Enrique Aguilar
School of Engineering and Science,
Tecnológico de Monterrey, Tlalpan,
Mexico City 14380, Mexico
e-mail: enriqueam1409@gmail.com

Hugo Elizalde[1]
School of Engineering and Science,
Tecnológico de Monterrey, Tlalpan,
Mexico City 14380, Mexico
e-mail: hugo.elizalde@itesm.mx

Diego Cárdenas
School of Engineering and Science,
Tecnológico de Monterrey, Tlalpan,
Mexico City 14380, Mexico
e-mail: diego.cardenas@itesm.mx

Oliver Probst
School of Engineering and Science,
Tecnológico de Monterrey,
Monterrey 64849, Nuevo Leon, Mexico
e-mail: oprobst@itesm.mx

Pier Marzocca
Aerospace, Mech. & Manuf. Eng. Department,
RMIT University,
P.O. Box 71,
Bundoora 3083, Victoria, Australia
e-mail: pier.marzocca@rmit.edu.au

Ricardo A. Ramirez-Mendoza
School of Engineering and Science,
Tecnológico de Monterrey,
Monterrey 64849, Nuevo Leon, Mexico
e-mail: ricardo.ramirez@itesm.mx

An Adaptive Curvature-Guided Approach for the Knot-Placement Problem in Fitted Splines

This paper presents an adaptive and computationally efficient curvature-guided algorithm for localizing optimum knot locations in fitted splines based on the local minimization of an objective error function. Curvature information is used to narrow the searching area down to a data subset where the local error function becomes one-dimensional, convex, and bounded, thus guaranteeing a fast numerical solution. Unlike standard curvature-guided methods, typically relying on heuristic rules, the novel method here presented is based on a phenomenological approach as the error function to be minimized represents geometrical properties of the curve to be fitted, consequently reducing case-sensitivity issues and the possibility of defining spurious knots. A knot-readjustment procedure performed in the vicinity of a newly created knot has the ability of dispersing knots from otherwise highly knot-populated regions, a feature known to generate undesired local oscillations. The performance of the introduced method is tested against three other methods described in the literature, each handling the knot-placement problem via a different paradigm. The quality of the fitted splines for several datasets is compared in terms of the overall accuracy, the number of knots, and the computing efficiency. It is demonstrated that the novel algorithm leads to a significantly smaller knot vector and a much lower computing time, while preserving or improving the overall accuracy. [DOI: 10.1115/1.4040981]

Keywords: knot-placement, spline fitting, curvature, noise, error minimization

1 Introduction

Current applications in computer-aided design make an extensive use of spline-based geometry, where the spline's accuracy (i.e., its error against its associated dataset) is typically the main criterion assessing its quality [1–3]. In more specialized applications, computational efficiency becomes equally relevant, where even small savings can have a significant impact in terms of the computational burden [4–6]. For example, the computing time involved in generating a spline has a direct influence in analyses requiring multiple (i.e., hundreds or thousands) updates of spline-based geometry during a nonlinear analysis or topological optimization [7–8]. These new trends demand more efficient approaches for generating optimized splines.

In spite of formidable advances in the field, the optimum fitting of splines is still considered a complex endeavor, particularly for datasets with high curvature, noise, and/or high density, where a key input for achieving a high-quality fit is a suitable knot-vector providing the location and parameterization of each knot along the arc-length suggested by the dataset [1–3,7–17]. In principle, the knot-placement problem can be stated as a multi-objective, multivariable, nonconvex, and multimodal constrained optimization problem, where the sought variables are the number and location of the spline's knots as well as its control points [13,14]. For large general datasets, this approach is difficult to solve, requires intensive computational resources, and cannot guarantee convergence to a feasible result in view of the many mathematical and geometrical constraints involved. The problem can be greatly simplified by prespecifying the number and location of the knots, leading to a linear system of equations solved for the control points, where the overall error of the generated spline with respect to the dataset is sought to be minimized by iteratively relocating and/or re-parameterizing the knots [10–12]. This recursive procedure, although more robust than the former, can still be computationally expensive for datasets with high density; in addition, a globally optimum solution cannot be guaranteed [1–3,15].

For datasets featuring high density and/or high geometrical complexity, where relatively many knots are required to capture the underlying features suggested by the dataset, alternative optimization methods are preferred. An example of the latter is genetic algorithm, a particular type of metaheuristic methods, where an initial group of potential solutions with randomly generated genes is iteratively combined and mutated, selecting those with the best phenotype to survive the next generation [1–3,14–18]. These class of methods have a number of significant drawbacks, including the specification of parameters not intrinsic to the problem of curve reconstruction (i.e., a certain initial size of the population), the lack of mathematical proof of convergence, and sometimes an erratic iterative behavior. Yet another paradigm for solving the knot-placement problem is based on the curvature of the dataset. This class of empirical methods assume that the

[1]Corresponding author.

Contributed by the Computers and Information Division of ASME for publication in the Journal of Computing and Information Science in Engineering. Manuscript received November 27, 2015; final manuscript received July 18, 2018; published online September 5, 2018. Editor: Satyandra K. Gupta.

optimal locations of knots, i.e., those minimizing the error of the fitted spline against the dataset, are found in the vicinity of high curvature regions [9–12,19]. Although computationally more efficient than metaheuristic methods, the approach may suffer from some degree of case-sensitivity and can therefore not ensure a truly optimal solution for general datasets. The problem of curvature extraction from raw data, particularly for the case of dense and noisy data sets, is a main topic in its own right and many methods have been proposed for tackling a wide range of applications [20–30].

The knot-placement method proposed in this paper uses curvature information for defining a few initial knots, as well as for narrowing the searching region for inserting additional knots. The location of each new knot is governed by the minimization of an objective error function measuring the local deviation of a data subset (i.e., a small portion of the whole dataset) to an increasingly refined piecewise linear approximation to the sought spline. This data subset gets smaller with each new knot defined, thus an increasingly refined solution can be obtained at an increasingly smaller penalty. By working at a local level (i.e., the data subset), the error function to be minimized becomes one-dimensional, convex, and bounded, hence a fast convergence to a locally optimum knot-location can be guaranteed, where the term "optimum" is here used in the context of a mathematical optimization procedure. It is highlighted that this method does not require multiple evaluations of the spline, while new knots are defined because the error measure is evaluated locally. Once all knots are defined, the method proceeds to generate the spline once and for all. The new method includes a "knot-readjustment" scheme, an innovative feature absent in the methods reviewed by these authors, bringing the ability of readjusting the location of existing knots in the vicinity of a newly defined knot while maintaining the "locally iterative" feature (i.e., without resorting to multiple spline's iterations). This knot-readjustment reduces the development of highly knot-populated regions due to high-curvature or geometrical complexity, a primary cause of spline's local oscillations in fitted splines. Finally, it is stressed that the novelty and scope of the introduced method is circumscribed to the process of knot-placement, while all other steps necessary to complete the spline generation (i.e., filtering, curvature extraction, and parameterization) are here performed by standard techniques [1–3,9–12,29–33].

The rest of the paper is organized as follows: Sec. 2 briefly explains the mathematical nomenclature for defining splines in the context of curve fitting, while Sec. 3 details the knot-placement algorithm here proposed. In Sec. 4, the introduced method is tested against other three methods borrowed from the literature [1,9,12], each based on a different paradigm for handling the knot-placement problem, for several sample datasets. A comparative analysis of the generated splines is carried out in terms of their accuracy, number of knots, and computing efficiency involved. Finally, Sec. 5 offers some concluding remarks.

2 Brief Background on Splines

A kth-degree spline $B(u)$ can be written as [32]

$$B(u) = \sum_{i=0}^{n} N_{i,k}(u) Q_i \tag{1}$$

where $u \in [u_0, u_m]$ is a parameterized position along the curve's path representing its arc-length as a function of the total path-length, $u_0$andu_m represent the curve's first and last such positions, respectively, $N_{i,k}(u)$ are $n+1$ basis functions ($i = 0 \ldots n$) having a common degree k, and Q_i are a set of $n+1$ control points. The knot-vector $T = [u_0, u_1, \ldots, u_m]$ gathers $m+1$ knots each representing a junction between the individual polynomial segments conforming the spline. The number of knots, the number of control points, and the degree of the spline cannot be set independently but are tied by the equality $n + 1 = m - k$. In a curve fitting context [31], where an ordered set of points $P_j (j = 0 \ldots p$, typically $p \gg m + 1$ for dense data) is to be fitted by a kth-degree spline, each data point is first parameterized according to its expected arc-length position along the spline's path, then knots are assigned to a few feature points according to a certain criterion (the main topic of this paper). Afterward, the $n+1$ basis functions $N_{i,k}$ are built recursively [32,34] for a predefined degree n, and the $n+1$ control points Q_i are found by minimizing an error function (typically a square error) between the points P_j and the corresponding points generated by the spline

$$\frac{\partial}{\partial Q_i}\left[\sum_{j=0}^{p} \left(B(u) - P_j\right)^2\right] = 0, \qquad \text{for } i = 0 \ldots n \tag{2}$$

The goodness of fit can be measured via the root-mean-square error E_{RMS} as defined by

$$E_{\text{RMS}} = \sqrt{\frac{1}{p+1}\sum_{j=0}^{p} \left(B(u) - P_j\right)^2} \tag{3}$$

3 The Knot-Placement Algorithm

The knot-placement algorithm introduced in this work is next exemplified with reference to the generic illustrations shown in Figs. 1(*a*)–1(*d*) and the flowchart of Fig. 2, where it is assumed that a filtered curvature function is already available. The algorithm starts by defining the initial knots (here called "parent knots") at feature datapoints corresponding to the end-points and local maxima of the curvature function, labeled in Fig. 1(*a*) as K_0 to K_6. A polygon traced along these knots in the dataset (see Fig. 1(*b*)) can be viewed as a piecewise linear approximation to the sought spline and provides a geometrical interpretation to the knot-placement scheme. In this polygon, segment $t-1$ is bounded by knots K_{t-1} and K_t, while segment t is bounded by knots K_t and K_{t+1}. The datapoints contained within the region encompassed by any given segment t are referred to as data subset t, representing a narrowed searching region where the algorithm will later perform a mathematical optimization provided some condition is met, as explained next.

The integral of the absolute value of the curvature ($|\kappa|$) along a given data subset t bounded by knots K_t and K_{t+1} (see Fig. 1(*b*)) represents the total angle θ_t swept by the tangent vector, and it is a measure of how sharply a curve bends

$$\theta_t = \int_{K_t}^{K_{t+1}} |\kappa| \tag{4}$$

If θ_t surpasses some given threshold $\theta_{\max}$ (to be defined later), it is an indication that a new knot needs to be inserted at some suitable location within the data subset t in order to provide an improved description. Given this scenario, data subset t (and its associated segment t) will be partitioned by inserting a new knot $K_{t+\gamma}$ (where $0 < \gamma < 1$), that is, γ is a unity-normalized length between knots K_t and K_{t+1}, thus refining the original segment $\overline{K_t K_{t+1}}$ into two subsegments $\overline{K_t K_{t+\gamma}}$ and $\overline{K_{t+\gamma} K_{t+1}}$ (see Fig. 1(*c*)).

The optimum location is chosen as to minimize an error function representing the accumulated deviation of both subsegments with respect to the datapoints contained in the data subset t, as explained next. As illustrated in Fig. 1(*d*), the orthogonal deviation e_i of any point F_i contained within the first subsegment $\overline{K_t K_{t+\gamma}}$ with respect to the chord described by its end-knots can be measured as

$$e_i = \left|P_i - P_i'\right| = (P_i - K_t)\frac{N}{|N|} \tag{5}$$

where P_i' is the normal projection of P_i onto the chord $\overline{K_t K_{t+\gamma}}$ and N is a vector perpendicular to it, defined by a 90 deg counter-clockwise rotation of the chord

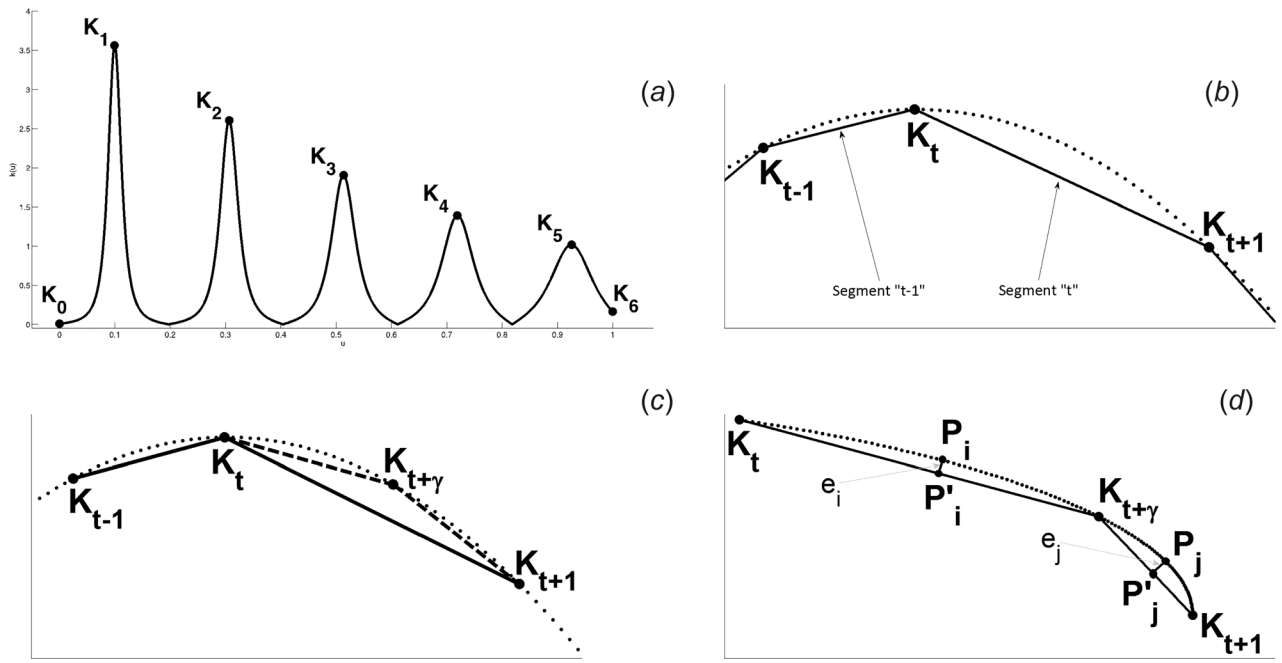

Fig. 1 Generic illustrations for describing the knot-placement algorithm. The solid lines represent the original polygon segments, the dashed lines represent the new subsegments, and the dotted lines represent the dataset. (*a*) Representative curvature function and "parent knots"; (*b*) piecewise linear polygon (solid line), segments (and associated data subsets) $t-1$ and t; (*c*) a new knot $K_{t+\gamma}$ is inserted at an optimum location within segment t; and (*d*) orthogonal deviations e_i and e_j.

$$N = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (P_i - K_t) \tag{6}$$

Therefore, e_i can be expressed in terms of the known x, y coordinates of F_i $(P_{i,x}, P_{i,y})$ and K_t $(P_{t,x}, P_{t,y})$, and of the unknown y-coordinate of $K_{t+\gamma}$ $(K_{t+\gamma,y})$

$$e_i(P_i, K_{t+\gamma}) = \frac{(P_{i,y} - K_{t,y})(K_{t+\gamma,x} - K_{t,x}) + (K_{t,x} - P_{i,x})(K_{t+\gamma,y} - K_{t,y})}{\sqrt{(K_{t+\gamma,x} - K_{t,x})^2 + (K_{t+\gamma,y} - K_{t,y})^2}} \tag{7}$$

where $K_{t+\gamma,x}$ is the x-coordinate of knot $K_{t+\gamma}$, a known quantity for any given $K_{t+\gamma,y}$. Similarly, the orthogonal deviation e_j of any point P_j contained within the second subsegment $\overline{K_{t+\gamma}K_{t+1}}$ with respect to the chord described by its end-knots can be expressed in terms of the known x, y coordinates of P_j and K_{t+1}, and of the unknown y-coordinate of $K_{t+\gamma}$

$$e_j(P_j, K_{t+\gamma}) = \frac{(P_{j,y} - K_{t+\gamma,y})(K_{t+1,x} - K_{t+\gamma,y}) + (K_{t+\gamma,x} - P_{j,x})(K_{t+1,y} - K_{t+\gamma,y})}{\sqrt{(K_{t+1,x} - K_{t+\gamma,x})^2 + (K_{t+1,y} - K_{t+\gamma,y})^2}} \tag{8}$$

Therefore, the error function to be minimized can be built as

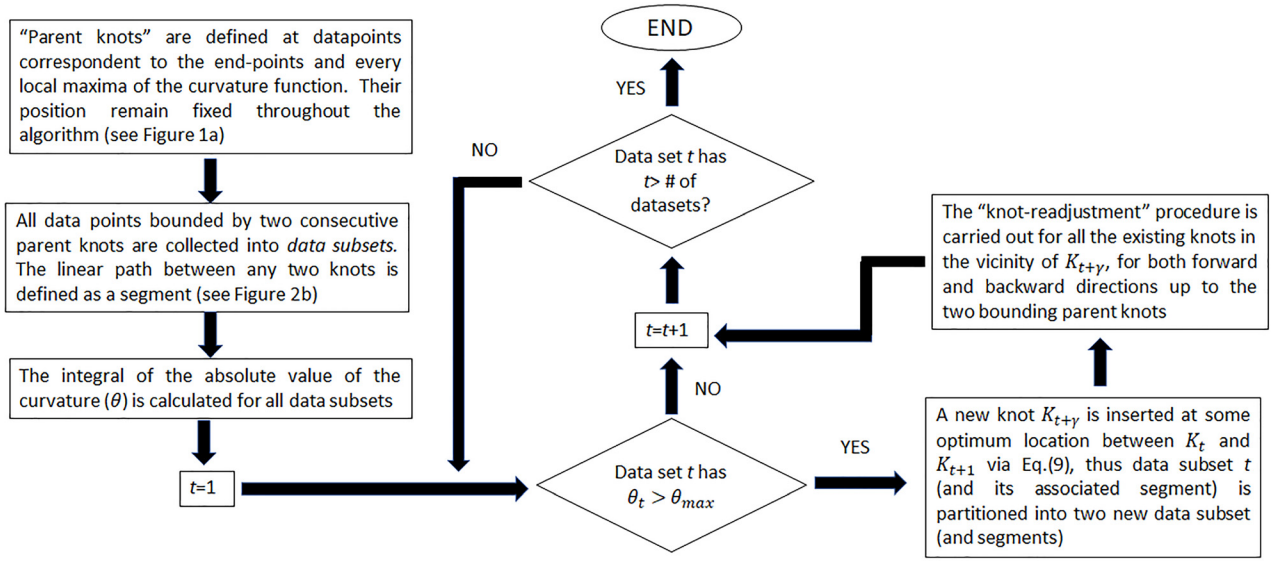

Fig. 2 Flowchart of the main algorithm

$$E(K_{t+\gamma}) = \sum_{i=t}^{t+\gamma} e_i^2 + \sum_{j=t+\gamma}^{t+1} e_j^2 \tag{9}$$

corresponding to a locally convex function bounded by K_t and K_{t+1} and with a single unknown $K_{t+\gamma,y}$ for which a fast numerical solution can be guaranteed via any closed method (i.e., bisection). With the new knot $K_{t+\gamma}$ defined, segment t (and its associated data subset) is partitioned to reflect this new vertex (see Fig. 1(*d*)), thus representing a closer approximation to the sought spline within the analyzed data subset. The above description completes the main algorithm introduced in this work, which should be performed at all data subsets for which $\theta_t \geq \theta_{\max}$. Keep in mind that, every time a data subset is partitioned by inserting a new knot, two new data subsets are created and each needs to be checked whether or not it meets the above criterion in which case the algorithm is applied for inserting a new knot.

With respect to the threshold $\theta_{\max}$ defined earlier, it was demonstrated in Ref. [35] that a cubic spline experiences locally small deflections for $\theta < \pi/6$, therefore the criterion $\theta_{\max} = \pi/6$ was here adopted for all the numerical experiments carried out, where exclusively cubic splines were generated. In more specialized applications, this threshold can be specified by the user to allow some control on the refinement sought in the knot-vector, where lower values will result in more knots to be defined (although the latter number needs not being specified a priori).

3.1 Knot Re-Adjustment. The main algorithm described earlier can be slightly modified to include an innovative feature absent in all methods so far reviewed by these authors, consisting in updating the location of all the existing knots in the vicinity of a newly inserted knot, while still working at the local (i.e., data subset) level. In the context of this work, "the vicinity of a newly inserted knot" refers to that portion of the dataset encompassed by two parent knots (which are fixed by design) where a new knot has just been inserted. The creation of any new knot, particularly in already highly knot-populated regions, will bring out new relationships between existing knots around its vicinity, which current locations could no longer be optimum. This effect can propagate, although with decreasing strength, in both forward and backward directions of the dataset (that is, for arc-length positions above and below the newly inserted knot, respectively), thus all the existing knots between the newly inserted knot and the two bounding parent-knots (which represent fixed boundaries thus their location cannot be updated) should be readjusted to new optimum locations. As confirmed in the numerical experiments reported in Sec. 4, the most obvious effect of this knot-readjustment technique is that the existing knots in the vicinity of a newly inserted knot are "pushed" away from it, thus dissolving to some extent the occurrence of highly clustered knot regions, a well-known pathology which generates high-frequency local oscillations in splines.

The idea outlined earlier can be easily implemented by an almost identical procedure to that already described in Sec. 3 and Fig. 1 for defining knot $K_{t+\gamma}$, that is, by minimizing Eq. (9) for a given knot located between two bounding knots. To this end, the same figures exemplifying the insertion of knot $K_{t+\gamma}$ between its bounding knots K_t and K_{t+1} (see Figs. 1(*c*) and 1(*d*)) are still valid for exemplifying the relocation of any given knot bounded by any given two knots, except that now the current location of the knot to be readjusted can provide a very close initial guess for finding its new location, thus making an already inexpensive calculation even cheaper. This can be summarized in the following two-stage (forward and backward) pseudocode:

(1) Forward readjustment: proceeding from the knot K_{t+1} (i.e., located one after the newly inserted knot $K_{t+\gamma}$) and up to the knot located just before the next forward parent knot, as follows: for $i = 1$ up to $i =$ (index of the forward parent knot $- 1$): knot K_{t+i} (bounded by the knots $K_{t+(i-1)+\gamma}$ and K_{t+i-1}) is readjusted to an updated position $K_{t+i+\gamma}$ via Eq. (9); end.
(2) Backward readjustment: proceeding from the knot K_t (i.e., located one before the newly inserted knot $K_{t+\gamma}$) and down to the knot located just before the next backward parent knot: for $i = 0$ down to $i =$ (index of the downstream parent knot $+ 1$): knot K_{t-i} (bounded by the knots $K_{t-(i+1)}$ and $K_{t-i+\gamma}$) is readjusted to an updated position $K_{t-i-\gamma}$ via Eq. (9); end. The above pseudocode is represented by the bottom box of the flowchart illustrated in Fig. 2.

4 Numerical Experiments

The performance of the knot-placement algorithm introduced in this work was tested against three representative methods borrowed from the literature [1,9,12], each handling the knot-placement problem via a different paradigm, by comparing the generated splines and their properties for a variety of sample datasets. The selected methods are shortlisted in Table 1, labeled as A, B, C, and D, briefly describing the main features of each. In this table, "curvature-guided" refer to methods which rely on a curvature function for localizing knots; "globally iterative" imply recursive generation of splines with the objective of reducing an overall measure of error (deemed an expensive calculation), while locally iterative refers to iterations performed at a narrowed data subset but without generating the spline (thus relatively inexpensive); finally, "heuristic" and "meta-heuristic" refer to the reliance on user-defined rules (not necessarily phenomenological) governing the decision-making process for localizing knots.

Nine dense sample datasets, each briefly described in Table 2 and shown in Fig. 3, were numerically generated with varying levels of complexity and noise-content, and processed by all methods for yielding corresponding splines. The noise was generated by the MATLAB function "rand" based on a uniform distribution and with an average peak-to-peak amplitude equivalent to some percentage of the dataset's maximum amplitude (between 0% and 6%), added to the y-coordinate of the data points (vertical axis in Fig. 3) in all cases except when indicated otherwise in Table 2. For example, noise was added to the x-coordinate (horizontal axis) of sample dataset 8 for the purpose of simulating sampling-uncertainty, while in sample dataset 9 different amounts of noise were added to both coordinates. In all cases, cubic splines were generated and a threshold of $\theta_{\max} = \pi/6$ was used for method A (as justified in Sec. 3.1). Sample datasets 1–4, featuring a function-like behavior, were used to compare methods A and B as the latter can only handle this type of datasets. Sample datasets 5–9, corresponding to parametric curves which cannot be expressed as functions, were used to compare methods A, C, and D.

The comparative analysis was based on three criteria: (a) the overall E_{RMS} error of the fitted spline against the dataset, (b) the size of the knot-vector (i.e., the number of knots), and (c) the computing time involved in tackling the knot-placement process alone, leaving aside all other tasks related to the spline's generation. While the E_{RMS} error and the number of knots of a given spline can be easily measured, finding an objective measure of the computing time is more challenging as this is biased by the coding style used by the programmer. Therefore, the following approach was here adopted: those repetitive tasks associated with the knot-placement process were submitted to MATLAB built-in functions (running on a processor Intel® Core™ i7-4600 U @ 2.10 GHz 64-bit Windows OS with 16GB in RAM), which directly reported the time consumed in delivering a result. For method A, the computing time was represented by the accumulated time reported by the MATLAB function "fminbnd" for solving each submitted nonlinear equation associated with the method. The computed time accounted for method B was represented by the assembly and solution of a linear system for a given knot-vector, as reported by the MATLAB function "spap2" for all global iterations requested (20, 50, and 100, in this study).

Table 1 Selected methods

Method	Label	Main features
This work	A	Curvature-guided Locally iterative Knot-placement criteria based on a mathematical optimization
Yoshimoto et al [1]	B	Based on genetic algorithms Globally iterative, obtaining solutions at 20, 50 and 100 global iterations Knot-placement criteria based on metaheuristic rules
Li et al [9]	C	Curvature-guided Non-iterative Knot-placement criteria based on heuristic rules.
Park and Lee [12]	D	Curvature-guided Globally iterative Knot-placement criteria based on heuristic rules.

Table 2 Sample datasets

Label	Type	Noise level	Main features
1	Function	0.3%	A lightly damped oscillatory signal superimposed over a ramp-like curve; very low noise-content.
2	Function	0.6%	A highly damped oscillatory signal featuring one rapid oscillation followed by two increasingly slower oscillations; low noise-content.
3	Function	3%	A complex, random-like signal conformed of several superimposed high-frequency signals mounted over a ramp-like signal; moderate noise-content.
4	Function	6%	A high frequency sinusoidal superimposed over a low frequency sinusoidal; high noise-content.
5	Parametric	0%	A sequence of low-to-high-to-low curvature regions; no inflexion points; noise-free.
6	Parametric	0%	Two extended low-curvature regions joined by a short high-curvature knee; noise-free.
7	Parametric	1%	Two sequences of a moderate high-curvature followed by an extended low-curvature region, both joined by a short high-curvature knee; low noise-content.
8	Parametric	1.3% @X	An extended low curvature region followed by a rapid change to a high curvature region containing a knee; unlike the previous sample-datasets, the low noise-content is added to the x-coordinate, simulating sampling uncertainty.
9	Parametric	1% @X 0.5% @Y	Extended quasilinear regions with a superimposed slow oscillatory signal; the low noise-content is added both to the x- and y-coordinates.

For method C, various arithmetic operations associated with the process of defining the knots were accounted for, while the computing time consumed by method D was that reported by the MATLAB function "spap2" for all the global iterations needed to converge to a final knot-vector. In the following description, the computing time and the number of iterations required by each method are reported as a range, according to the minimum and maximum values found during the analyses of all sample datasets.

4.1 Method a Versus Method B. Comparative results between methods A and B are summarized in Table 3. The most striking although expected difference observed between both methods is related to the computing time, with method B consuming between 50 and 6260 times that of method A, depending on the dataset considered. The reason behind this can be explained as follows: method B took between 18 and 99 local iterations to solve a linear system of equations for each knot-vector proposed, and this process occurred for each of the 20, 50, and 100 global iterations requested, for a total computing time between 6 and 75 s. On the other hand, method A took about 3–6 local iterations to solve a one-dimensional, convex, and bounded nonlinear equation for each knot defined, for a total computing time between 0.02 and 0.14 s.

Beyond the demanding computational resources, a significant drawback of method B is that its iterative behavior does not follow an increasing monotonic trend in terms of accuracy. That is, more iterations do not necessarily yield higher accuracy, and the latter is not necessarily linked to a smaller knot-vector. On one hand, the error and the number of knots exhibited an opposite trend in sample dataset 1, with the error growing with increasing iterations (1.19, 1.51, and 2.52×10^{-3}, respectively), while the number of knots decreased (58, 49, and 38, respectively), and with the largest error coincident with the smallest number of knots. On the other hand, the analysis for sample dataset 2 yielded the lowest error (1.93×10^{-3}) at 20 iterations, while the smallest knot-vector (13 knots) was obtained at 50 iterations. Yet a different behavior was observed for sample dataset 3, where a low-high-low zigzagging trend in the error was observed as iterations increased, but this was associated with an inverted trend (high-low-high) in the number of knots. Again, the highest error (75.1×10^{-3}) was coincident with the smallest number of knots (22). A particular observation for sample dataset 4 is that an identical number of knots (16 knots) was associated with two very different error measures, 41.0 and 18.1×10^{-3}, for 20 and 50 iterations, respectively. This erratic behavior leaves the user with no choice but to request a large number of iterations to identify the best solution, which often cannot be unambiguously asserted, thus exacerbating an already heavy computational burden.

In spite of the huge difference in computing time, method A consistently yielded a number of knots at least comparable but often much lower than the low bound provided by method B, for all sample datasets.

The following remark is worth mentioning with regard to the lowest number of knots yielded in sample dataset 3 (22 knots in method B versus 35 knots in method A, suggesting a superior performance of method B): the solution found by method B really represents a faulty fit in view of the very high error obtained (75.1×10^{-3} in method B versus 7.65×10^{-3} in method A). This can happen in any situation in which very few knots attempt to capture a very complex signal; the best knot-vector in this case is still provided by method A, and a similar remark can be made for sample dataset 4.

With regard to noise-sensitivity, both methods seemed equally vulnerable as evidenced by the sample dataset 4, exhibiting the highest noise-content (6%) and about 10 times higher errors than those yielded by the sample dataset 1, having a very low noise-content (0.3%). Furthermore, both methods seemed equally vulnerable to the level of complexity of the dataset. This can be noticed in the complex sample dataset 3, yielding errors of comparable magnitude to those of the less complex sample case 4 in spite of the latter having a higher noise-content. It is reminded that the highest error observed in Table 3 (75.1×10^{-3}) is really associated with a faulty fit caused by very few knots attempting to fit a complex curve, rather than to the noise level. Having said this, method A was able to at least closely match the low bound error of method B for all sample datasets, while often obtaining much lower error measures and consistently producing smaller knot vectors.

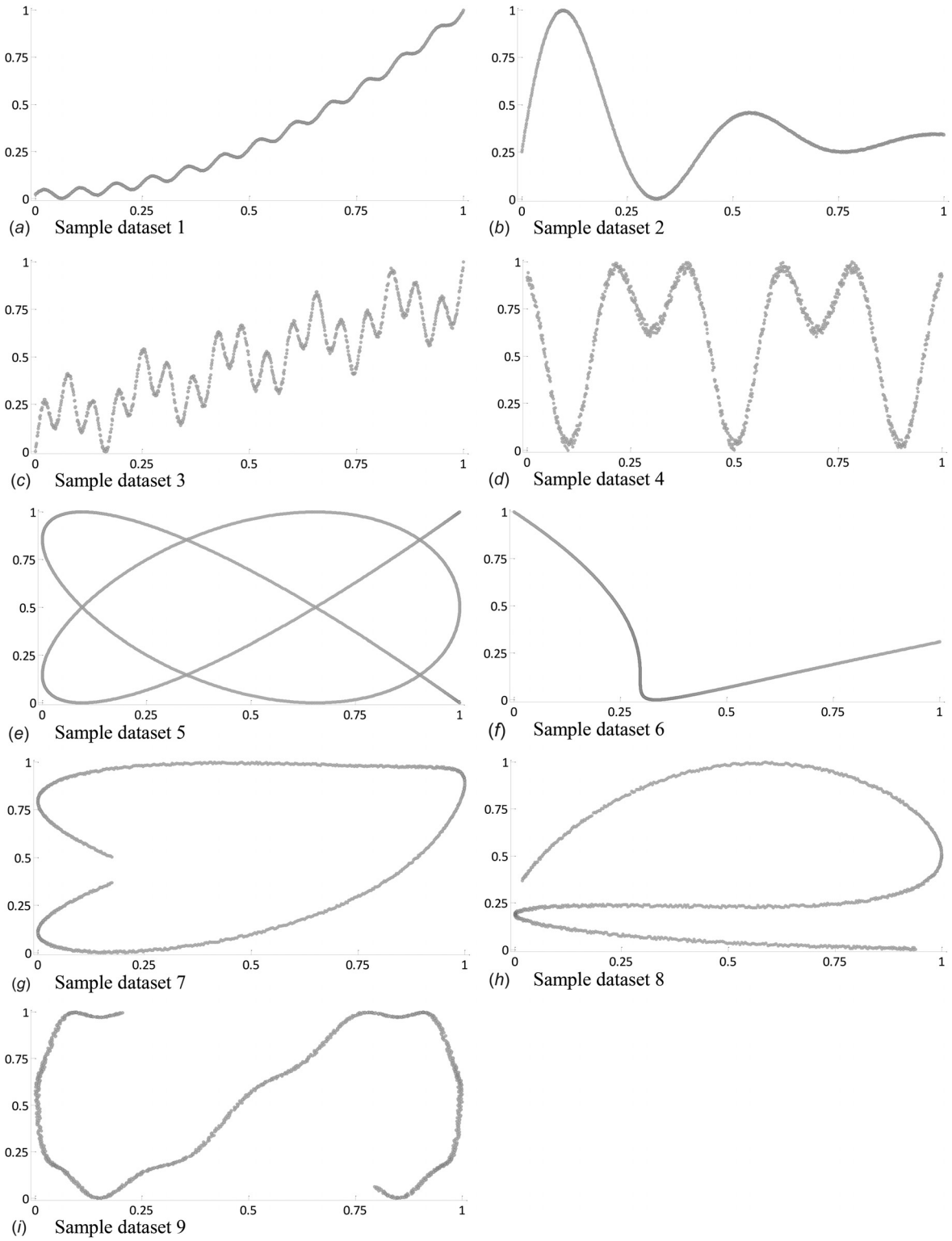

Fig. 3 The nine sample datasets used in this work for comparing the quality of the splines generated by each of the selected methods

Table 3 Method A versus method B

Sample dataset	Method	E_{RMS} ($10\times^{-3}$)	# of knots	Normalized computing time
1	A	1.19	25	1
	B (20 iter)	1.19	58	1264
	B (50 iter)	1.51	49	3200
	B (100 iter)	2.52	38	6260
2	A	2.34	14	1
	B (20 iter)	1.93	15	50
	B (50 iter)	1.98	13	129
	B (100 iter)	1.94	14	253
3	A	7.65	35	1
	B (20 iter)	12.6	67	658
	B (50 iter)	75.1	22	1627
	B (100 iter)	7.53	72	3387
4	A	19.5	15	1
	B (20 iter)	41.0	16	156
	B (50 iter)	18.1	16	367
	B (100 iter)	30.5	10	768

4.2 Method a Versus Methods C and D. Comparative results between methods A, C, and D are summarized in Table 4. These methods are classified as "curvature-guided," relying on a curvature function as key input for the knot-placement process, where the quality of the curvature function (overall smoothness, well-defined and accurate extrema and inflexion points, etc.) has a direct impact on the algorithm's ability to search for suitable knot locations. In the interest of performing an objective comparison of the knot-placement approaches only, without possible bias introduced by the quality of the input data and other side-processes, a standardized curvature function was sought to be supplied to the three methods.

On one hand, method C [9] reports a specific technique for obtaining a filtered curvature function, implemented in this work for the purpose of a faithful reproduction of the method. However, it was found that, although it worked fairly well for datasets with little noise-content (<1%), it yielded unusable results for datasets with moderate-to-high levels of noise (3–6%). In comparison, the digital filter adopted in this work [29,30] consistently yielded a higher-quality curvature function and, when applied to method C, better end results in terms of spline's accuracy and number of knots. On the other hand, method D does not report using a specific technique for obtaining the curvature function [12]. In view of the above, it was decided to adopt the filter described in Refs. [29] and [30] to obtain a curvature function to be supplied to the three methods. In all cases, a chord-length parameterization was used, as reported for methods C and D [9,12] and also adopted by method A. The aforementioned scheme ensures that any difference in the comparative analyses can be attributed to each method's knot-placement process.

Table 4 Method A versus C and D

Sample dataset	Method	E_{RMS} ($\times 10^{-3}$)	# of knots	Normalized computing time
5	A	0.92	25	1
	C	0.99	29	<1
	D	3.69	30	7.6
6	A	0.17	10	1
	C	0.41	10	<1
	D	0.35	10	4.5
7	A	2.87	20	1
	C	3.31	78	<1
	D	3.00	35	11
8	A	4.36	15	1
	C	4.24	80	<1
	D	4.45	25	9.4
9	A	3.23	43	1
	C	2.85	253	≈2
	D	3.24	55	56

With regard to the number of knots, method A consistently yielded the lowest measure by a factor of 1–6, depending on the compared method and sample dataset. In some cases, all three methods yielded similar error measures for a very different number of knots, which is evidence that the extra knots defined by methods C and D represent redundant information. Moreover, method A was able to yield at least comparable but often much lower error measures than the low bound obtained in all sample datasets. As expected, a significant difference was found in the computing time consumed by the "locally iterative" method A versus the "globally iterative" method D, with the latter taking as much as 4.5–56 times that of the former, according to the information reported by the MATLAB functions involved. However, it was not possible to reliably measure the computing time of method C as the arithmetic operations involved in its knot-placement process did not require any specific MATLAB function to be recalled, thus the estimation could be biased by the coding style used. Nonetheless, although it is almost certain that the noniterative nature of method C lends to the shortest computing time among those compared, estimations of its performance during the sample dataset 9, which yielded a large number of knots (253), indicate that its computing time was about twice as that of method A (which defined 43 knots only). This suggests that, in general, the computing time of methods A and B lie within the same order of magnitude.

4.3 The Effect of Knot-Adjustment in Method A. The assessment of the quality of a fitted spline on the basis of its E_{RMS} error only is somewhat subjective, as the global nature of this tool makes it almost insensitive to spline's local oscillations which, although small, can potentially generate significant inaccuracies when the splines are submitted to arithmetic and differential operations, as occurring in recently developed applications [6]. During the course of this research, it was observed that a primary cause of such oscillations is the existence of tightly clustered knots, a condition likely to develop at regions of high curvature or geometrical complexity if the knot-placement method does not have the ability of readjusting the location of existing knots in the vicinity of a newly defined knot. The knot-placement scheme of Method A does contemplate such readjustment and one of its practical implications is that the existing knots in the vicinity of highly knot-populated regions are likely to be "pushed away" from a newly defined knot, thus reducing the possibility of local oscillations. This feature is absent in methods C and D, which keep adding knots to an existing set until some error criterion is met, while method B does not readjust individual knots but rather conforms a whole new knot-vector at every iteration.

An example of the effectiveness of the knot-readjustment scheme of method A can be found in Table 5 and Fig. 4, the latter displaying zoomed views of the "knee" region in sample dataset 6 as well as the splines and correspondent knots generated by methods A, C, and D.

This case study was selected because all methods yielded identical number of knots (10) and very low E_{RMS} errors, thus it is possible to investigate the effect of the knot-readjustment scheme without the bias induced by differences in those two criteria. To this end, method A was applied without and with the readjustment scheme in which results are shown in Figs. 4(*a*) and 4(*b*), respectively. The effect of the readjustment in method A can be noticed straightaway, first generating a spline with an oscillatory behavior around the dataset (without) then a smooth fit virtually superimposed over the dataset (with). This improvement can be attributed to the wider spacing between knots in Fig. 4(*b*) as compared to Fig. 4(*a*), the direct result of the knot-readjustment scheme. Although the three methods yielded very low error measures, the effect of the knot-readjustment in method A led to a reduction of about 60% in the E_{RMS} error.

Table 5 Effect of the knot-readjustment scheme in method A

Sample dataset	Method	E_{RMS} (10×10^{-4})	# of knots (total)	# of knots within zoomed region
6	A (with knot-readjustment)	1.777	10	5
	A (without knot-readjustment)	5.277	10	5
	C	4.087	10	7
	D	3.480	10	6

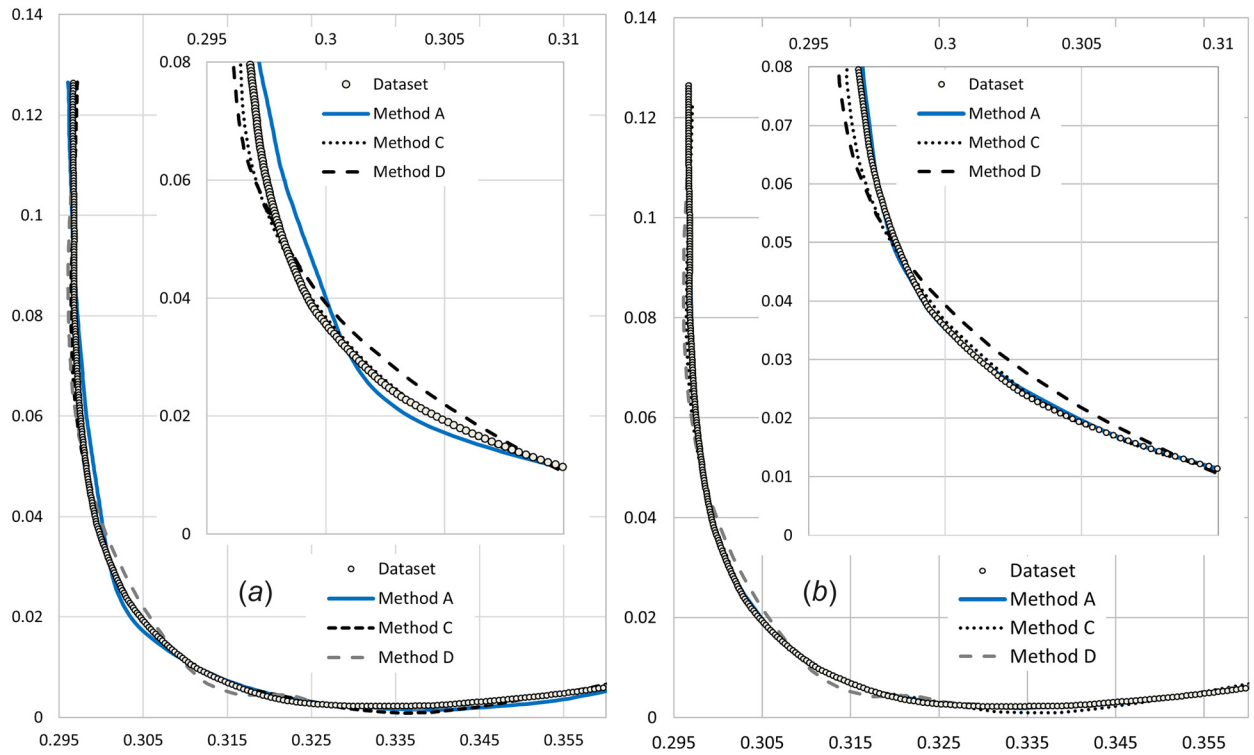

Fig. 4 Splines and correspondent knots generated by methods A, C, and D for a zoomed view of the sample dataset 6. Method A is applied (*a*) without the knot-readjustment scheme and (*b*) with the knot-readjustment scheme. *Insets:* Zoom into the critical near-vertical regions.

5 Conclusion

This paper introduced an adaptive curvature-guided knot-placement algorithm for fitting splines, demonstrated to be particularly effective for datasets with high curvature and/or high data point density for which the location of the knots is a nontrivial task. The scope of the method is restricted to the knot-placement scheme, while all other tasks required for generating the spline (i.e., filtering, curvature extraction, and parameterization) were performed via the existing techniques. The main strengths of the novel algorithm can be stated as (1) efficient curvature-guided approach based on a phenomenological criterion instead of heuristic rules, leading to a compact-sized knot-vector, (2) locally iterative (i.e., at a data subset level), leading to very low computing times, and (3) adaptive, that is, virtually free of user-intervention. The introduced method was tested against three established methods, each tackling the knot-placement problem via a different paradigm. In all numerical experiments performed, the introduced method consistently obtained comparable results (at least) or greatly exceeded (in the vast majority of cases) the best results obtained by the benchmarks in terms of the three criterions considered: (a) RMS error (up to ten times less), (b) number of knots (up to six times less), and (c) computing time (up to 6260 times less).

A knot-readjustment scheme, developed as part of the new method, can be applied to the existing knots in the vicinity of a newly created knot. This provides the ability of dispersing knots from otherwise highly knot-populated regions typically occurring at high curvature zones, an undesirable feature in fitted splines known to generate local oscillations. In summary, the novel methodology presented in this paper has been shown to be a significant contribution to the state of the art in spline-based curve fitting. Compact knot-vectors lead to high data compression ratios, which is why the new method should be useful for a host of applications where the curve fitting process may have to be repeated frequently while maintaining high accuracy, as demonstrated by the authors' group in Ref. [6].

Acknowledgment

Support from the following Institutions is greatly acknowledged: Tecnológico de Monterrey, through the School of Engineering and Science; CONACYT (Mexico), through the Grant SEP-CONACYT CB-2009-01-127942; The National Science Foundation (USA) under Grant no. 1031036; The New York State Energy Research and Development Authority (NYSERDA) under Grant nos. 18812 and 18460; Julie Smith for proofreading the text.

Funding Data

- Consejo Nacional de Ciencia y Tecnologia (SEPCONACYT CB-2009-).
- Instituto Tecnologico y de Estudios Superiores de Monterrey (School of Engineering).
- National Science Foundation (1031036).
- New York State Energy Research and Development Authority (18812 and 18460).

References

[1] Yoshimoto, F., Moriyana, M., and Harada, T., 1999, "Automatic Knots Placement by a Genetic Algorithm for Data Fitting With a Spline," SMI'99, Washington, DC, Mar. 1–4, p. 162.

[2] Yoshimoto, F., Harada, T., and Yoshimoto, Y., 2003, "Data Fitting With a Spline Using a Real Coded Genetic Algorithm," Comput.-Aided Des., **35**(8), pp. 751–760.

[3] Gálvez, A., and Iglesias, A., 2011, "Efficient Particle Swarm Optimization Approach for Data Fitting With Free Knot B-Splines," Comput. Aided Des., **43**(12), pp. 1683–1692.

[4] Hughes, T. J. R., Cottrell, J. A., and Bazilevs, Y., 2005, "Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement," Comput. Methods Appl. Mech. Eng., **194**(39–41), pp. 4135–4195.

[5] Cottrell, J. A., Hughes, T. J. R., and Bazilevs, Y., 2009, *Isogeometric Analysis: Toward Integration of CAD and FEA*, Wiley, Hoboken, NJ.

[6] Cárdenas, D., Elizalde, H., Marzocca, P., Probst, O., Ramirez, R., and Toledo, J. P., 2014, "The Poly-SAFE Method: A Semi-Analytical Representation of Finite Element Models Via Nested Polynomial Reduction of Modal Data," Compos. Struct., **111**, pp. 301–316.

[7] Seo, Y. D., Kim, H. J., and Youn, S. K., 2010, "Shape Optimization and Its Extension to Topological Design Based on Isogeometric Analysis," Int. J. Solids Struct., **47**(11–12), pp. 1618–1640.

[8] Seo, Y. D., Kim, H. J., and Youn, S. K., 2010, "Isogeometric Topology Optimization Using Trimmed Spline Surfaces," Comput. Methods Appl. Mech. Eng., **199**(49–52), pp. 3270–3296.

[9] Li, W., Xu, S., Zhao, G., and Goh, L. P., 2005, "Adaptive Knot Placement in B-Spline Curve Approximation," Comput.-Aided Des., **37**(8), pp. 791–797.

[10] Park, H., Kim, K., and Lee, S.-C., 2000, "A Method for Approximate NURBS Curve Compatibility Based on Multiple Curve Refitting," Comput.-Aided Des., **32**(4), pp. 237–252.

[11] Park, H., 2004, "An Error-Bounded Approximate Method for Representing Planar Curves Is B-Splines," Comput.-Aided Geom. Des., **21**(5), pp. 479–497.

[12] Park, H., and Lee, J.-H. B., 2007, "Spline Curve Fitting Based on Adaptive Curve Refinement Using Dominant Points," Comput.-Aided Des., **39**(6), pp. 439–451.

[13] Robinson, E. C., Jbabdi, S., Andersson, J., Smith, S., Glasser, M. F., Van Essen, D., and Jenkinson, M., 2013, "Multimodal Surface Matching: Fast and Generalizable Cortical Registration Using Discrete Optimization," *Information Processing in Medical Imaging*, Springer, Berlin, pp. 475–486.

[14] Zhao, X., Zhang, C., Yang, B., and Li, P., 2011, "Adaptive Knot Placement Using a GMM-Based Continuous Optimization Algorithm in B-Spline Curve Approximation," Comput.-Aided Des., **43**(6), pp. 598–604.

[15] Ulker, E., and Arslan, A., 2009, "Automatic Knot Adjustment Using an Artificial Immune System for B-Spline Curve Approximation," Inf. Sci., **179**(1), pp. 1483–1494.

[16] Jupp, A. L., 1978, "B Approximation to Data by Splines With Free Knots," J. Numer. Anal., **15**(2), pp. 328–343.

[17] Valenzuela, O., Delgado-Marquez, B., and Pasadas, M., 2013, "Evolutionary Computation for Optimal Knots Allocation in Smoothing Splines," Appl. Math. Modell., **37**(8), pp. 5851–5863.

[18] Jiang, S., Wang, Y., and Zhicheng, J., 2014, "Convergence Analysis and Performance of an Improved Gravitational Search Algorithm," Appl. Soft Comput., **24**, pp. 363–384.

[19] Razdan, A., 1999, *Knot Placement for B-Spline Curve Approximation*, Arizona State University, Tempe, AZ.

[20] Asada, H., and Brady, M., 1986, "The Curvature Primal Sketch," IEEE Trans. Pattern Anal. Mach. Intell., **8**(1), pp. 2–14.

[21] Hamann, B., and Chen, J.-L., 1994, "Data Point Selection for Piecewise Linear Curve Approximation," Comput. Aided Geometric Des., **11**(3), pp. 289–301.

[22] Pei, S.-C., and Horng, J.-H., 1995, "Fitting Digital Curve Using Circular Arcs," Pattern Recognit., **28** (1), pp. 107–116.

[23] Rattarangsi, A., and Chin, R. T., 1992, "Scaled-Detection of Corners of Planar Curves," IEEE Trans. Pattern Anal. Mach. Intell., **14**(4), pp. 430–449.

[24] Horn, J.-H., 2003, "An Adaptive Smoothing Approach for Fitting Digital Planar Curves With Line Segments and Circular Arcs," Pattern Recognit. Lett., **24**(1–3), pp. 565–577.

[25] Cao, M. R., Xu, M., and Ostachowicz, W., 2014, "W. Indentification of Multiple Damage in Beams Based on Robust Curvature Mode Shapes," Mech. Syst. Signal Process., **46**(2), pp. 468–480.

[26] Lin, W.-Y., Chiu, Y.-L., Widder, K. R., Hu, Y. H., and Boston, N., 2010, "Robust and Accurate Curvature Estimation Using Adaptive Line Integrals," EURASIP J. Adv. Signal Process., **2010**(1), p. 240309.

[27] Sohn, K., Alexander, W. E., Kim, J. H., Kim, Y., Yoon, S. H., Park, E. H., and Ntuen, C. A., 1991, "Optimal Boundary Smoothing for Curvature Estimation," Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems and Computers, pp. 1220–1224.

[28] Lee, I.-K., 2000, "Curve Reconstruction From Unorganized Points," Comput. Aided Des., **17**(2), pp. 161–177.

[29] García, D., 2010, "Robust Smoothing of Gridded Data in One and Higher Dimensions With Missing Values," Comput. Stat. Data Anal., **54**(4), pp. 1167–1178.

[30] Weinert, H. L., 2007, "Efficient Computation for Whittaker-Henderson Smoothing," Comput. Stat. Data Anal., **57**(2), pp. 959–974.

[31] Farin, G., 2002, *Curves and Surfaces for CADG: A Practical Guide*, Morgan Kaufmann Publishers, San Francisco, CA.

[32] Piegl, L., and Tiller, W., 1995, *The NURBS Book*, Springer-Verlag, New York.

[33] Savitzky, A., and Golay, M. J. E., 1964, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," Anal. Chem., **36**(8), pp. 1627–1639.

[34] Prautzsch, H., Boehm, W., and Paluszny, M., 2002, *Bézier and B-Spline Techniques*, Springer-Verlag, New York.

[35] Su, B. Q., and Liu, D. Y., 1989, *Computational Geometry—Curve and Surface Modeling*, Academic Press, Boston, MA.