iter} = 200$ 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 $N_{iter} = 50$ for the SNR $= 10$ case.

A. 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.

For ease of comparison, we have picked only the benchmark functions ($f_1$ to $f_6$) used in [17]. The SNR of each function matches the value one would obtain using the noise standard deviation tabulated in [17]. 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., Sec. IV C), (c) keeping the end 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.

B. 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 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.

Fig. 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 $\lambda$. 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.

Fig. 7 and Fig. 8 present a statistically more rigorous study of the effect of $\lambda$ 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 \( \lambda \), 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 Sec. VII D.) The improvement becomes more pronounced as SNR is lowered. (There is no statistically significant effect of \( \lambda \) on the number of knots in the best fit model at either SNR.)

The higher values of the regulator gains in Fig. 7 and Fig. 8 – \( \lambda = 0.1 \) and \( \lambda = 5.0 \) 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.

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; 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 \( LP_{SNR,0.1,100,FKM} \) (c.f., Table II).

FIG. 6: Effect of regulator gain, \( \lambda \), on estimation. The solid and dashed curves are the estimates obtained with \( \lambda = 0.1 \) and \( \lambda = 5.0 \) respectively, where \( \lambda \) 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 \( \lambda \), the algorithm settings – \( LP_{100,0.1,100,FKM} \) (see Table II) – were identical for the two estimated curves.

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 \( \lambda = 5.0 \) respectively. The other algorithm settings used for this plot can be read off from the key strings shown in the legend using Table II. The data points correspond to the benchmark functions shown on the abscissa. The error bars show \( \pm 1\sigma \) deviations, where \( \sigma \) is the estimated standard deviation.
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 $\text{SNR} = 10$ 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 II. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where $\sigma$ is the estimated standard deviation.

C. Results for $\text{SNR} = 100$

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.

For the $\text{SNR} = 100$ 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 Fig. 10 and Fig. 11 where the point-wise sample mean and $\pm 2\sigma$ deviation, $\sigma$ 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 Sec. VII A.

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 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.

D. Results for $\text{SNR} = 10$

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

Unlike the $\text{SNR} = 100$ 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. In fact, the results

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 $\text{SNR} = 100$ 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 II. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where $\sigma$ is the estimated standard deviation.
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 \( \sigma \) 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 II.

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 \( \sigma \) 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 II.

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

VIII. 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 best 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...
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 $\sigma$ 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 III.

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 cases examined was high enough to completely mask the location of discontinuities and they were not well localized. Nonetheless, even with a conservative error envelope of $\pm 2\sigma$ around the mean estimated signal, the overall shape of the true function is visually clear in all the examples. This shows that the estimated functions are responding at a statistically significant level to the presence of the true function in the data. 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.

Further characterization of the performance of SHAPES for the low SNR case requires incorporating a hypothesis test, a topic that we are currently investigating. For the case of transient signals represented by $f_7$ to $f_{10}$, the fitness value returned by SHAPES may itself serve as a powerful detection statistic since the estimated functions depart quite significantly from the expected behavior under the null hypothesis. (Hypotheses testing for $f_7$ was explored in this manner in [23] with a method that did not include bias reduction.) However, this expectation may need to be modified for the other benchmark functions, in particular for $f_1$. Another issue that comes up in a hypothesis test is that of bias correction: it must be used only if the detection threshold of the test is crossed, otherwise estimates under both the null and alternative hypotheses will get amplified. This creates an interesting interplay between hypothesis testing and estimation that needs a careful scrutiny.

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 settings were found to depend on only some coarse features of a function, such as its behavior at data
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 $\sigma$ 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 III.

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 $\lambda$. Ideally, however, $\lambda$ 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 $\lambda$ 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.

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 III. The data points correspond to the benchmark functions shown on the abscissa. The error bars show $\pm 1\sigma$ deviations, where $\sigma$ is the estimated standard deviation.

FIG. 16: Comparison of plain and centered-monotonic maps under the FKM and VDH algorithm settings respectively at SNR = 10. See Table III 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 $\sigma$ is the estimated standard deviation.

IX. 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.

[1] C. de Boor, A Practical Guide to Splines (Applied Mathematical Sciences), Springer, 2001.
[2] E. J. Wegman, I. W. Wright, Splines in statistics, Journal of the American Statistical Association 78 (382) (1983) 351–365.
[3] W. Hardle, Applied nonparametric regression, Econometric Society Monographs 19, Cambridge University Press.
[4] S. Wold, Spline functions in data analysis, Technometrics 16 (1) (1974) 1–11.
[5] D. L. Jupp, Approximation to data by splines with free knots, SIAM Journal on Numerical Analysis 15 (2) (1978) 328–343.
[6] Z. Luo, G. Wahba, Hybrid adaptive splines, Journal of the American Statistical Association 92 (437) (1997) 107–116.
[7] P. L. Smith, Curve fitting and modeling with splines using statistical variable selection techniques, Tech. rep., NASA, Langley Research Center, Hampton, VA, report NASA-166034 (1982).
[8] J. H. Friedman, B. W. Silverman, Flexible parsimonious smoothing and additive modeling, Technometrics 31 (1) (1989) 3–21.
[9] J. H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1) (1991) 1–67.
[10] G. H. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (2) (1979) 215–223.
[11] M. Mitchell, An Introduction to Genetic Algorithms by Melanie Mitchell, Bradford, 1998.
[12] P. J. Green, Reversible jump markov chain monte carlo computation and bayesian model determination, Biometrika 82 (4) (1995) 711–732.
[13] J. Pittman, Adaptive splines and genetic algorithms, Journal of Computational and Graphical Statistics 11 (3) (2002) 615–638.
[14] I. DiMatteo, C. R. Genovese, R. E. Kass, Bayesian curve-fitting with free-knot splines, Biometrika 88 (4) (2001) 1055–1071.
[15] F. Yoshimoto, T. Harada, Y. Yoshimoto, Data fitting with a spline using a real-coded genetic algorithm, Computer-Aided Design 35 (8) (2003) 751–760.
[16] S. Miyata, X. Shen, Adaptive free-knot splines, Journal of Computational and Graphical Statistics 12 (1) (2003) 197–213.
[17] A. Gálvez, A. Iglesias, Efficient particle swarm optimization approach for data fitting with free knot b-splines, Computer-Aided Design 43 (12) (2011) 1683–1692.
[18] S. D. Mohanty, Particle swarm optimization and regression analysis I, Astronomical Review 7 (2) (2012) 29–35.
[19] J. Kennedy, R. C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks: Perth, WA, Australia, Vol. 4, IEEE, 1995, p. 1942.
[20] L. S. Collaboration, V. Collaboration, et al., Gwrtc-1: a gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs, arXiv preprint arXiv:1811.12007.
[21] C. W. Helstrom, Statistical Theory of Signal Detection, Pergamon, London, 1968.
[22] S. D. Mohanty, Spline based search method for unmodeled transient gravitational wave chirps, Physical Review D 96 (10) (2017) 102008.
[23] S. D. Mohanty, Swarm Intelligence Methods for Statistical Regression, Chapman and Hall/CRC, 2018.
[24] D. Ruppert, M. P. Wand, R. J. Carroll, Semiparametric regression, Vol. 12, Cambridge University Press, 2003.
[25] C. H. Reinsch, Smoothing by spline functions, Numerische mathematik 10 (3) (1967) 177–183.
[26] G. Claeskens, T. Krivobokova, J. D. Opsomer, Asymptotic properties of penalized spline estimators, Biometrika 96 (3) (2009) 529–544.
[27] H. B. Curry, I. J. Schoenberg, On spline distributions and their limits—the polya distribution functions, Bulletin of the American Mathematical Society 53 (11) (1947) 1114–1114.
[28] C. de Boor, On calculating with b-splines, Journal of Approximation Theory 6 (1) (1972) 50 – 62.
[29] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: Selected Papers of Hirotugu Akaike, Springer, 1998, pp. 199–213.
[30] A. P. Engelbrecht, Fundamentals of computational swarm intelligence, Vol. 1, Wiley Chichester, 2005.
[31] J. Kennedy, 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, 1999, pp. 1931–1938.
[32] M. E. Norman, S. D. Mohanty, T. S. Weerathunga, Particle swarm optimization based search for gravitational waves from compact binary coalescences: Performance improvements, Physical Review D 96 (4) (2018) 044029.
[33] F. J. Solis, R. J.-B. Wets, Minimization by random search techniques, Mathematics of Operations Research 6 (1) (1981) 19–30.
[34] D. Bruton, J. Kennedy, Defining a standard for particle swarm optimization, in: Swarm Intelligence Symposium, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Vol. 3, IEEE, 1999, pp. 1931–1938.
[35] C. H. Reinsch, Smoothing by spline functions, Numerische mathematik 10 (3) (1967) 177–183.
[36] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, IEEE, 2007, pp. 120–127.
[37] C. Leung, Estimation of unmodeled gravitational wave transients with spline regression and particle swarm optimization, SIAM Undergraduate Research Online (SIURO) 8.
[38] D. Denison, B. Mallick, A. Smith, Automatic bayesian curve fitting, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 60 (2) (1998) 333–350.
[39] M. Li, Y. Yan, Bayesian adaptive penalized splines, Journal of Academy of Business and Economics 2 (2006) 129–141.
[40] T. Lee, On algorithms for ordinary least squares regression spline fitting: a comparative study, Journal of statistical computation and simulation 72 (8) (2002) 647–663.
[39] Matlab Release2018b, The MathWorks, Inc., Natick, Massachusetts, United States.; http://www.mathworks.com/.