University of Cantabria

Department of Applied Mathematics and Computational Sciences

Free-form Curve and Surface Reconstruction Through Simulated Annealing-based Metaheuristic Techniques

---

Reconstrucción de Curvas y Superficies de Forma Libre Mediante Metaheurísticas Basadas en Simulated Annealing

Author/Autor: Carlos Loucera Muñecas

Supervisor/Director: Andrés Iglesias Prieto

Supervisor/Directora: Akemi Gálvez Tomida

2017

You must understand that there is more than one path to the top of the mountain

— *Miyamoto Musashi*

Affiliations

Advanced Signal Processing Group (GTAS)
Department of Communications Engineering
University of Cantabria

Thanks to my sponsors.

Acknowledgements

This Thesis, the compilation of almost five years of work, would not have been possible without the support of my friends, colleagues and family.

First of all I wish to thank my advisors, Dr. Andrés Iglesias and Dr. Akemi Gálvez, for putting their trust on me and my abilities to bring the research to a successful conclusion. Their excellent advice has guided my work, particularly in difficult times. I would also like thank the numerous colleagues that I have met during my career, specially Steven, Nacho and Jacobo at the GTAS research group for helping me to grow up as a (data) scientist with their insightful comments.

I am quite grateful to the excellent colleagues that made my life happier at the ICANE, specially the IT section: let the Kanban flow...

Many thanks to Elena and Inés for making my stay in Madrid as life-enriching as it could be. Thanks to Melu, Alberto, María, Juanje and Dani at the AEMET in Madrid, for opening my eyes to the wonderful world of meteorology. Maybe the latter part of this work would not be possible without my pythonic friends Jesús and Mauri.

Thanks to 3DINTELLIGENCE for providing a wonderful dataset to work with. Specially Óscar for his ability to search for new and interesting projects.

La Tesis nunca se habría podido llevar a cabo si no llega a ser por la confianza que siempre pusieron en mí, tanto mis pardres, como mis hermanos y tiucus. Si mi castellano se puede leer, es gracias a Feli, gracias por tu paciencia infinita.

Finalemente, mis más sinceros agradecimientos a quien me ha acompañado desde el principio. La Tesis se ha escrito con lágrimas, café y Mar. Mi vida y la Tesis son mejores gracias a ti.

Carlos Loucera

Santander, April 26, 2017

Abstract

Reverse engineering has become ubiquitous in the computer aided design and manufacturing industry (CAD/CAM). One of the most sought after tools in many industries (automotive, aerospace, shipbuilding) and scientific fields (computer tomography, magnetic resonance) is the ability to build a digital model from a 3D-scanned real world object. In this Thesis we propose a methodology to automatically find an optimal free-form model that fits a given point cloud: the most common output of real-world measurements.

In most real-world scenarios the data provided is devoid of any kind of information beyond the sample points, i.e. both the geometry and topology of the point cloud are unknown. Furthermore, the given data is usually of massive size with many different types of noise added: from error measurements to missing points.

On the contrary, a free-form mathematical model can represent a given shape with very few parameters, so if the quality of the fit is guaranteed, we can represent a large point cloud with a very compact model. In this Thesis we propose a set of spline models for approximating the data. We start with the simple, yet powerful, non-rational Bézier curves and surfaces, a linear combination of Bernstein polynomials, and as the complexity of the data increases we introduce the rational versions. Finally, we use B-spline models, a superset of the Bézier family, to reconstruct the most challenging scenarios.

Our methodology is based on three techniques, namely: least-squares regression, the Simulated Annealing optimization algorithm and two information sciences criteria. The first step consist of transforming the geometrical problem of reconstructing the shape of data into an optimization problem taking advantage of the the least-squares regression procedure. The resulting problem turns to be a highly non-linear system of very difficult solution. To overcome the minimization of such a challenging functional we make use of the Simulated Annealing algorithm, a powerful meta-heuristic that mimics the thermodynamics behind the cooling of a metal. By means of this stochastic-driven optimization method we retrieve the functional architecture of our baseline spline model. These two steps are repeated for a range of baseline splines of varying complexity. Finally we search the best among these candidate models by means of either the Akaike or Bayes Information Criteria.

The Thesis is divided into four main parts. First, we introduce the mathematical concepts needed to understand the problem. We continue with the proposed methodology, with specific emphasis on the Simulated Annealing. Once all the necessary elements have been carefully explained, we provide a set of illustrative examples to show the performance of our method. We conclude with an outline of the main re-

sults of this work, our contributions to the state of the art and some future lines of research.

Resumen del Trabajo, Resultados y Propuestas Futuras

Procuremos agradar e instruir;
nunca asombrar.

Santiago Ramón y Cajal

De acuerdo con la normativa que regula los estudios de doctorado de la Universidad de Cantabria, se incluye a continuación un resumen de los principales resultados y conclusiones de la Tesis Doctoral.

Ingeniería inversa

La creación de modelos digitales se ha convertido en una de las principales herramientas en la cadena de producción de la industria de diseño y manufactura. A medida que las tecnologías para la fabricación avanzan, como por ejemplo el auge en la impresión 3D casera, la capacidad para producir objetos reales a partir de diseños digitales se ha convertido en una tarea casi trivial. En cambio, el proceso inverso, consistente en la reconstrucción de un modelo computacional a partir de un objeto real, continúa siendo un problema de enorme dificultad que en su forma más general está aún por resolver. Éste último proceso, conocido como ingeniería inversa, juega un papel vital en la industria de la fabricación de hoy en día, especialmente en sectores como la automoción, el aeroespacial y el naval [Pottmann et al., 2005].

A lo largo del presente trabajo nos vamos a centrar en un escenario muy concreto dentro del campo de la ingeniería inversa, a saber: el ajuste de curvas y superficies. El problema consiste en construir una curva o superficie a partir de una nube de puntos ruidosos, por lo general de tamaño gigantesco [Pauly et al., 2002]. Para tratar el problema en su forma más general, no se asume la existencia de más información aparte de los propios puntos dato, es decir, no se conoce ni la geometría ni la topología de la nube, por lo que la conectividad entre los puntos de la muestra es desconocida. El objetivo principal de la ingeniería inversa de curvas y superficies reside en la reconstrucción de un modelo matemático que capture el máximo posible de información de la función subyacente tras los datos: forma, topología, geometría etc.

Son numerosos los campos de la ciencia donde la reconstrucción de curvas y superficies juega un papel esencial. Así, por ejemplo, el ajuste de datos mediante splines es una importante herramienta en el aprendizaje automático, dado que

es el soporte de varias técnicas de regresión (tanto lineales, como no lineales) [Marsh and Cormier, 2001], la teoría de la aproximación [Rice and Saloin, 1969, Cox, 1990] y el diseño geométrico asistido por computador [Dierckx, 1995]. Más allá del universo de las matemáticas más clásicas, la ingeniería inversa de curvas y superficies es clave en distintas áreas de la medicina no invasiva, como por ejemplo la recreación de superficies a partir de distintas secciones laminadas adquiridas mediante tomografía computacional o resonancia magnética [Bajaj et al., 1995, Park and Kim, 1996].

Otra área donde la búsqueda de una curva o superficie cumple una función importante es en la visualización de determinados parámetros en las ciencias atmosféricas, como por ejemplo el uso de las Thin Plate Spline (un modelo matemático basado en funciones de base radial) para estimar la lluvia [Tait et al., 2006], un uso aprobado por la propia Organización Meteorológica Mundial (WMO) [Hutchinson, 1995]. En general, en la mayoría de los escenarios donde se usa el kriging (interpolación mediante procesos Gaussianos), su uso se puede intercambiar por la aproximación mediante algún tipo de spline [Hutchinson and Gessler, 1994], como los modelos de temperatura y elevación en superficie.

Tal y como ya se ha mencionado anteriormente, lo más frecuente es tener que reconstruir curvas y superficies mediante el ajuste de nubes de puntos desorganizadas, por lo general de gran tamaño y con un gran contenido de información errónea (ruido) debido a errores en los procesos mecánicos u ópticos de captura de puntos, como escáneres 3D [Hoppe et al., 1992, Eck and Hoppe, 1996, Kazhdan and Hoppe, 2013]. A fin de reconstruir el modelo matemático subyacente existen numerosas propuestas en la literatura. En general se puede decir que existen tres grandes familias de técnicas de reconstrucción, a saber: la aproximación mediante mallados poligonales, la geometría constructiva de sólidos y el modelado mediante curvas y superficies matemáticas de forma libre. El método propuesto en la presente Tesis pertenece a la tercera categoría.

Nuestro objetivo consiste en reconstruir nubes de puntos ruidosos, mediante el uso de unos modelos matemáticos de forma libre, conocidos como splines. A tal efecto, en primer lugar, se transforma el problema geométrico de la reconstrucción, en uno de optimización matemática. Dicho problema se sabe que es continuo, altamente no lineal, no-convexo y multi-modal [Jupp, 1978, Lane and Riesenfeld, 1983], por lo que es de muy difícil solución. Durante las décadas de los sesenta y setenta aparecieron las primeras investigaciones dedicadas al estudio del ajuste mediante splines, al albor del desarrollo de una teoría general matemática, en torno a los propios modelos matemáticos. Estas incipientes técnicas se basaban principalmente en el uso de los algoritmos clásicos de optimización [de Boor and Rice, 1968, Rice and Saloin, 1969, Powell, 1970] que, aunque capaces de resolver algunos ejemplos, eran incapaces de resolver el problema general, debido a la existencia de numerosas cuencas de atracción, de las que resultaba casi imposible escapar [Jupp, 1978]. Durante las siguientes décadas aparecieron multitud de nuevas técnicas, tales como la división de los datos en pequeñas regiones apropiadas para la interpolación mediante funciones polinómicas [Cox et al., 1990], métodos de búsqueda lineal basados en umbrales sobre el error [Park, 2004], la simplificación del proble-

ma de aproximación mediante el uso de puntos *dominantes* [Park and Lee, 2007] y varios otros métodos caracterizados por el uso de técnicas iterativas, basadas en la construcción de un primer modelo base que posteriormente es mejorado mediante heurísticas (basadas en datos), entre las que destacamos las presentadas en [Ma and Kruth, 1995, Piegl and Tiller, 2001, Brujic et al., 2011]. Estos métodos tienden a fallar cuando los datos están corruptos o incompletos, algo habitual en los ejemplos no académicos. Durante los últimos años, la comunidad científica ha mostrado que el problema puede abordarse de forma más general mediante la aplicación de técnicas de optimización difíciles de encuadrar dentro de la matemática clásica, tales como la inteligencia artificial o la computación bio-inspirada. Dado que nuestro método se encuentra encuadrado dentro de éste último grupo, se analizará con algo más de detalle la literatura al respecto.

Figura 1: Ejemplo de una nube de puntos obtenida a partir de una talla de madera de una Virgen en la catedral de Santander. Proporcionada por la empresa 3DINTELLIGENCE.

Modelado con splines

El modelado con splines se ha convertido en el estándar de referencia para la industria del diseño y la manufactura (CAD/CAM). En términos matemáticos un spline es una combinación lineal de un conjunto de funciones, que forman una base del espacio funcional, a través de unos coeficientes conocidos como puntos de control. Las funciones base, también llamadas de *modelado* o *deformación*, poseen una serie de propiedades que las hacen especialmente adecuadas para su uso en el campo

de los gráficos por computador. A continuación, se presentarán algunas de las más relevantes:

- *Compresión de información*. Las splines, en su forma canónica, contienen grandes dosis de información acerca de la forma que representan, con el uso de unos pocos parámetros, por lo que son modelos especialmente adecuados para su uso computacional.
- *Construcción de formas.* Con un único modelo matemático se pueden llegar a representar una gran variedad de formas, desde los modelos clásicos (cónicas, superficies regladas y de revolución, etc.) hasta modelos paramétricos de forma libre.
- *Facilidad de manipulación.* Gracias a las propiedades geométricas y analíticas de las funciones base, la forma final se puede modificar de forma sencilla, incluso *localmente* dependiendo de la familia spline, con tan sólo modificar los puntos de control. Además la interpretación geométrica del modelo es clara y concisa, por lo que son excelentes candidatos para su uso en el diseño geométrico asistido por computador.
- *Invariancia geométrica.* Son invariantes bajo transformaciones afines (escalado, rotación, traslación); es más, basta con aplicar la transformación sobre los puntos de control.
- *Estabilidad numérica.* Existen algoritmos que permiten evaluar y modificar un spline de forma numéricamente estable y rápida.

Técnicas meta-heurísticas

Tal y como se ha mencionado anteriormente, el problema de la reconstrucción de curvas y superficies es de una enorme dificultad, y no puede ser debidamente resuelto con métodos matemáticos clásicos. A fin de superar dichas limitaciones, la comunidad científica ha trasladado su atención hacia otras aproximaciones menos ortodoxas: desde el uso de técnicas propias de la inteligencia artificial, hasta la computación bio-inspirada.

Dentro del paradigma de la inteligencia artificial, las redes neuronales artificiales (ANN) se han utilizado con diversos grados de éxito para la resolución del problema en cuestión; así, por ejemplo en [Gu and Yan, 1995, Hoffmann and Varady, 1998, Barhak and Fischer, 2001], se resuelve parcialmente el caso del ajuste de nubes de puntos desorganizadas mediante superficies. Mientras que en [Kumar et al., 2004, Hoffmann, 2005] se resuelve aprovechando las capacidades de los mapas auto-organizados (SOM) para aprender la conectividad de los puntos dato. Una de las grandes ventajas de estas aproximaciones radica en la capacidad que tienen las redes artificiales para auto-inferir la topología de la nube de puntos. Sin embargo, los modelos resultantes son incapaces de aprender la estructura funcional del problema, necesitando, además, ser provistos de una arquitectura de red: un problema

muy dependiente de los datos y de gran dificultad en sí mismo. Las redes funcionales solucionan parte de dichos problemas, al sustituir los pesos escalares por pesos funcionales más apropiados para el problema. Un buen ejemplo de tal aplicación lo encontramos en [Iglesias et al., 2004], donde se resuelve la reconstrucción de superficies B-spline, y en [Iglesias et al., 2004] donde las redes funcionales se hibridan con algoritmos genéticos, a fin de resolver el problema del ajuste mediante curvas y superficies de Bézier. Finalmente, en [Iglesias and Gálvez, 2008] se extiende la aproximación funcional, para incluir las fucniones base de las B-spline racionales para la aproximación de curvas.

La optimización bio-inspirada es un área de conocimiento que se basa en la premisa de tratar de emular computacionalmente los procesos propios de la Naturaleza: desde la manera de pensar y procesar la información, hasta cómo resolver complejos sistemas biológicos. La mayoría de los algoritmos de optimización bio-inpirada se basan en sistemas de generación estocástica de soluciones, cuyo soporte se basa en metáforas que explican el proceso natural que imitan. Cabe señalar que en el corazón de nuestra metodología reside una implementación propia del algoritmo conocido como Simulated Annealing [Kirkpatrick et al., 1983], una meta-heurística basada en el proceso termodinámico del recocido de un metal. A continuación, presentamos una serie de meta-heurísticas que han sido empleadas con diferentes grados de éxito al problema de la reconstrucción de nubes de puntos ruidosos.

Algoritmos Evolutivos. Los algoritmos evolutivos (EA) forman parte del campo de la computación natural, un área de la ciencia que trata de emular los mecanismos evolutivos empleados por el mundo natural, tales como la mutación, selección, reproducción, adaptación etc. Los algoritmos genéticos (GA) son un subconjunto de los algoritmos evolutivos que trata de imitar computacionalmente el proceso de selección natural. En [Yoshimoto et al., 1999, Yoshimoto et al., 2003] se aplican al caso de las B-splines con nodos libres, con un éxito moderado, pues necesitan de heurísticas a posteriori, para resolver el problema completo; en [Valenzuela et al., 2013] el problema se resuelve utilizando una variante multi-objetivo de los algoritmos genéticos. Los algoritmos de selección clónica (CSA), dentro de los basados en sistemas inmunes (AIS), son un tipo de algoritmo evolutivo, cuya metáfora está basada en la emulación de cómo el sistema inmune es capaz de generar ciertos anticuerpos específicos para contrarrestar la presencia de antígenos invasores. En [Ülker and Arslan, 2009] se utilizan los AIS a fin de resolver el problema de la reconstrucción mediante B-splines, mientras que en [Iglesias et al., 2013, Gálvez et al., 2015] se resuelven los problemas del ajuste, mediante modelos de Bézier y B-spline, a través de la optimización mediante CSA.

Inteligencia de enjambre. Los métodos bajo el paradigma de la inteligencia de enjambre, en cambio, tratan de emular el comportamiento socio-cognitivo de los sistemas poblacionales, en los que se crean reglas idealizadas de cómo los individuos (ya sean humanos, aves, partículas físicas) cooperan para tratar de alcanzar una meta común. La optimización mediante sistemas de partículas (PSO) es un al-

goritmo poblacional basado en sistemas estocásticos que, partiendo de una población aleatoria de soluciones, es capaz de hacerlas evolucionar hasta el óptimo global, mediante una serie de heurísticas que emulan la diseminación de la información acerca del espacio entre las distintas partículas. El algoritmo PSO se usa en [Gálvez and Iglesias, 2011, Gálvez and Iglesias, 2012] para la obtención de curvas B-spline y superficies NURBS, respectivamente. Otras técnicas dentro del paradigma de la inteligencia de enjambre tratan de imitar el comportamiento de ciertos sistemas biológicos; así, por ejemplo, en [Gálvez and Iglesias, 2013] se emula el complejo comportamiento de las luciérnagas, mientras que en [Iglesias et al., 2015a] se hace lo propio con los enjambres de murciélagos.

Basados en la Física. Los algoritmos de optimización basados en la emulación de procesos físicos son todos muy distintos entre sí, pues su única conexión radica en la metáfora derivada de la Física sobre la que soportan sus teorías. La optimización basada en algoritmos electromagnéticos (EMA) basa su metáfora en la emulación de los procesos de atracción-repulsión que pueden mover una partícula hasta los puntos de mínima energía (los óptimos). Este algoritmo se ha usado con éxito para el caso de la reconstrucción mediante curvas de Bézier [Gálvez and Iglesias, 2013] y posteriormente se amplió con el uso de técnicas miméticas, optimizadores locales embebidos dentro de la estructura del algoritmo principal, en [Iglesias and Gálvez, 2016] para el caso de las superficies racionales de Bézier. El algoritmo de Simulated Annealing está encuadrado en esta categoría, siendo unos de los primeros, puesto que su metáfora está basada por completo en los procesos termodinámicos que llevan a la mejora de la estructura interna de un metal. El SA se ha usado con éxito en el caso de la reconstrucción de la triangulación de mallados [Sen and Zheng, 1992] en el caso de las B-spline de nodos libres [Valenzuela and Pasadas, 2010].

Objetivos de la Tesis

El objetivo del presente trabajo consiste en la construcción de una metodología general para la reconstrucción de nubes de puntos ruidosos, mediante el uso de modelos spline. Para construir el método de reconstrucción, se propone el uso de diversas implementaciones del algoritmo Simulated Annealing que sean capaces de mejorar lo existente en la literatura, para el problema del ajuste mediante splines.

La motivación de tal propuesta radica en la capacidad de los algoritmos basados en procesos estocásticos, como el SA, para adaptarse al problema sin necesidad de disponer de toda la información o en caso de que se encuentre distorsionada (ruido). Como ya se ha descrito con anterioridad, el escenario que se asume a lo largo de la Tesis cumple con dichos requisitos: nubes de puntos ruidosos en los que, en su forma más general, se carece incluso de la geometría y topología del conjunto.

Para resolver el problema proponemos el uso de una serie de variantes del SA, adaptados para cada modelo específico de spline: tanto las curvas y superficies de Bézier, como las B-spline, tanto en su modalidad racional, como no-racional. A fin de

poder automatizar la tarea, a la par que evitar el sobre-ajuste de los modelos, envolvemos a los meta-heurísticos dentro de una capa iterativa de selección de modelos, mediante el uso de técnicas de ciencias de la información. Más concretamente, los criterios de información de Akaike y Bayesiano (AIC/BIC).

Finalmente, para el caso de las nubes masivas y desorganizadas, proporcionamos una serie de extensiones a la metodología que mediante una serie de transformaciones son capaces de reorganizar y pre-procesar la nube de forma que pueda ser atacada con la metodología anteriormente descrita. En la Figura 2 se puede ver una represetación simplidficada del diagrama general de nuestro método para el ajuste de curvas y superficies, así como el capítulo en el que se describen cada una de las distintas técnicas.

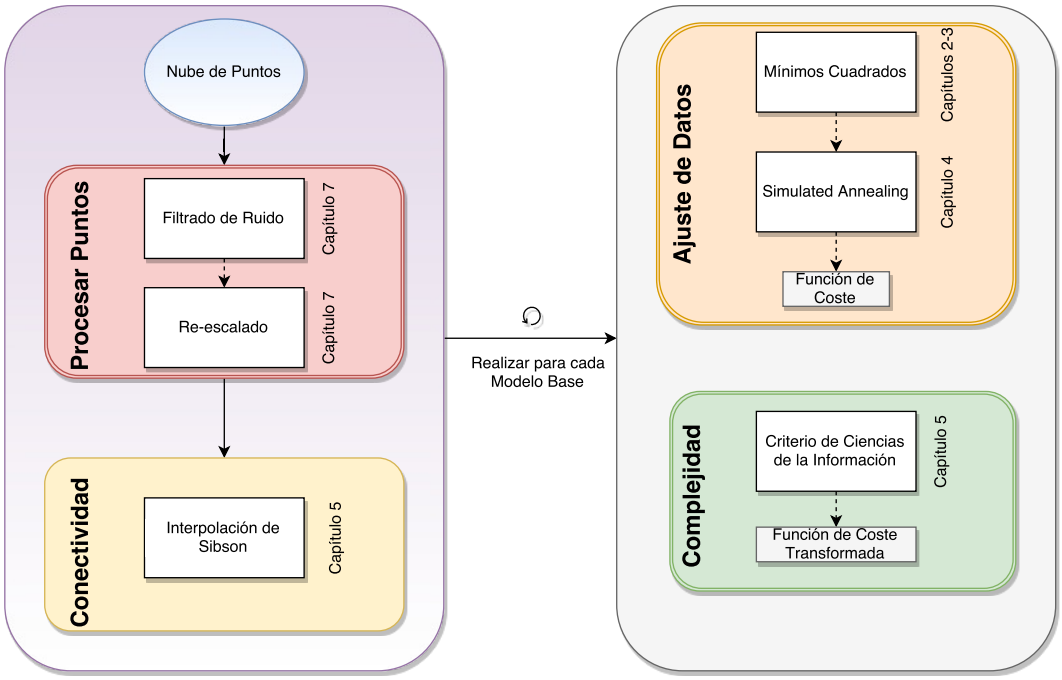


Figura 2: Diagrama general del método.

Estructura de la Tesis

Aunque todos los capítulos están fuertemente ligados entre sí, bajo el marco de la ingeniería inversa, la Tesis está dividida en tres grandes bloques de información. La introducción, tanto en inglés como en castellano, y el trasfondo matemático tras el problema, forman el primer gran bloque. La Tesis continúa con una presentación de los métodos meta-heurísticos empleados, así como un análisis de la metodología empleada. Posteriormente, un tercer bloque queda conformado por la presentación de diversas aplicaciones de la metodología. Finalmente, en el último bloque se presentan las conclusiones y contribuciones derivadas del presente trabajo, así como una breve reseña de las principales líneas de trabajo de cara al futuro.

A fin de comprender mejor el problema resuelto en este trabajo, se incluye a continuación, Figura 3, un diagrama que trata de resumir la metodología propuesta, junto con un breve resumen de cada uno de los bloques de los bloques:

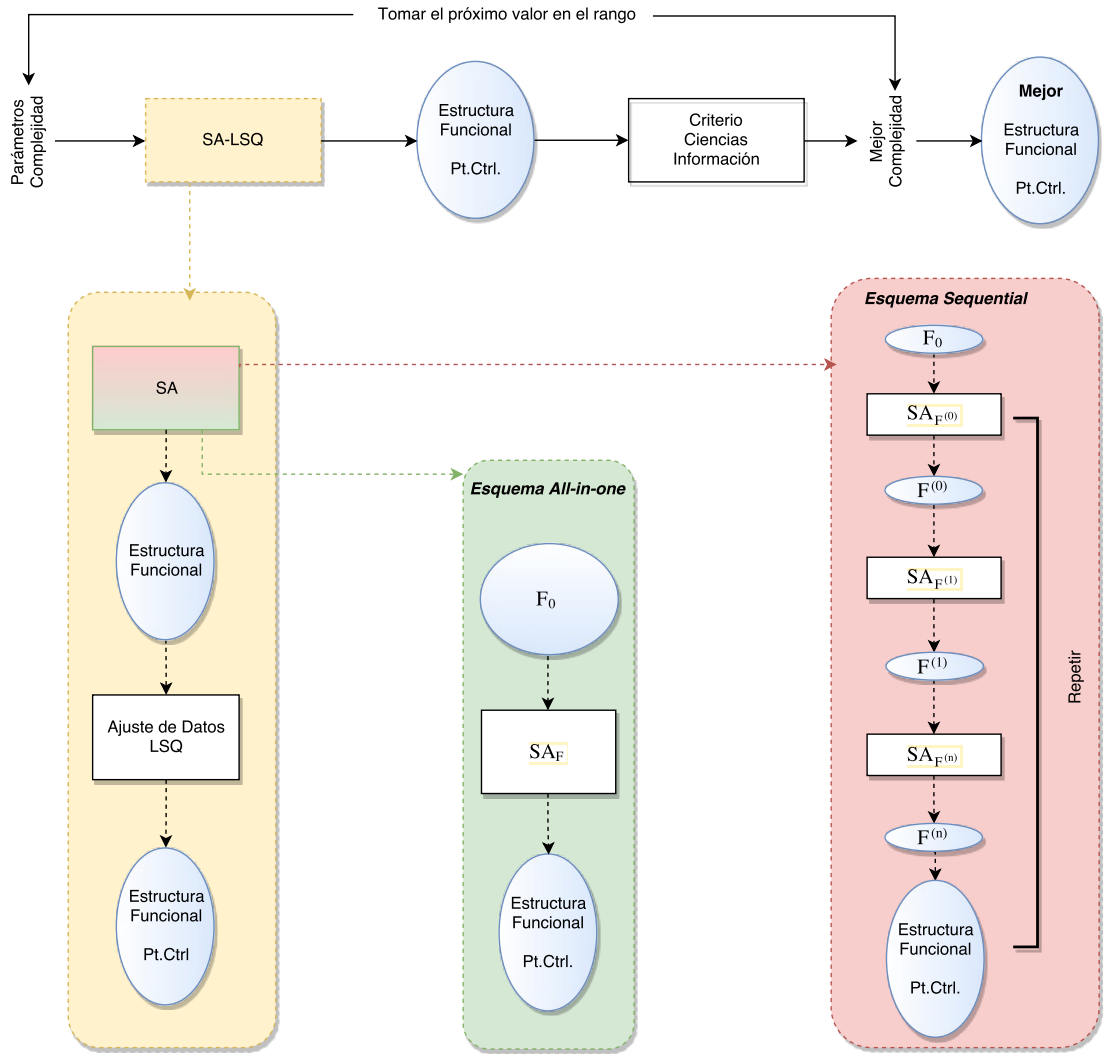


Figura 3: Metodología propuesta para la reconstrucción de curvas y superficies mediante splines.

Complejidad Selección de los rangos para los parámetros que definen la complejidad del modelo subyacente: número de nodos, grado de la curva etc.

LSQ Tranformación del problema geométrico en uno de optimización mediante el asjute por mínimos cuadrados (LSQ). Es decir, se construye el funcional que nos devuelve la *energía* del sistema.

SA Una implementación del algoritmo Simulated Annealing, que recibe o bien la estructura funcional completa del modelo spline, esquema *all-in-one*, o bien una parte de dicha estructura, esquema *sequential*.

CIC Selección del modelo mediante el uso de critierios de ciencias de la información (Akaike o Bayes).

All-in-one Esquema de uso mediante el que se computan todos los parámetros del modelo subyacente en un único paso.

Sequential Esquema de uso en el que se computan secuencialmente los distintos parámetros del modelo subyacente. Se inicializa la estructura funcional completa y progresivamente se van optimizando cada uno de los distintos tipos de parámetros que requiera el modelo: parametrización, nodos, pesos etc.

Resumen de los resultados y conclusiones derivadas

La tercera sección de la Tesis recoge los principales resultados obtenidos al aplicar nuestra metodología, a una serie de ejemplos paradigmáticos del problema a resolver.

Ajuste mediante modelos de Bézier

En el capítulo 6 se presentan los resultados obtenidos al aplicar nuestra metodología al caso del ajuste de nubes de puntos ruidosos, mediante el uso de curvas y superficies de Bézier, tanto en su modalidad racional, como no-racional.

Curvas de Bézier

Los primeros resultados presentados tienen como origen un artículo publicado por los autores en la conferencia internacional *Cyberworlds* del 2014, bajo el nombre *Simulated Annealing Algorithm for Bézier Curve Approximation*. En ese artículo se emplea el método propuesto para el ajuste de nubes de puntos, mediante curvas no racionales de Bézier, a través de una serie de ejemplos paradigmáticos que muestran diversas dificultades desde el punto de vista geométrico: auto-intersecciones, ruido, rápidos cambios de curvatura etc. A fin de computar los parámetros óptimos para la curva de Bézier, se comparan el uso de dos distribuciones para la generación de soluciones combinadas con dos esquemas de enfriamiento.

El método se puede resumir de la siguiente forma: dado un grado n, el método computa una parametrización óptima para la curva de Bézier, mediante el Simulated Annealing, representando la energía del sistema mediante la suma de los residuos del ajuste mediante mínimos cuadrados. A la hora de seleccionar el modelo que ofrece una mejor compensación entre fidelidad y complejidad, se comparan dos criterios del campo de las ciencias de la información: el criterio de Akaike y el de información Bayesiana.

De todas las posibles combinaciones probadas, el esquema de enfriamiento y generación de entornos que aproxima el Fast Simulated Annealing [Szu and Hartley, 1987], junto con el uso del Criterio de Información Bayesiana (BIC), son los que ofrecen resultados más estables, además de converger con mayor velocidad que el resto de esquemas propuestos.

Superficies de Bézier

Los resultados presentados para el ajuste de superficies de Bézier se basan en los presentados bajo el artículo *A simulated annealing approach for data fitting with Bézier surfaces*, publicado en *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication (2015)*. En este artículo se presentó una extensión de la metodología propuesta en [Loucera et al., 2014] para el ajuste mediante curvas de Bézier, al caso de la aproximación mediante productos tensoriales de polinomios de Bernstein. Dados los resultados obtenidos para el caso unidimensional, se optó por continuar con el uso de un esquema basado en la aproximación del Fast Simulated Annealing. Dado que para el caso de superficies es necesaria la computación de las funciones bi-variadas de Bernstein, esto supone un incremento de los parámetros a ajustar, puesto que ahora es necesario un mallado bidimensional, así como la elección del grado óptimo en cada dirección paramétrica. Además, ahora las nubes de puntos son de mayor tamaño, pues necesitan *explicar* una superficie en 3D. Debido al aumento significativo en el número de parámetros a ajustar, y por tanto a modelar al tratarse de superficies de Bézier, el problema de optimización se complica considerablemente. A fin de adaptar el SA a este nuevo escenario, se proponen una serie de modificaciones:

- La generación de nuevos candidatos se realiza mediante la composición de tres funciones, a saber: en primer lugar, se utiliza la aproximación al FSA mediante perturbaciones gaussianas, a su resultado se aplica una función de ajuste a la frontera y finalmente se explota el entorno de la solución mediante una rápida búsqueda local.
- A fin de evitar la alta dependencia en la selección de los parámetros iniciales del algoritmo, se emplea una técnica de reinicialización: se lanzan varias instancias del algoritmo con pocas iteraciones hasta que se alcanza una proporción cercana la 80 % en el cociente de aceptación.
- Estrategia de memorización. En todo momento se mantienen en memoria tanto la mejor solución global como la mejor solución de cada ciclo de equilibrio térmico.
- Al final del ciclo exterior se realiza una búsqueda para cada una de las soluciones memorizadas.

Con objeto de asegurar la validez del método, se ha probado el método en una serie de ejemplos académicos, mostrando diversas dificultades geométricas. En todos ellos, nuestro método es capaz de obtener excelentes resultados, tanto desde el punto de vista de minimización, como de visualización.

Curvas racionales de Bézier

Las curvas de Bézier adolecen de cierta rigidez, ya que, dada su composición polinómica, el abanico de formas que pueden representar se ve limitado. Gracias a la

introducción de unas funciones base de carácter racional, dichas curvas son capaces de representar un gran número de formas, entre otras las cónicas. Sin embargo, las nuevas capacidades tienen un coste en complejidad: además de unas funciones base más complicadas, se introducen una serie de escalares, llamadas pesos, que alteran el comportamiento de la forma de la curva (existe un peso por cada punto de control). En el artículo *Two Simulated Annealing Optimization Schemas for Rational Bézier Curve Fitting in the Presence of Noise*, publicado en la revista *Mathematical Problems in Engineering*, se propone el uso de curvas racionales de Bézier para el ajuste de nubes de puntos ruidosos. La propuesta consiste en adaptar la metodología usada en el caso no racional para la computación de los pesos, presentándose así dos esquemas. El primer esquema, denominado *all-in-one*, consiste en buscar todos los parámetros de los que depende el modelo, a través de una única instancia del Simulated Annealing. Por otra parte, el segundo esquema, denominado *sequential*, consiste en la búsqueda de cada tipo de variable por separado, usando distintas instancias del Simulated Annealing que toman como entrada la salida del paso anterior. Dadas las nuevas dificultades, se hizo necesario adaptar de nuevo el Simulated Annealing. En primer lugar, la estrategia computacionalmente costosa de *re-inicialización*, fue sustituida por una heurística para el computo de la temperatura inicial, adaptada de [Ben-Ameur, 2004]. Por primera vez en nuestro trabajo, y hasta donde conocen los autores, por primera vez en el ajuste de curvas y superficies, se propone la utilización de dos conjuntos de temperaturas, método basado en el Adaptive Simulated Annealing [Ingber, 1993a]. Por una parte, la *temperatura de aceptación* trata de mantener un equilibrio sobre los candidatos que sobreviven a una iteración, mientras las *temperaturas de generación* tratan (una por cada variable libre del sistema), de adaptarse a las fluctuaciones de energía del sistema, tratando de explotar direcciones potencialmente buenas, mediante la modificación de la función de distribución usada en la generación de nuevos candidatos. La implementación se prueba con éxito en varios experimentos: se mejoran los resultados obtenidos en [Loucera et al., 2014], a la par que se evalúa exitosamente sobre un nuevo conjunto de ejemplos.

Superficies racionales de Bézier

La adaptación de nuestra metodología al caso de superficies racionales de Bézier, se realiza en el artículo *Simulated Annealing and Natural Neighbor for Rational Bézier Surface Reconstruction from Scattered Data Points*, publicado en *Harmony Search Algorithm: Proceedings of the 3rd International Conference on Harmony Search Algorithm* (ICHSA 2017). En dicha adaptación se trata por primera vez el problema de las nubes de puntos desorganizadas. Para su ajuste mediante superficies racionales de Bézier, nuestra metodología requirió de, entro otros, dos grandes ajustes: por una parte, la generación de entornos se realiza mediante una distribución de Cauchy que se adapta a la temperatura de generación. En segundo lugar, en el caso de datos desordenados, se dota a la nube de puntos de una topología de conectividad, mediante la construcción de una superficie base, a través de *la interpolación mediante entornos naturales* [Sibson, 1981]. La metodología se valida mediante una serie de ejemplos, tanto académicos, como reales, con muy buenos resultados.

Ajuste mediante B-spline

El capítulo 7 está dedicado al ajuste de curvas y superficies mediante B-spline, tanto racionales como no racionales. Estos nuevos modelos introducen, aún en el escenario más sencillo, una serie de dificultades, a través de la inclusión de los llamados nodos: puntos de rotura en el espacio paramétrico que permiten un control local sobre la forma final del modelo. El incremento de su complejidad, los nodos añaden una capa extra de no-linealidad, viene acompañado de una mayor flexibilidad en cuanto a formas reproducibles. Cabe señalar que todo modelo de Bézier es a su vez una B-spline, pero el reverso no es cierto. Además, gracias a lo que se denominan *nodos* múltiples (nodos internos exactamente iguales) se pueden reproducir formas con puntos cúspide (perdida de la diferenciabilidad), *codos*, saltos de continuidad etc.

Ajuste de curvas explícitas mediante B-splines de nodos libres

La sección de ajuste mediante B-spline comienza con un problema de gran interés, desde el punto de vista de la optimización, especialmente en el campo de la regresión no lineal: el ajuste de curvas explícitas. Es decir, suponiendo que la forma subyacente se puede representar como una función explícita, tratar de buscar la B-spline óptima que ajusta los datos. De nuevo, nuestra metodología transforma el problema geométrico, en uno de optimización, mediante el ajuste de la suma del residuo de los mínimos cuadrados. En este caso, los principales ajustes en nuestra metodología se han realizado en el apartado del Simulated Annealing. Nuestra propuesta consiste en sustituir las búsquedas locales mediante algoritmos clásicos por una serie de heurísticas basadas en la búsqueda de patrones directos [Hedar and Fukushima, 2004], completamente hibridados con la generación de entornos mediante distribuciones de Cauchy. Para asegurar la viabilidad de nuestra renovada metodología, se han realizado numerosos experimentos con una batería de ejemplos recogidos en la literatura (que aseguran que todas las dificultades geométricas están recogidas). Los resultados son alentadores en cuanto a su calidad, siendo capaces de reconstruir auténticos nodos múltiples, y están a la par de otros métodos bio-inspirados. Sin embargo, la gran cantidad de parámetros de los que depende el método, teniendo que ser ajustados para cada tipo de problema, supone una desventaja con respecto a dichos métodos.

Ajuste de curvas paramétricas mediante B-spline

Para resolver el caso del ajuste de curvas mediante B-spline paramétricas, nuestra propuesta consiste en ampliar nuestra metodología con una variante mimética, de diseño propio, del Simulated Annealing (MeSA). Dicha variante aparece por primera vez en el artículo *Memetic Simulated Annealing for Data Approximation with Local-Support Curves* que será publicado en los *Procedia of Computer Science* y presentado en *International Conference on Computational Science 2017*. Nuestra variante mimética utiliza una serie de procesos, integrados dentro del propio algoritmo, para *aprender* ciertos parámetros (las temperaturas de aceptación y generación) así como una búsqueda local que trata de aproximar las fluctuaciones de energía del siste-

ma mediante aproximaciones lineales, basado en el método *constrained optimization by linear approximations* (COBYLA) [Powell, 1994]. Por otra parte, la generación de nuevos candidatos se basa en una adaptación de la ley μ^{-1} de teoría de las comunicaciones. La metodología se valida contra una serie de ejemplos extraídos de la literatura con excelentes resultados. Cabe señalar que ninguno de los ejemplos pertenece al mundo del diseño asistido por computador, por tanto, no son curvas *apropiadas* para su optimización mediante splines. Sin embargo, los resultados son excelentes desde un punto de vista de gráficos por computador.

Ajuste mediante superficies NURBS

Es, sin duda, el caso más difícil: ajuste de nubes de puntos, mediante el uso de superficies racionales B-spline (NURBS). A la ya consabida dificultad de aproximar funciones racionales y un nuevo juego de pesos, hay que añadir la dificultad de un conjunto de nodos independientes por cada dirección paramétrica.

Uno de los principales problemas derivado del uso de técnicas de *búsqueda directa*, como COBYLA, reside en lo que se conoce como *la maldición de las muchas dimensiones* (the curse of dimensionality): a medida que el número de variables crece, las heurísticas de búsqueda local necesitan de un gran número de iteraciones para poder mejorar la solución, a la par que la complejidad de cada ciclo aumenta sustancialmente. Para superar dichos obstáculos, nuestra propuesta consiste en sustituir el *aprendizaje local* mediante heurísticas, por una cadena de *vuelos de Lévy* que sean capaces de capturar las fluctuaciones no-lineales de la energía del sistema. Los resultados obtenidos al aplicar esta nueva metodología sobre una nube de puntos extraída de la evaluación de una NURBS son excelentes, se obtiene con una precisión del orden de 1e-14 en los parámetros del modelo subyacente.

En esa ocasión, además, se ha resuelto el problema *real*: una nube de puntos, desorganizada, de tamaño masivo (unos 400 mil puntos), procedente de un escáner 3D. Se trata de una nube de puntos obtenida al escanear la frente de una talla de madera de una Virgen, de importante valor patrimonial para la ciudad de Santander. Para resolver este problema, se aplican una serie de procesos previos al ajuste, consistentes en la aplicación de un filtrado de ruido, mediante el uso de k-vecinos más cercanos (KNN), para posteriormente aplicar una reducción de su tamaño, mediante el uso de técnicas de *voxelización*. Los resultados son muy prometedores y serán incluidos en un artículo futuro que cerrará esta fase de nuestra investigación.

Principales contribuciones

A continuación, se resumen las principales aportaciones de la Tesis al estado del conocimiento:

- Se ha elaborado una metodología general para la aproximar nubes de puntos ruidosos, mediante modelos de Bézier y B-spline. El método no requiere de ningún parámetro subjetivo, y es capaz de obtener el modelo óptimo de forma

automática. Nuestra metodología fusiona varias técnicas, desde la optimización por mínimos cuadrados, hasta la selección de modelos mediante métricas de ciencias de la información, pasando por el uso de meta-heurísticas.

- Para cada tipo de problema tratado, se ha adaptado una versión específica del Simulated Annealing.
- En el caso de datos desorganizados, se ha elaborado un método mediante el que dotar de una estructura de conectividad a la nube de puntos, usando métodos de interpolación basados en entornos naturales o de Sibson.
- Con respecto al Simulated Annealing, la Tesis concluye con la implementación de dos novedosas técnicas:
  - Un esquema de optimización mimético, mediante aproximaciones lineales capaces de *explotar* los entornos más prometedores.
  - Un esquema de optimización mimético, mediante la adaptación de las distribuciones de generación de entornos, a las fluctuaciones de la energía del sistema.

Futuras líneas de trabajo

A continuación, mostraremos una serie de líneas abiertas de investigación. En esencia podemos distinguir dos vertientes: las mejoras sobre el Simulated Annealing y los trabajos futuros para la reconstrucción de curvas y superficies.

Meta-heurísticos

La línea abierta con la inclusión de los vuelos de Lévy muestra un prometedor futuro, por lo que nuestra investigación se centrará en la mejora del aspecto mimético del MeSA. Para ello planteamos la creación de un *pool* de distribuciones, de tal manera que el algoritmo vaya eligiendo en cada momento qué distribución se adapta mejor a las fluctuaciones del espacio de la energía. A medida que el algoritmo avanza, cuando se detecte un atasco en el flujo de candidatos aceptados mediante heurísticas sobre la variancia de la energía, se procederá a realizar un vuelo de Lévy a partir de puntos ya obtenidos. De esta forma se puede escapar, tanto de las cuencas de atracción, como de las *mesetas energéticas*. En cambio, la generación de entornos estará basada en el FSA mediante la distribución de Cauchy, mientras que la explotación de las regiones prometedoras se realizará mediante búsquedas tipo down-hill basadas en la función μ^{-1}.

Resolución del problema completo

Con respecto a la ingeniería inversa de curvas y superficies, el primer trabajo en el horizonte consistirá en realizar más experimentos con nuestro esquema mimético para

el caso de superficies NURBS y comparar los resultados con los principales métodos desarrollados en la literatura. Una vez concluida esta tarea, se dará por finalizada la primera fase de nuestra investigación, para dar comienzo a la segunda fase: la resolución del problema completo, así como la mejora de la metodología.

El problema completo

El problema completo hace referencia al caso en el que la nube de puntos es de tamaño masivo, desorganizada y no puede ser representada por una única superficie. En este caso el problema se puede dividir en cuatro fases claramente diferenciadas, de las cuales ya hemos resuelto una en el presente trabajo.

1. *Segmentación*: Dado que la nube no se puede representar con una única superficie, la primera tarea consiste en dividirla (segmentarla) en regiones que sí se puedan representar con un único modelo.

2. *Detección de agujeros*. El método necesita detectar la presencia de agujeros, que pueden ser, principalmente, de dos tipos: aquellos intrínsecos a la superficie subyacente (p.ej. un toro) o bien aquellos que surgen como resultado de errores en los procesos de adquisición. Un problema importante de solución no trivial: si un agujero es rellanado con el modelo resultante, entonces no se está siendo fiel a la superficie subyacente. Por otra parte los agujeros debidos a datos perdidos pueden suponer todo un problema para la estabilidad de los algoritmos. Nuestra propuesta al respecto pasa por la utilización de métodos de aprendizaje no supervisado.

3. *Ajuste de superficies*. Aquí entra en juego el presente trabajo y sería la parte ya resuelta.

4. *Unión de segmentos*. Una vez se haya realizado el ajuste de una superficie para cada uno de los segmentos, es necesario volver a reunir cada uno de los modelos (proceso de *pegado*). Se puede entender como un problema multi-objetivo en que haya que satisfacer ciertas condiciones, en cada una de las fronteras de pegado: continuidad, resolución de la intersección de puntos comunes etc. A tal efecto, se está trabajando en un esquema mixto, en el que las restricciones sean manejadas mediante sistemas inmunes.

Splines de nodo libre mediante técnicas de un único paso

La metodología presentada en el presente trabajo emplea la optimización de un conjunto dado de modelos candidatos, en lo que comúnmente se denomina procesos en dos-pasos: en primer lugar se seleccionan una serie de modelos base, a continuación, para cada modelo se realiza una optimización de sus parámetros, para finalmente obtener una puntuación. Dicha puntuación es transformada a posteriori, mediante algún criterio que tenga en cuenta tanto la calidad de la reconstrucción, como su complejidad. Finalmente se elige el modelo con mejor puntuación transformada.

Nuestra propuesta contempla emplear una serie de meta-heurísticas que son capaces de mezclar información continua, con otra de carácter binario, usando una única función objetivo que condensa toda la información del problema. Para emplear dicho método, planteamos la creación de un problema dual en el que se doblan la cantidad de variables. Por cada nodo se crea una nueva variable ficticia binaria, que indica si dicho nodo se emplea o no. La función objetivo devolverá de forma automática el BIC, por lo que ya recogerá el balance entre complejidad y fidelidad.

Actualmente existen en la literatura varios métodos que emplean técnicas de un único paso, como por ejemplo: en [Yoshimoto et al., 2003], la propuesta mediante algoritmos genéticos ofrece buenos resultados, pero no consigue de manera automática auténticos nodos múltiples (los aproxima mediante los denominados quasi-múltiples nodos), mientras que en [Valenzuela et al., 2013] se emplea un algoritmo genético multi-objetivo que, aunque reconstruye la forma subyacente, es incapaz de devolver el número óptimo de nodos, produciendo numerosos nodos *inútiles*, por lo que las formas con discontinuidades o puntos cúspides quedan fuera de su alcance. Actualmente los autores están trabajando en la adaptación de los métodos de evolución evolutiva, mediante procesos de búsqueda desorganizada [Egea et al., 2009, Egea et al., 2010]. Los primeros resultados son esperanzadores, pues ya es posible la reconstrucción de todo tipo de funciones continuas, incluso aquellas en las que se pierde la diferenciabilidad gracias a que se encuentra el número óptimo de nodos (produciendo a la par nodos múltiples auténticos).

Acerca del autor

Cabe señalar, que el presente trabajo viene avalado por una serie de publicaciones en el campo de la reconstrucción de curvas y superficies [Loucera et al., 2014, Iglesias et al., 2015b, Iglesias et al., 2016b, Loucera et al., 2017b] y el que se publicará en breve, con nuestra propia variante mimética del Simulated Annealing [Loucera et al., 2017a], que nos ha proporcionado el esqueleto conductor de la Tesis. Además, otra serie de trabajos han sido realizados en paralelo a la Tesis, conectando bien con la computación bio-inspirada [Cosido et al., 2013] o bien con la reconstrucción de modelos matemáticos [Cosido et al., 2014]. Lejos del campo de los meta-heurísticos o la ingeniería inversa, el autor ha comenzado a colaborar en el campo de la ingeniería de comunicaciones con una publicaciÃşn en ciernes *Experimental Comparison of Non-Coherent SU-MIMO Schemes* cuyo autor principal es Fanjul, J..

Actualmente, el autor trabaja como investigador en el *Grupo de tratamiento Avanzado de la Señal* de la Universidad de Cantabria, donde desarrolla tareas en el área de reconocimiento de patrones en señales, mediante aprendizaje automático, bajo los proyectos financiados por el Ministerio de Economía, Industria y Competitividad (MINECO) TEC2013-47141-C4-R (RACHEL) y el proyecto industrial *Development of an Automatic System for Detection and Classification of Defects in Steam Generator Tubes* (AAUT). En el pasado, el autor ha trabajado en diversos puestos, dentro de proyectos de investigación y/o técnicos, de entre los que destacan:

- Agencia Estatal de Meteorología (AEMET). Proyecto 14: Apoyo de sistemas a la puesta a punto de modelos numéricos operativos y en desarrollo (Resolución de 3 de diciembre de 2014).
- Instituto de Estadística de Cantabria (ICANE): Técnico de software y modelización estadística.
- CODELSE, Proyecto SIeV: Propuesta, desarrollo e implementación de algoritmos meta-heurísticos con el fin de crear un Sistema Inteligente de Soporte a la Evacuación de Edificios en Situaciones de Emergencia. Uso de técnicas de aprendizaje automático en la predicción y clasificación de los niveles de alarma.

Notation and Acronyms

Used Notation

a	Scalar (lowercase)
$\mathbf{a}$	Column vector (lowercase boldface)
$\mathbf{S}$	Multi dimensional array (uppercase boldface)
$\mathbf{a}[i]$	access element i of vector $\mathbf{a}$
$\|\cdot\|_p$	p-norm
$\|\cdot\|$	cardinal
$\mathcal{C}^k(I)$	Functions with first k continuous derivatives (calligraphic)
$\mathcal{P}^k$	Space of polynomials of degree k (calligraphic)
$.^T$	Matrix transpose
$.^\dagger$	Moore-Penrose pseudo-inverse
$\mathrm{vec}(\cdot)$	Column-wise vectorization
$\odot$	Element-wise multiplication of two vectors
$\ni$	Sampled from
$\mathbb{R}$	Set fo real numbers
$\mathbb{N}$	Set of natural numbers
$\mathcal{N}(\mu, \sigma)$	Normal distribution with norm μ and variance σ
$\mathcal{U}([a, b])$	Uniform distribution over $[a, b]$

Acronyms

AIC	Akaike Information Criterion
ASA	Adaptive Simulated Annealing
BIC	Bayesian Information Criterion
CSA	Classic Simulated Annealing
FSA	Fast Simulated Annealing
GSA	General Simulated Annealing
MSE	Mean Square Error
NMSE	Normalized Mean Square Error
RMSE	Root Mean Square Error
RSS	Residual Sum of Squares
SA	Simulated Annealing
NURBS	Nobody Understands Rational B-Splines

Contents

Part I

Introduction and Background

Chapter 1
Introduction

An author never does more
damage to his readers than
when he hides a difficulty

Evariste Galois

In this chapter we discuss the issue of data fitting with curves and surfaces in reverse engineering, a key component in the manufacturing processes of today. The chapter provides an overview of the rest of this work, starting with an outline of the reverse engineering problem and concluding with a brief overview of the goals and main contributions of this Thesis.

1.1 Reverse engineering

Digital modeling has become one of the core tools in the design and manufacturing industry. As the technologies to manufacture advance (e.g. 3D home printing), the capability to produce physical objects from computer models has become trivial. On the other hand, the ability to reconstruct a digital model from a given real world object continues to be a very difficult problem. This process, known as *reverse engineering*, plays a fundamental role in the manufacturing industries of today. This is specially true in key areas of the current industrial infrastructure such as the automobile, aerospace, and shipbuilding sectors [Pottmann et al., 2005].

In this work, we focus on the issue of data fitting with curves and surfaces in reverse engineering. The problem starts with a point cloud, usually of massive size, that typically contains high quantities of noise due to measurement errors and other artifacts produced by the acquisition method [Pauly et al., 2002]. In the general scenario, the topology and geometry of the data is not provided, so the sample points connectivity is of unknown nature. The main objective behind the reverse engineering procedure is the construction of a model that captures the underlying information of the point cloud (shape, topology, geometry and so on).

The curve and surface reconstruction problem arises in several scientific fields. For instance, spline fitting is an important tool in machine learning, as it forms the core of many regression techniques [Marsh and Cormier, 2001], approximation

theory [Rice and Saloin, 1969, Cox, 1990] and CAGD [Dierckx, 1995]. Beyond the mathematics field, curve/surface-fitting plays an essential role in many other fields of expertise such as noninvasive techniques in the medical and health areas, where a typical problem involves the creation of a surface from a set of cross-sections [Bajaj et al., 1995, Park and Kim, 1996], acquired by computer tomography or other techniques such as magnetic resonance.

In climatology and meteorology thin-plate splines are often used to model some parameter [Tait et al., 2006], usually by incorporating surface elevation constraints. In general, in most scenarios where kriging is used [Cressie, 1990], such as surface temperature and elevation models, a spline interpolation/regression model could be used [Hutchinson and Gessler, 1994].

As it has been hinted before, the most common scenario for data fitting is an unordered, massive and noisy point cloud [Pauly et al., 2002] usually obtained by making use of scanner devices [Hoppe et al., 1992, Eck and Hoppe, 1996, Kazhdan and Hoppe, 2013] which often introduce noise and other artifacts due to measurement errors and mechanical deficiencies.

To construct the underlying model there exist various approaches that differ not only on the inner workings of the method but in the resulting solution. The most common approaches to fit a model to a point cloud can be divided into three categories: approximation by polygonal meshes, constructive solid-geometry modeling and curve and surface mathematical modeling. Our methodology falls within the latter approach; we aim to reconstruct the underlying shape of the data by means of either Bézier or B-spline curves and surfaces. In chapter 3 we review some of the most important curve and surface reconstruction methods in the literature.

In this work we provide a general methodology to fit, in an automatic way, a spline model to the point cloud. This is done by transforming the geometrical problem into a non-linear multivariate continuous optimization problem, known to be non-convex and multi-modal. The first research techniques addressing the data fitting issue with free-form parametric curves and surfaces appeared during the sixties and seventies and were focused on the use of classical numerical algorithms [de Boor and Rice, 1968, Rice and Saloin, 1969, Powell, 1970, Jupp, 1978]. Although many scenarios were successfully solved, the general problem was beyond the capabilities of traditional mathematical tools. During the next two decades various approaches appeared, such as line use error bounds [Park, 2004], approximations to dominant points [Park and Lee, 2007] and many other methods, usually based on iterative procedures that start from a rough *baseline* model which is latter refined by a set of heuristics [Ma and Kruth, 1995, Piegl and Tiller, 2001, Brujic et al., 2011]. These methods tend to fail with noisy data or require a set of constraints which are hard to enforce or meet in the general case. The last few years have seen the rise of popularity of bio-inspired optimization [Yang, 2010, Yang, 2014], which in the case of curve and surface fitting have been used with great success [Iglesias and Gálvez, 2016].

Figure 1.1: A 3D scanned point cloud of a Spanish Virgin wooden statue, kindly provided by the company 3DINTELLIGENCE.

1.2 Why splines?

Spline models are nowadays ubiquitous in the CAD/CAM industry and the computer graphics field of research. These free-form parametric models consist of a linear combination over a set of functions, a basis in the functional space, which have a series of properties specially well suited for computer graphics. The coefficients of the combination, usually referred as the poles or control points, can be used to directly modify the shape of the curve or surface. Its widespread acceptance and popularity from the eighties onwards is due to the following (no-exhaustive) list of properties:

I *Information comprehension.* A canonic mathematical form can store high quantities of information about the shape with only *relatively* few parameters.

II *Shape construction.* The set of shapes and forms that can be reproduced are limitless and includes the all classic models: conics, ruled and revolution surfaces etc. Thus, they provided a unified mathematical model to represent both classical and free-form shapes.

III *Easy of manipulation.* Due to the mathematical properties of the blending functions (the basis) the final shape can be locally manipulated by modifying the control points (a must in interactive design). Furthermore, the geometrical interpretations of the model are so clearly codified that a spline is very well suited to geometric design.

IV *Invariance.* The shapes are invariant under affine transformations (scaling, rotation, translation) and some projective transformations. Even more, the transformation could be applied to the spline poles and the result would be the same shape as to applying it to the whole model.

V *Computationally stable.* There are very fast and stable algorithms to evaluate, modify and interrogate a spline model.

1.3 Stochastic-driven optimization

As it has been mentioned before, the free-form curve and surface fitting optimization problem is a really difficult one. To overcome the limitations of traditional mathematical methods when dealing with the problem, the scientific community has shifted its attention towards other approaches: from artificial intelligence to nature-inspired computation.

A diverse set of techniques belonging to the artificial intelligence paradigm have been applied to the problem with varying grades of success. For instance, in [Gu and Yan, 1995, Hoffmann and Varady, 1998, Barhak and Fischer, 2001] artificial neural networks are used to approximate scattered data with free-form surfaces, whereas in [Kumar et al., 2004, Hoffmann, 2005] self-organizing maps are used to the same effect. These artificial intelligence approaches share one key advantage, the ability to *learn* the point cloud topology in a *natural* way. However, they present many limitations from a model construction point of view, the main one being the inability to infer the correct functional structure of the problem. Furthermore, the initial network topology must be provided beforehand, a difficult problem-dependent task in itself. These limitations are partially surpassed with the introduction of functional networks as in the case of [Iglesias et al., 2004] with B-spline surface reconstruction or [Gálvez et al., 2007] where genetic algorithms are combined with functional networks for curve and surface fitting (using Bézier models). The functional approach is further extended in [Iglesias and Gálvez, 2008] by the introduction of rational B-spline functions which can reconstruct the functional structure of a rational B-spline model for curve fitting.

Nature-inspired optimization is a very promising area of research based on the idea that Nature by itself is able to solve problems in very efficient ways, which leads to the computational imitation of such processes. Most of these optimization algorithms are driven by stochastic meta-heuristics backed by powerful metaphors of the natural processes mimicked. For instance, at the core of our methodology lies the Simulated Annealing algorithm [Kirkpatrick et al., 1983], a meta-heuristic that mimics the thermodynamics behind the annealing of a metal. Other stochastic-driven algorithms have been successfully applied to curve and surface fitting. What follows is a brief overview of some of the most popular.

Evolutionary algorithms. The family of population-based meta-heuristics known as evolutionary algorithms (EA) try to mimic the evolutionary mechanisms of Nature,

such as mutation, reproduction, selection, adaptation etc. Genetic Algorithms (GA) are one of the most developed subsets of EA, the meta-heuristic behind a GA tries to model the process of natural selection. Genetic algorithms have been successfully applied to B-spline curve fitting [Yoshimoto et al., 1999, Yoshimoto et al., 2003]. In [Valenzuela et al., 2013] a multi-objective variant is proposed for the same problem. Other types of EA haven been also used to spline fitting, most notably artificial immune systems and clonal selection algorithms. AIS-CSA are based on the way the immune system is able to adapt certain cells to counter antigen invaders. Free-knot spline fitting is provided in [Ülker and Arslan, 2009] with the use of AIS whereas in [Iglesias et al., 2013, Gálvez et al., 2015] CSA is used to fit Bézier and free-knot splines.

Swarm intelligence. Swarm intelligence deals with natural and artificial systems that are comprised of individuals with the ability to cooperate towards a common goal by sharing information about the problem being solved. Particle swarm optimization is a stochastic-driven algorithm for global optimization which updates a set of candidate solutions (the *swarm*) through a certain set of iterations by sharing space-related information among them. PSO is used in [Gálvez and Iglesias, 2011] for free-knot spline fitting and in [Gálvez and Iglesias, 2012] for point cloud approximation with NURBS surfaces. Other swarm intelligence techniques applied in data fitting mimic the patterns of biological systems, such as the firefly behavior [Gálvez and Iglesias, 2013] or bat swarms [Iglesias et al., 2015a].

Physics based. This is a much broader field than the previous ones, as it encompasses all the optimization algorithms which mimic a certain physical process. In [Gálvez and Iglesias, 2013] the electromagnetism algorithm (EMA) is used for Bézier curve fitting. The EMA method is based on the attraction-repulsion mechanism to move sample points towards the optimum. This algorithm is further improved in [Iglesias and Gálvez, 2016] with the addition of a memetic approach (a local learning phase) for rational Bézier surface reconstruction. There are other physics-based algorithms such as Big Bang-Big Crunch Algorithm (BB-BC) [Erol and Eksin, 2006] which is inspired by the entropy fluctuations formulated in the homonymous theory dealing with the evolution of the universe, and the Gravitational Search Algorithm [Rashedi et al., 2009] which mimics the theory by the same name. The Simulated Annealing method used in this work, also belongs to this category of algorithms as it is backed by a thermodynamics metaphor. SA has been used to solve a specific set of curve and surface fitting problems, such as: mesh triangulation [Sen and Zheng, 1992] and free-knot spline fitting [Valenzuela and Pasadas, 2010].

1.4 Goal of the Thesis

The aim of this thesis is to construct a general methodology for point cloud fitting by means of spline models. To construct such a methodology we provide a set of Simu-

lated Annealing variants which draws from the literature on meta-heuristic methods to build optimization techniques tailored to the problem at hand. The motivation of this work lies in the ability of the Simulated Annealing, as other meta-heuristics, to adapt itself to the problem without the need of subjective decisions, even in the case of incomplete or noisy information (a must when dealing with point clouds).

First, we present the problem of data fitting and the mathematical background needed for its understanding. Although the canonical definition in mathematical terms is clear and spline models are a very powerful tool for shape-driven tasks, several difficulties arise from the problem itself (non-linear, multi-modal) and the complexity of the models.

To deal with the problem difficulties we provide one specialized SA algorithm for each kind of spline model: rational and non-rational curve and surface Bézier/ B-spline fitting models. To prevent over-fitting problems we make use of information sciences criteria which provide a trade-off between data-fidelity and model complexity.

Next, we provide a set of extensions to the methodology to treat unorganized point clouds of massive size. Thus, making our solution real-world ready. To show the validity of our techniques we provide a set of experimental results, mostly taken from an academic background. However, we conclude with the application to a real-world scenario by fitting a piece of a 3D-scanned Virgin.

Figure 1.2 provides a general outline of the method with the chapter where each technique can be found.

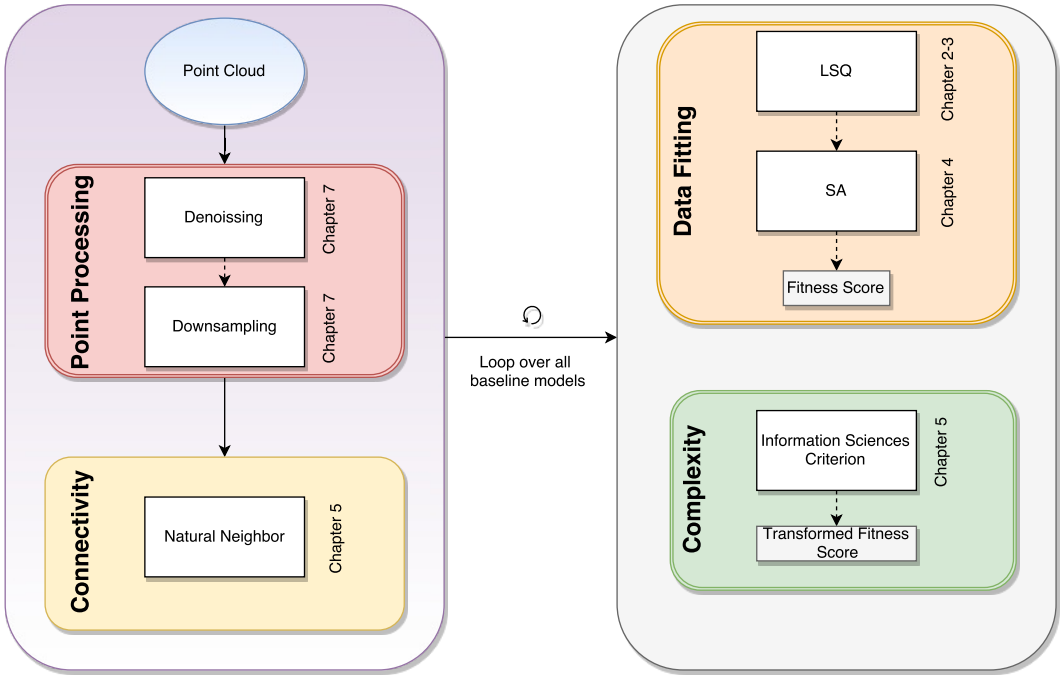


Figure 1.2: Outline of the methodology.

1.5 Thesis overview

Although all chapters are closely interlinked within a reverse-engineering framework, the Thesis is divided into four blocks of information. The introduction and background correspond to the first part. We continue with an outline of the optimization methodology, while the third part deals with the presentation of some reverse engineering applications of our data fitting framework. Finally, we conclude with a chapter devoted to discuss the conclusions derived from our work and some hints about future lines of research.

Part I presents an introduction to reverse engineering in CAD/CAM and the mathematical background needed to understand the problem.

Chapter 1. The current chapter. We provide the motivation behind reverse engineering and a general overview of the problem and the solutions found in the literature.

Chapter 2. This chapter explains the mathematical models used to fit the underlying shape of the point cloud. We describe the general concept of a spline and then we provide definitions for the specific families used in this work, with an explanation of the most relevant characteristics.

Chapter 3. In this chapter we present the data fitting problem in its most general form: to fit a NURBS surface. We continue explaining the main difficulties that arise from the given formulation and the relevant literature on the approximations taken to deal with it.

In part II we present the core of our methodology: the Simulated Annealing and the model selection framework.

Chapter 4. The classical Simulated Annealing is presented in this chapter, along with the relevant literature on the subject. We explain each component and provide the main proposals that have influenced our work.

Chapter 5. The full methodology is revealed in this chapter, once all the inner workings have been presented in the receding chapters.

Different reverse engineering applications of our work are explained in detail in part III.

Chapter 6. In this chapter we include the reverse engineering applications of our methodology for the case of Bézier curves and surfaces.

Chapter 7. This chapter provides the results for the application of our methodology to different reverse engineering fitting scenarios: free-knot splines, parametric B-Spline curves and NURBS.

In part IV we summarize the conclusions and contributions of this Thesis and provide a general outline of our future lines of research. Beyond chapter 8 we include the appendices and bibliographic references.

1.6 About the author

This Thesis is built on top of the results that have lead to the publications of various research articles in the reverse engineering field [Loucera et al., 2014, Iglesias et al., 2015b, Iglesias et al., 2016b, Loucera et al., 2017b] and a paper where we introduce our own memetic variant of the Simulated Annealing for B-spline fitting [Loucera et al., 2017a]. In addition to these works the author has also collaborated in the publication of [Cosido et al., 2013] where a meta-heuristic multi-objective graph resolution technique based on the ant colony algorithm was used to route selection. In [Cosido et al., 2014] reverse engineering models are used to reconstruct an historical building. Beyond the meta-heuristic field, the author has begun to collaborate in other areas such as communications where he has already collaborated in *Experimental Comparison of Non-Coherent SU-MIMO Schemes* a soon to be published paper by Jacobo Fanjul and others.

Nowadays the author has a full-time job as a machine learning researcher in the *Advanced Signal Processing Group* (GTAS, in Spanish) under projects supported by the Ministerio de Economía, Industria y Competitividad of Spain (MINECO) TEC2013-47141-C4-R (RACHEL) and the industrial project *Development of an Automatic System for Detection and Classification of Defects in Steam Generator Tubes* (AAUT). Other related jobs by the author include:

- State Agency of Meteorology (AEMET). Project 14: Numerical models support, research and development (2014 December resolution).
- Cantabria Statistics Institute (ICANE): Software engineering and statistical model building
- CODELSE, SIeV Porject: Research and development on meta-heuristic algorithms for near real-time route computation in fire-escape scenarios. Machine learning research on how to predict a fire-alarm level.

Chapter 2
Mathematical Background

Attend to your Configuration.

Edwin A. Abbott, Flatland

In this chapter we will discuss the main properties of the mathematical models at the core of our methodology: the spline space of functions.

Definition 2.1. *Let* $\{u_i\}_{i=0}^{n+1}$ *a partition of the interval* $I = [\alpha, \beta] \subset \mathbb{R}$ *such that* $\alpha = u_0 < \ldots < u_{n+1} = \beta$. *The functional* $f : I \to \mathbb{R}$ *is a polynomial spline of degree* $k \in \mathbb{N}$ *if:*

- $f \in \mathcal{C}^{k-1}(I)$
- $f|_{[u_i, u_{i+1}]} \in \mathcal{P}_k \quad \forall i$

The interior points u_i are usually referenced as the spline breakpoints or the knot vector. As different conditions are imposed on the polynomials and their derivatives on the breakpoints, different families of splines are formulated. In this paper we are interested in the Bézier and B-spline (basic splines) families, the de-facto standard for computer graphics.

We begin our discussion of spline curves and surfaces by looking at the Bézier family. Although in this work we make use of the most current formulation of each curve and surface presented, these parametric curves were developed by Pierre Bézier [Bézier, 1974] by taking into account geometrical considerations of the design problems he was working on, automobile body description and aircraft wing design, at Renault. Later on, Forrest [Forrest, 1972] demonstrated the equivalence between the Bézier formulation and the Bernstein basis approximation function (used in this work).

The geometric considerations considered by Bézier play an important role, as the resulting curves and surfaces must adhere to design problems where a trade-off between aesthetic and functional requirements must be fulfilled. The rest of the parametric splines presented in this work could be seen as extensions to the Bézier family with the premise of an increase in its design capabilities while maintaining the functional requirements and the geometrical *beautifulness*. For an historical perspective of how each spline family came to be, see [Rogers, 2000].

2.1 Bézier curves

A Bézier curve of degree k in $\mathbb{R}^d$, with control points $\{\mathbf{P_i}\}_{i=0}^{k}$, is a free-form parametric curve defined as follows:

$$\mathbf{b} = \sum_{i=0}^{k} B_{i,k}(u)\mathbf{P}_i \quad \text{with } u \in [0,1] \tag{2.1}$$

where $B_{i,k}$ is the i-th Bernstein polynomial of degree k defined by:

$$B_{i,k}(u) = \binom{k}{i} u^i (1-u)^{k-i} \tag{2.2}$$

2.1.1 Bernstein basis properties

The Bernstein basis of polynomial functions has several geometric and analytical properties which play a very important role in the computer graphics field, such as:

Non negativity. $B_{i,k}(u) \geq 0 \quad \forall u \in [0,1]\,, i \in 0,k.$

Partition of unity. $\sum_{i=0}^{k} B_{i,k}(u) = 1 \quad \forall u \in [0,1].$

Symmetry. $B_{i,k}(t) = B_{k-i,k}(1-t).$

Linear precision. The polynomial u can be expressed as a linear combination of the Bernstein basis: $u = \sum_{i=0}^{k} \frac{i}{k} B_{i,k}(u)$.

Differentiability. The derivative of a Bernstein polynomial can be expressed as a linear combination of the basis: $B'_{i,k}(u) = k\left(B_{i-1,k-1}(u) - B_{i,k-1}(u)\right)$.

From a computationally point of view, there are two properties that facilitate both the algorithm implementation and its numerical stability. From now on we take for granted that $B_{0,0}(u) = 1$ and $B_{i,k} = 0$ if $i < 0$ or $i > k$.

Recursion. The Bernstein polynomial of degree k can be expressed as a combination of those of the preceding degree:

$$B_{i,k}(u) = (1-u)B_{i,k-1}(u) + uB_{i-1,k-1}(u) \tag{2.3}$$

Degree elevation. The Bernstein polynomial of degree k can be expressed as a combination of those of degree $k+1$:

$$B_{i,k}(u) = \left(1 - \frac{i}{n+1}B_{i,k+1}(u) + \frac{i+1}{k+1}B_{i+1,k+1}(u)\right) \tag{2.4}$$

For a more rigorous and deep treatment of the Bernstein basis, beyond the computer graphics field, see [Lorentz, 2012].

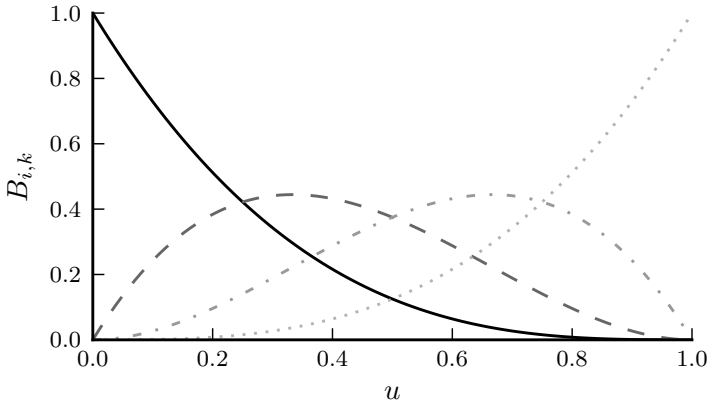


Figure 2.1: Cubic Bézier basis functions.

2.1.2 Curve properties

Bézier curves have the following properties, which form the backbone of every major digital modeling tool.

Transformation invariance. To apply an affine transformation to a Bézier curve it is only required to apply it to its control points. Furthermore, the shape of the curve is preserved.

End point interpolation. The curve passes through the last and first control points.

Convex hull. Bézier curves are contained within the convex hull of its control points.

Variation diminishing property. Let λ be the number of times a given $d-1$ linear variety H embedded in $\mathbb{R}^d$ intersects the control polygon of the curve, then $\mathbf{b}$ has at most λ intersections with the variety H.

Differentiability. The derivative of a Bézier curve $\mathbf{b}$ is another Bézier curve of degree $k-1$ and can be expressed as:

$$\mathbf{b}'(u) = \sum_{i=0}^{k-1} B_{i,k}(u)(k\mathbf{P}_i)$$

End point differentiability. The j-th derivative of a Bézier curve at an extreme point depends on the derivative of $\mathbf{b}$ at the extreme point and its j consecutive control points.

Although in this work the last property does not play an essential role, as we try to reconstruct a point cloud with a single curve, it is very important when joining Bézier curves or computing the normals of a given surface. See Figure 2.2 for a visual representation of how the `\Omega` symbol is internally represented by means of a Bézier *path*, a series of interconnected Bézier curves, in the `Times New Roman` font. See section A.1 for the code needed to generate the glyph.

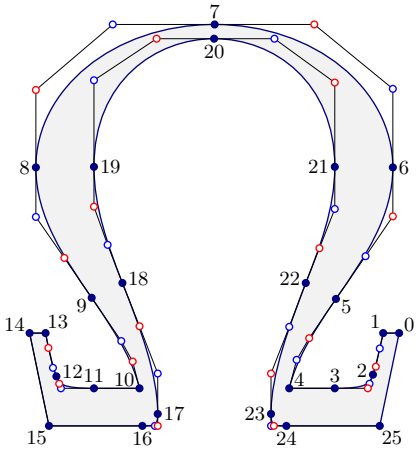


Figure 2.2: Cubic Bézier basis functions.

2.1.3 de Casteljau Algorithm.

From the recursion formula given by 2.3, a Bézier curve of degree k with control points $\{\mathbf{P}_i\}_{i=0}^{k}$, denoted as $\mathbf{b}_k\left(\{\mathbf{P}_i\}\right)$, can be written as:

$$\mathbf{b}_k\left(\{\mathbf{P}_i\}_{i=0}^{k}\right) = (1-u)\mathbf{b}_{k-1}\left(\{\mathbf{P}_i\}_{i=0}^{k-1}\right) + u\mathbf{b}_{k-1}\left(\{\mathbf{P}_i\}_{i=0}^{k}\right) \tag{2.5}$$

which is, in fact, a procedure to recursively compute a point of a Bézier curve. If we fix $u = u_0$ and rename $\mathbf{P}_i$ as $\mathbf{P}_{0,i}$, then follows:

$$\mathbf{P}_{l,i}\left(u_0\right) = \left(1-u_0\right)\mathbf{P}_{l-1,i}\left(u_0\right) + u_0\mathbf{P}_{l-1,i+1}\left(u_0\right) \tag{2.6}$$

Equation (2.6) is known as the *de Casteljau Algorithm*: a geometric process to compute a point of the Bézier curve. Note that this is a purely geometrical way of computing the points of a Bézier curve without even needing the Bernstein polynomials. In fact, the curve can be drawn with a pencil and a ruler by following the geometrical interpretation. For the interested reader we recommend the lecture of [Rogers, 2000].

2.2 Rational Bézier curves

A rational Bézier curve of degree n in $\mathbb{R}^d$, with control points $\mathbf{P}_i$, is a parametric free-form curve represented by:

$$\mathbf{b} = \sum_{i=0}^{k} R_{i,n}(u)\mathbf{P}_i \quad \text{with } u \in [0,1] \tag{2.7}$$

were $R_{i,n}$ are the rational extension of the Bernstein basis given by:

$$R_{i,n} = \frac{w_i B_{i,n}(u)}{\sum_{k=0}^{n} w_k B_{k,n}(u)} \quad \text{with } u, v \in [0,1] \tag{2.8}$$

2.2.1 Curve and basis properties

Given that the basis functions of a rational Bézier curve are an extension of the non-rational ones, there are some analytical and geometrical properties inherited from the non-rational case, such as:

Projective invariance. The shape of a rational surface is preserved through projective transformations. Furthermore if the transformation is applied to the control points, the resulting curve is exactly the same as if the transformation is directly applied to equation (2.7).

End points interpolation. The curve passes through the extreme points of its control polygon.

Convex hull. The curve is fully contained within the convex hull of its control points.

2.2.2 Motivation

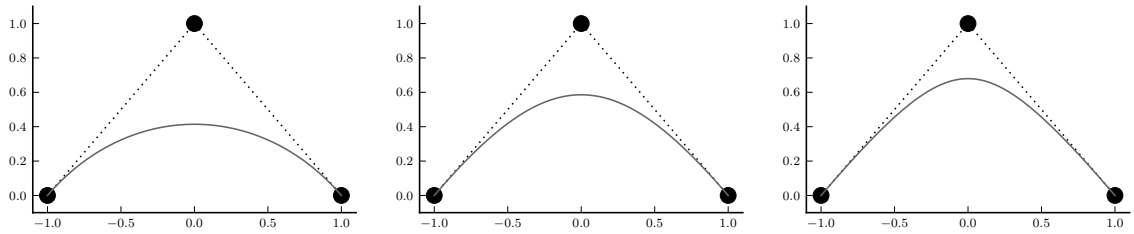
Non-rational Bézier curves, as other polynomial-based constructions, are unable to represent some classical figures, such as conics. However, rational Bézier curves can be used to represent those forms (Theorem 2.2), thus providing a significant trade-off between the added complexity and the flexibility gained. Furthermore, equation

(2.1) can be retrieved from equation (2.7) by taking all weights equal to 1, so a non-rational Bézier curve can be expressed as a rational one.

Theorem 2.2. *Any non-degerative rational Bézier curve defined by three non-collinear control points* $\{\mathbf{P}_i\}_{i=0}^2$ *and weights* $\{w_i\}_{i=0}^2$*, represents a conic section that is in correspondence with the sign of* $d = w_0 w_2 - w_1$ *as follows:*

- *An ellipse if* $d > 0$.
- *A parabola if* $d = 0$.
- *A hyperbola if* $d < 0$.

Figure 2.3 summarizes the findings of Theorem 2.2, with the control points and the rational Bézier curve represented as dots and a continuous line, respectively.



(a) Ellipse $d < 0$. **(b)** Parabola $d = 0$. **(c)** Hyperbola $d < 0$.

Figure 2.3: Conics construction by means of rational Bézier curves.

2.3 Bézier surfaces

A polynomial Bézier surface of degree (m, n) is a tensor-product parametric patch with the following formula:

$$\mathbf{S}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_{i,m}(u) B_{j,n}(v) \mathbf{P}_{ij}, \quad \textbf{with } u, v \in [0, 1] \tag{2.9}$$

where $B_{i,m}(u)$ and $B_{j,n}(v)$ are the Bernstein basis functions (2.2) in the u, v parametric space. The coefficients $\mathbf{P}_{i,j} \in \mathbb{R}^d$ are the control net, which roughly determine the shape of the surface.

2.3.1 Surface properties

Given that the basis functions are those of the Bézier curves, many properties are inherited. From a computer graphics point of view the most important are:

End point interpolation. The surface passes through the corner points of its control net.

Affine invariance. In order to apply an affine transformation to a Bézier surface one can apply the transformation to the control net. The resulting Bézier surface formed by the transformed control points is the same as if applied to the whole equation (2.9).

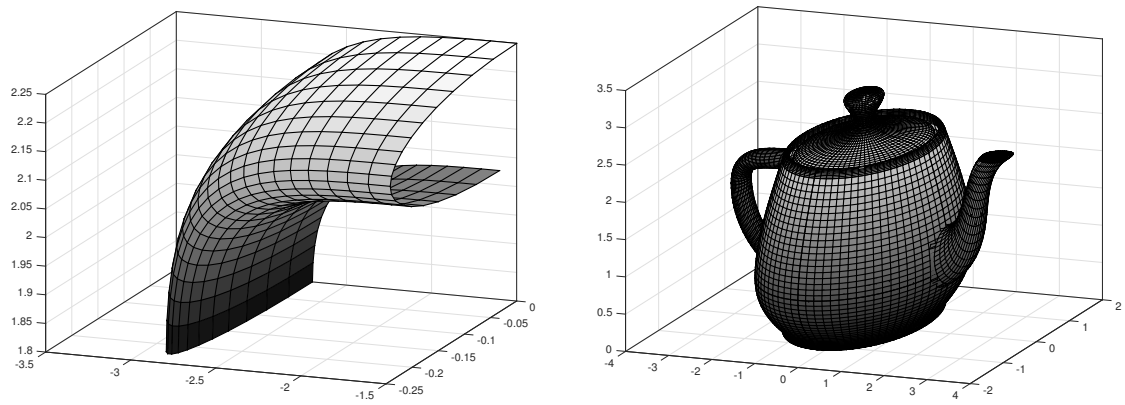
Non negativity. The basis functions are non-negative everywhere.

Partition of unity. $\sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,n}(u)B_{j,m}(v) = 1$

Convex hull. The surface lies in the convex hull of its control net.

Variation Diminishing. The variation-diminishing property for tensor surfaces is both undefined and unknown, [Prautzsch and Gallagher, 1992].

Figure 2.4 shows the *Utah Teapot* [Blinn and Newell, 1976]: without a doubt one of the most famous models in the history of computer graphics, and one of the first cases of a *hand-crafted* reverse-engineered digital model of a real world object [Crow, 1987].



(a) A patch from the Utah Teapot. **(b)** The complete Utah Teapot.

Figure 2.4: The Utah Teapot.

2.4 Rational Bézier surfaces

A rational Bézier surface of degree (m, n) can be formulated as:

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} R_{i,j}(u,v)\mathbf{P}_{i,j} \tag{2.10}$$

where $\mathbf{P}_{i,j}$ are the control points and $R_{i,j}(u,v)$ are the blending or basis functions given by:

$$R_{i,j} = \frac{w_{i,j} B_{i,m}(u) B_{j,n}(v)}{\sum_{k=0}^{m} \sum_{l=0}^{n} w_{k,l} B_{k,m}(u) B_{l,n}(v)} \quad \text{with } u, v \in [0,1] \tag{2.11}$$

2.4.1 Surface properties

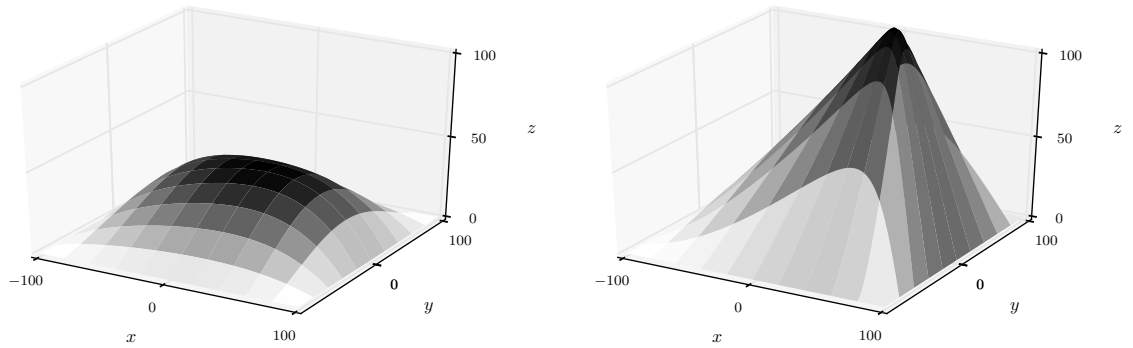
Some of the properties for non-rational patches can be easily generalized directly to the rational case:

Projective invariance. The shape of a rational surface is preserved through projective transformations. Furthermore, if the transformation is applied to the control net, the resulting surface is exactly the same as results from applying the transformation directly to equation (2.10).

End point interpolation. The surface passes through the corner points of its control net.

Convex hull. The surface is contained within the convex hull of its control net.

Shape parameters. If a weight is increased, relative to its neighbors, the surface is pushed towards the respective control point. For a visual example of the dramatic effect of this property see Figure 2.5.



(a) Non rational example. **(b)** One weight increased.

Figure 2.5: Effect of increasing a weight in a rational Bézier patch.

2.4.2 Motivation

Non rational Bézier patches are unable to represent some classic surfaces, such as the sphere, which are ubiquitous in the CAD/CAM industry. This is not the case for rational Bézier patches as it could be seen in [Piegl, 1986], where a formal construction of the sphere via bivariate rational surfaces is presented.

2.5 B-spline curves

A parametric B-spline curve $\mathbf{s} \subset \mathbb{R}^d$ of degree k is a piecewise function that can be expressed as:

$$\mathbf{s}(u) = \sum_{i=0}^{n} N_{i,k}(u)\mathbf{P}_i \tag{2.12}$$

where $u \in [\alpha, \beta]$ represents the data parameters, $\{\mathbf{P}_i\}$ the control net of the curve and $\{N_i\}$ are the so called B-spline basis functions.

The blending functions of degree k on $[u_i, u_{i+1}]$ and breakpoints $u_0 = \alpha <= u_1 <= \ldots <= u_m = \beta$, can be computed via the Cox de-Boor recursion formula [de Boor, 1978]:

$$N_{i,k}(u) = \varphi^{+}_{i,k}(u)N_{i,k-1}(u) + \varphi^{-}_{i,k}(u)N_{i+1,k-1}(u) \tag{2.13}$$

where $\varphi^{+}_{i,k}(u) = \frac{u-u_i}{u_{i+k-1}-u_i}$ and $\varphi^{-}_{i,k}(u) = \frac{u_{i+k}-u}{u_{i+k}-u_{i+1}}$ and $N_{i,0}(u)$ is the unit function with support on $[u_i, u_{i+1})$. In this work we only use splines clamped at the edges, so $u_0 = \cdots = u_k = \alpha$ and $u_{m-k} = \cdots = u_m = \beta$. Note that each blending function is a local support function, so perturbations on a given interval do not affect the global shape of the curve, a must for the CAD/CAM industry. It is agreed that $0/0 = 0$ for each basic function.

The degree, knots and poles are bound together by the formula:

$$m = n + k + 1 \tag{2.14}$$

Note that we have relaxed the knot vector condition of definition 2.1 to allow knots with multiplicity higher than one. Although this change may introduce some stability issues, as we will see in the next chapter, it allows the B-spline to reproduce more complicated shapes such as curves with cusps and discontinuities.

2.5.1 Basic functions

The basic functions $N_{i,k}(u)$ possess a series of fundamental properties that can be summarized as follows:

Non-negativity. $N_{i,k}(u) \geq 0, \quad \forall i, k, u.$

Partition of unity. $\sum_{i=0}^{n}(u) = 1, \quad \forall u \in [u_0, u_m]$

Local support. $N_{i,k}(u) = 0, \quad \forall u \notin [u_0, y_m].$

Knot span support. For each knot span $[u_i, u_{i+1}]$ at most $k+1$ blending functions are non-zero.

Differentiability. $N_{i,k}(u) \in \mathcal{C}^{\infty}((u_i, u_{i+1}))$ for each knot span interior. Let $p \in \mathbb{N}$ be the multiplicity of knot u_l, then $N_{i,k}(u) \in \mathcal{C}^{k-p}(u_l)$.

Global maximum. Each basic function, with $k > 0$, $N_{i,k}(u)$ has exactly one maximum value.

For these and other fundamental properties of the spline space of functions see [Piegl and Tiller, 1995].

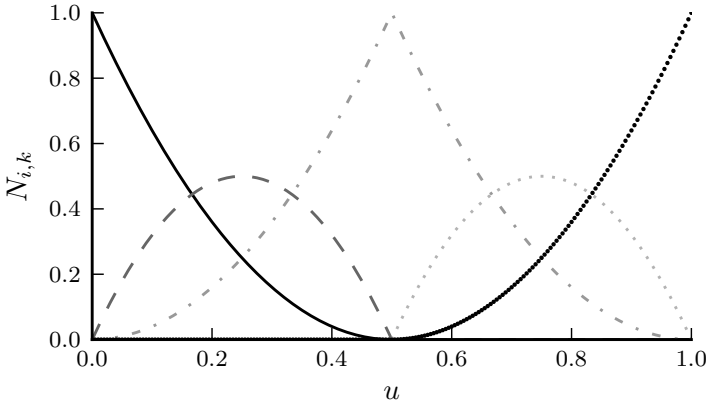


Figure 2.6: B-spline basis functions.

2.5.2 Curve properties

A B-spline curve of degree k as defined per equation (2.12) has a set of geometric properties of fundamental importance in the CAD/CAM field.

Affine invariance. An affine transformation can be reconstructed from the affine images of the B-spline control points.

End point interpolation. A clamped spline interpolates the control points at the extremes.

Strong convex hull property. A B-spline curve is contained in the convex hull of its control net. Furthermore, given a point u in a non-degenerated knot span $[u_l, u_{l+1})$, the curve point $\mathbf{s}(u)$ lies in the convex hull of the sequence $\{\mathbf{P}_{l-k}, \ldots, \mathbf{P}_l\}$.

Local shape parameters. From the local support properties of the blending functions it follows that a given control point can only affect at most $k+1$ knot spans.

Differentiability. A B-spline is infinitely differentiable at every point which is not a knot. Let $p_l \in \mathbb{N}$ be the multiplicity of the knot u_l, then $\mathbf{s} \in \mathcal{C}^{k-p_l}(u_l)$.

Variation diminishing property. Let λ be the number of times a given linear variety H of dimension $d-1$ in $\mathbb{R}^d$ intersects the control polygon of the curve, then $\mathbf{s}$ has at most λ intersections with the variety H. See [Lane and Riesenfeld, 1983] for a proof of this interesting property.

2.5.3 The Bézier sub-family

A Bézier curve is a special case of a B-spline. Let $\mathbf{s}$ be a clamped B-spline curve of degree k with control points P_i and knot vector $\boldsymbol{\tau}$ (with no interior knots), then the basic functions of $\mathbf{s}$ can be reduced to:

$$N_{i,k}(u) = \binom{k}{i} u^i (1-u)^{k-i}$$

which are in fact the Bernstein polynomials (2.2). So equation (2.12) gets transformed into (2.1) as the blending functions are exactly the same.

2.5.4 Algorithms

In this section we will introduce the main algorithms used to compute a B-spline curve point. For a full discussion of these and other computer graphics algorithms see [Piegl and Tiller, 2012]. Another major resource for spline computing is the `FITPACK` routines [Dierckx, 1995].

Lets assume that we want to compute the values of the basis functions and their derivatives at a point t which lies in the knot span $[u_i, u_{i+1})$. The first step is to find the span index i via the `get_span_index` routine. In order to simplify the notation lets set the span index to $j \leftarrow m-k-1$, thus t lies now in the span $[u_{m-k-1}, u_{m-k})$.

Once we know that t lies in the span i by means of Algorithm 2.1 we compute all the non-zero B-spline basis functions on the interval defined by the span index i as per Algorithm 2.2.

2.6 B-spline surfaces

A parametric B-spline surface of degree (k,l) is a tensor-product patch which can be expressed as:

$$\mathbf{S}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,k}(u) N_{j,l}(v) \mathbf{P}_{ij} \tag{2.15}$$

where $\mathbf{P}_{i,j}$ is the control net of the surface and $N_{i,k}(u)B_{j,l}(v)$ are the B-spline basis functions of degree k,l with knot vectors $\mathcal{U} = u_0 = \alpha_u <= u_1 <= \ldots <= u_{m_u} = \beta_u$ and $\mathcal{V} = v_0 = \alpha_v <= v_1 <= \ldots <= v_{m_v} = \beta_v$, as previously defined in equation (2.13).

Algorithm 2.1: `get_span_index`

Input: span index j, degree k, evaluation point t, knot vector $\boldsymbol{\tau}$
Output: knot span index i
if $t \neq \boldsymbol{\tau}[j+1]$ **then**
  $left \leftarrow k$
  $right \leftarrow j+1$
  $i \leftarrow {}^{(left+right)}/_{2}$
  while $t < \boldsymbol{\tau}[i]$ $||$ $t \geq \boldsymbol{\tau}[i+1]$ **do**
    if $t < \boldsymbol{\tau}[i]$ **then**
      $left \leftarrow i$
    else
      $left \leftarrow i$
    end
    $i \leftarrow {}^{(left+right)}/_{2}$
  end
else
  $i \leftarrow j$
end
return i

Algorithm 2.2: `bspline_basis_fun`

Input: point t, knot span i, degree k, knot vector $\boldsymbol{\tau}$
Output: Vector $\mathbf{N}$ of non-zero basis functions evaluated at t
$\mathbf{N} \leftarrow 1$
for $\ell = 1$ **to** k **do**
  $left[\ell] \leftarrow t - \boldsymbol{\tau}[i+1-\ell]$
  $right[\ell] \leftarrow -t + \boldsymbol{\tau}[i+\ell]$
  $val \leftarrow 0$
  for $r = 0$ **to** $r < \ell$ **do**
    $var \leftarrow \frac{\mathbf{N}[r]}{right[r+1]+left[\ell-r]}$
    $\mathbf{N}[r] \leftarrow val + right[r+1] \cdot val$
    $val \leftarrow left[\ell-r] \cdot var$
  end
  $\mathbf{N}[\ell] \leftarrow val$
end
return $\mathbf{N}$

2.6.1 Surface properties

Because the B-spline basis forms the blending functions of the surface, several properties are inherited from the curve definition:

Corner point interpolation. The surface passes through the corner points of its control net.

Affine invariance. The surface is invariant with respect to affine transformations, the shape of the surface is unaltered. Furthermore, it is the same to apply an affine transformation to either the whole surface, as per equation (2.15), or its control net.

Local control. The influence of a single control point is limited to $k+1, l+1$ knot spans in each parametric direction.

Differentiability. The surface is $\mathcal{C}^{k-1}$ and $\mathcal{C}^{l-1}$ in each parametric direction, u and v respectively.

Convex hull. The surface lies within the convex hull of the control net formed by the union of all the convex hulls of k, l neighboring control points.

Variation-diminishing. As with the rational Bézier surface the variation-diminishing property is currently unknown for B-spline surfaces, [Prautzsch and Gallagher, 1992].

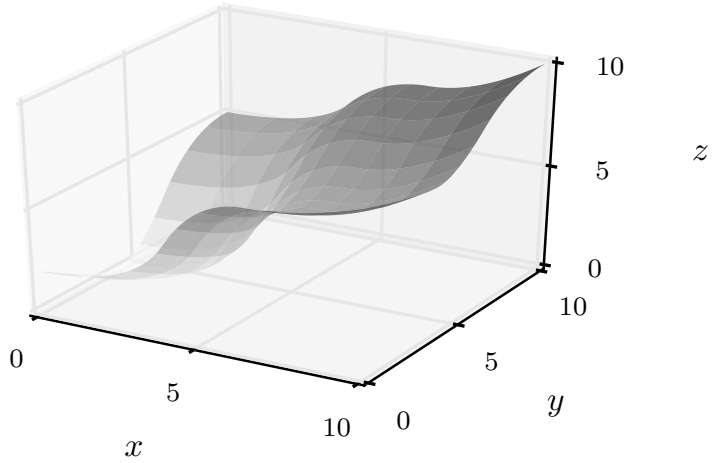


Figure 2.7: A bi-cubic B-spline surface.

2.7 Rational B-splines

The classical definition of a rational B-spline is to consider the projection of a standard polynomial, non-rational, B-spline curve in $\mathbb{R}^{d+1}$ with poles $\mathbf{P}_i^h$ back into $\mathbb{R}^d$.

Definition 2.3. *Let* $\mathbf{s}(u) = \sum_{i=0}^{n} N_{i,k}(u)\mathbf{P}_i^h$ *be a B-spline curve od degree* k *in* $\mathbb{R}^{d+1}$ *defined as per equation* (2.12)*, so* $\mathbf{P}_i^h \in \mathbb{R}^{d+1}$*. The projection to* $\mathbb{R}^d$ *yields the curve:*

$$\mathbf{s} = \frac{\sum_{i=0}^{n} N_{i,k}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,k}(u) w_i}$$

which is a rational curve, known as rational B-spline.

Thus, the basis functions of a rational B-spline defined over the knot vector $\mathcal{U} = \{u_0 = \alpha <= u_1 <= \ldots <= u_m = \beta\}$ and weights $\{w_i\}_{i=0}^{n}$ are:

$$R_{i,k} = \frac{w_i N_{i,k}(u)}{\sum_{i=0}^{n} N_{i,k}(u) w_i} \tag{2.16}$$

2.7.1 Curve properties

Given that the rational B-spline basis and curves are a generalization of the non-rational B-spline ones, they share several essential properties, such as:

Non-negativity. $R_{i,k} \geq 0$.

Partition of unity. $\sum_{i=0}^{n} R_{i,k} = 1$.

Global maximum. $R_{i,k}$ has exactly one maximum for all $k > 0$.

Variation-diminishing. A rational B-spline curve has the variation-diminishing property as explained for the non-rational B-splines.

Convex hull. The curve lies within the union of convex hulls formed by $k+1$ consecutive control points.

Projective invariant. The shape of a rational B-spline curve is preserved through projective transformations. Furthermore, to apply a projective transformation one has only to apply it to its control polygon.

2.7.2 Motivation

The rational B-splines are a superset of the B-splines. To retrieve the non-rational formulation from definition 2.3 it is sufficient to take all the weights equal to 1. The rational B-spline is an industry standard because every previously defined spline family can be written as one by varying the knot and weight vectors. Furthermore, a rational B-spline provides a higher range of shapes to reproduce as more local shape parameters are introduced into its formulation.

2.8 Rational B-spline surfaces

A rational B-spline surface, known as NURBS (Non-Uniform Rational B-splines), can be constructed in a similar way as in the case of non-rational B-spline surfaces with a similar set of properties. In the following chapter we provide a more concise treatment of NURBS models.

$$\mathbf{S}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} R_{i,j}(u,v)\,\mathbf{P}_{i,j} \tag{2.17}$$

where $R_{i,j}(u,v)$ are the rational B-spline basis functions defined as follows:

$$R_{i,j}(u,v) = \frac{w_{i,j} N_{i,k}(u)\, N_{j,l}(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} N_{i,k}(u)\, N_{j,l}(v)} \tag{2.18}$$

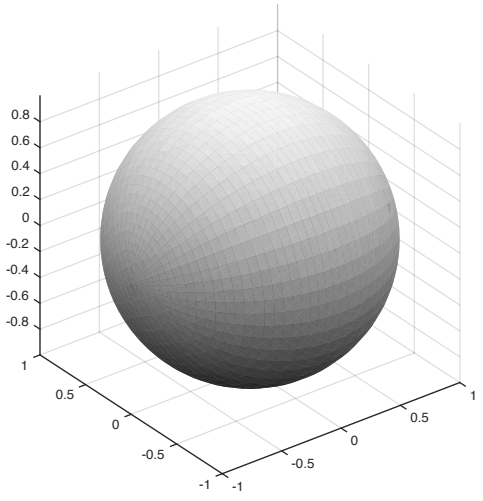


Figure 2.8: The sphere as a NURBS.

Chapter 3
The Problem

A problem well put is half solved.

John Dewey

Let $\mathcal{P}$ be a point cloud in $\mathbb{R}^d$, the aim of this work is to reconstruct a parametric curve, or surface, that fits the data by taking into account both, the quality of the approximation and the complexity of the model. In this chapter we will focus on the data-fidelity aspect of the reconstruction, and later on we will introduce our methodology for selecting a model which has a good compromise between complexity, aesthetics and goodness of fit. Since a Bézier curve (surface) can be seen as a special case of a B-spline curve (surface) we are going to present the problem in its most generic and complex form: to fit a rational B-spline surface.

3.1 Problem statement

Let $\left\{\mathbf{Q}_{p,q}\right\}_{p=0,q=0}^{N,M}$ be a set of points in $\mathbb{R}^d$, the problem consists of finding a rational B-spline surface that approximates the given data, by taking into account both the fidelity of the reconstruction and its complexity. From now on we refer to rational B-spline surfaces and curves, or its blending functions, with the acronym NURBS: Non-Uniform Rational B-splines.

In order to reconstruct the underlying shape of the data with a rational B-spline $\mathbf{S}$ of degree k, l, our method must perform the parametrization, i.e. find the parameters $\left\{u_p, v_l\right\}$ associated with the original data, compute the poles $\mathbf{P}_{i,j}$ and its weights $w_{i,j}$ with the corresponding breakpoints $(\tau_0, \ldots, \tau_{m_u})$ and $(\zeta_0, \ldots, \zeta_{m_v})$ and, finally, the method must deal with the model complexity: how to minimize the number of free parameters of the system. As a result, given a rational B-spline $\mathbf{S}$ surface defined as per equation (2.17), the fitting problem can be written as the least-squares minimization of:

$$E = \sum_{p=0}^{N}\sum_{q=0}^{M} \left\|\mathbf{Q}_{p,q} - \mathbf{S}\left(u_p, v_q\right)\right\|_2^2 = \sum_{p=0}^{N}\sum_{q=0}^{M} \left\|\mathbf{Q}_{p,q} - \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j} N_{i,k}\left(u\right) N_{j,l}\left(v\right)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j} N_{i,k}\left(u\right) N_{j,l}\left(v\right)}\right\|_2^2 \quad (3.1)$$

where $\|\cdot\|_2$ is the L^2 norm in $\mathbb{R}^d$. In many real world problems neither the order and degree, nor the parameterization and knots, nor the weight and poles of the surface are known. On these terms, the minimization of the residual sum of squares given by (3.1) becomes a highly non-linear, continuous and multivariate problem. A more concise presentation of these facts follows:

Multivariate. The unknowns of the problem are: the degree of the surface, $d(n+1)(m+1)$ variables from the poles $\{\mathbf{P}_{i,j}\}_{i,j} \in \mathbb{R}^d$, $(n+1)(m+1)$ weights as $\{w_i\}_{i,j} \in \mathbb{R}^+$, $\mu_\tau + \mu_\zeta$ internal breakpoints due to the imposed constraints on the boundary knots and finally the $M+N+2$ parameters. Let η be number of free variables of the system, then :

$$\eta = 2 + (d+1)(n+1)(m+1) + \mu_\tau + \mu_\zeta + M + N + 2 \tag{3.2}$$

Non-linear. A direct consequence of the basis expression (2.18).

Continuous and multi-modal. If we assume a known set of weights and poles the knot vector computation has been proved to be a non-convex and multi-modal optimization problem [de Boor, 1978, Laurent-Gengoux and Mekhilef, 1993].

Problem linearization. Note however, that if the knots, weights and parameters are known, the control net computation becomes an over-constrained linear system. This fact plays an essential role in our methodology as we will see in Chapter 5: on the one hand, the breakpoints, weights and parameterization are computed by means of a variant implementation of the Simulated Annealing optimization algorithm whereas the poles are a direct result of solving the linear system. On the other hand, the spline degree computation is done via the Bayesian Information Criterion BIC [Schwarz, 1978].

Scattered data. In the case of unorganized data we search for a projection through Principal Component Analysis (PCA) where the point cloud can be written in the $(\mathbf{u},\mathbf{v}) \mapsto \mathbf{S}(\mathbf{u},\mathbf{v})$ form. Finally, a structure is given to the point cloud by means of the Natural Neighbor Interpolant [Sibson, 1981]. Therefore, we have actually transformed the scattered data problem into the least square problem discussed in the current chapter.

Solution. The least squares fitting approximation of equation (3.1) can be rewritten as:

$$\mathbf{Q} = \mathfrak{R} \cdot \mathbf{P} \tag{3.3}$$

where $\mathbf{Q} = \text{vec}\left(\left\{\mathbf{Q}_{p,q}\right\}_{p,q}\right)$, $\mathbf{P} = \text{vec}\left(\left\{\mathbf{P}_{i,j}\right\}_{i,j}\right)$ and $\mathfrak{R}$ is constructed as the following matrix stacking:

$$\mathfrak{R} = \left\{\text{vec}\left(\left(\mathbf{R}\left(u_p, v_q\right)\right)^T\right)\right\}_{p,q} \quad \text{with} \quad \mathbf{R}(u,v) = \left\{\text{vec}\left(\left(R_{i,j}\left(u,v\right)\right)^T\right)\right\}_{i,j}$$

Note that $\mathbf{Q}$ is a vector of size $(N+1)(M+1)$ whereas $\mathbf{P}$ has length $(n+1)(m+1)$, so the system (3.3) is overdetermined as in most real world applications the inequality $(N+M) \gg (n+m)$ holds. Pre-multiplication on both sides of the system by the transpose of $\mathfrak{R}$ leads to:

$$\mathfrak{R}^T \cdot \mathbf{Q} = \mathfrak{R}^T \cdot \mathfrak{R} \cdot \mathbf{P}$$

which can be solved by classical least-squares methods. Through this work we have opted for the SVD decomposition, by means of the Moore-Penrose pseudo-inverse, of $\mathfrak{R}$. As we will see in the literature review that follows, the SVD decomposition is the preferred method when paired with the minimization of the Residual Sum of Squares, as other methods try to find the solution with most zeros, which is not appropriate in the case of surface fitting. So the Moore-Penrose solution for equation (3.3) is:

$$\mathbf{P} = \mathfrak{R}^\dagger \cdot \mathbf{Q} \tag{3.4}$$

3.2 Previous work

The creation of curves and surfaces, specially the B-spline family, from a set of measured points has been a central area of research in the Computer Aided Design and Manufacturing industry for several decades. The reconstruction of free-form models typically includes at least two phases: the parameterization phase, to search a suitable mapping between an artificial parametric domain and each measured point, and the fitting phase, where the model parameters are found, such as the knots, poles and weights in the NURBS case.

The rest of the current chapter is devoted to summarize the previous work in the field, by focusing on each aspect the reconstruction.

3.2.1 Fitting approximations

Although in this work we are focusing our literature review on the methods that use some sort of least-squares approximation, it is mandatory to note that, in general, traditional interpolation methodologies provide curves and surfaces with a very good parameterization and fitting errors with either a global [de Boor, 1978, Cox, 1990] or a series of local models [Akima, 1970, Renner, 1982]. These approaches are particularly useful in the manufacturing industry, as most pieces are composed of combinations of conics. However, when the point cloud is large or noisy, as is the case

when using modern data acquisition techniques, the interpolation methods tend to produce erratic models that do not capture the underlying shape of the data.

Least-squares approximation has been the leading technique for curve and surface fitting since its inception in the late sixties through the seventies [Reinsch, 1967, Powell, 1970, Hayes and Halliday, 1974]. These first approaches were characterized by their direct application of classical numerical mathematics methods, which were found to provide very good approximations for a certain set of applications. However, the general case was not solved, since classical methods tend to fall trapped at local minima. Also, its automation was not easy as these methods required to supply a set of *a priori* subjective parameters, as the expected general shape, smoothness etc.

3.2.2 Knot allocation

The knots placement and number have a profound impact on the resulting model. For instance, the modification of a single knot has direct consequences on the shape of its neighborhood, whereas the correct choice of the number of knots is directly correlated with the quality of the global solution. On the one hand, if there are not enough breakpoints it is not possible to capture the underlying shape of the data. On the other hand, weird behaviors must be expected when unnaturally increasing the size of the knot vector, as the stability of the system decreases while the computation time increases. Thus, a careful approach must be taken when determining the length and location of the knots.

The first *successful* strategies for the knot allocation problem were proposed by deBoor, Rice, Jupp and Diercx during the seventies. Nowadays, more than forty years later, this techniques conform the core of some of the most successful software packages for spline interpolation and approximation, such as the open source Scipy *interpolation* module [Jones et al., 2001–] and the spline toolbox for MATLAB [Mathworks, 2001–]. These strategies where the knots are treated as free variables of the system, such as our proposed methodology, are commonly referred in the literature as *free-knot splines*.

These early attempts share a common procedure based on solving the non-linear squares system by separating the non-linear and linear aspects when searching for both, the poles and the knots. For instance, deBoor and Rice [de Boor and Rice, 1968] start with a rough approximation of the knot vector to later apply a variational heuristic to each breakpoint, one at a time, in a cleverly constructed interval.

The free-knot spline problem is comprehensively explored in [Jupp, 1978] which demonstrates that the fitness landscape is full of local minima, leading the search for the global optimum to many stationary points where the classical minimization algorithms get trapped. To overcome such difficulties, Jupp proposes a knot transformation functional that increases the likelihood of convergence to the global optimum. This method has two major drawbacks: it depends on a very good starting guess, a very hard problem in itself, and no mechanism is provided in order to select the correct number of knots.

In [Dierckx, 1981] the knot vector is found via an iterative process that starts with a set without interior knots, i.e. a Bézier curve, and at each iteration a knot is inserted with its location optimized by means of a non-linear process. Each subprocess is computationally expensive and subject to fall into the stationary points previously pointed by Jupp. The global process ends by checking a tolerance on the RMSE.

Another popular approach for breakpoint allocation, proposed by [Ma and Kruth, 1995] and later refined by [Piegl and Tiller, 2001, Brujic et al., 2011], consists in extracting the knot and weight vectors from a *baseline* curve/surface, a rough initial approximation of the final fitting model, in order to linearize the system (3.1). These methods overemphasize the role of the control net as a way to overcome the limitations of the knot location heuristic, producing high fidelity models at the cost of a massive set of poles.

A meta-heuristic intermission

Although the variety of shapes that free-knot splines can reproduce outshine those of other strategies, the inherent difficulties of the problem, as pointed out in [de Boor, 1978, Jupp, 1978, Laurent-Gengoux and Mekhilef, 1993], has put the problem on hold. However, recent advances in evolutionary, nature-inspired and swarm intelligence computation have made possible to perform data fitting through splines with free-knots: from genetic algorithms [Yoshimoto et al., 2003], particle swarm optimization [Gálvez and Iglesias, 2011, Gálvez and Iglesias, 2012], clonal selection [De Castro and Von Zuben, 2000, Gálvez et al., 2015] and many other soft-computing algorithms. For a full review see [Iglesias and Gálvez, 2016].

Note that none of these models claim to be the better tool for all possible data fitting scenarios, a direct consequence of the no free launch theorem [Wolpert and Macready, 1997]. The Simulated Annealing optimization algorithm is a popular meta-heuristic that has been used to solve very specific problems in the area of spline fitting, such as linear-spline approximations [Kreylos and Hamann, 2001] or free-knot cubic splines [Valenzuela and Pasadas, 2010]. Our aim is to develop a Simulated Annealing driven methodology that can solve the general case and improves on the specific problems already solved.

3.2.3 Data parameterization

The most common curve and surface parametrization methods are the uniform, chord length and centripetal models. The most simple of all those three methods is the uniform, as it just builds an isometry between each parametric direction and a line, so it does not take into account the distribution of the point cloud.

Cumulative chord length is a scalar approximation of the arc length of the curve (the isoparametric curves in the case of model of dimension greater than one). The impact of both methods on the resulting model was studied in [Epstein, 1976].

Finally, the centripetal model [Lee, 1989] also observes the distribution of the data while trying to capture the changes in the curvature. A general formula, that

encapsulates the three methods was also published in [Lee, 1989], as we will later see in Chapter 5.

Many variants of the above methods have been published, the self-explained averaging method [Piegl and Tiller, 1987] and a set of methodologies [Rogers and Fog, 1989, Sarkar and Menq, 1991a, Sarkar and Menq, 1991b] that start from one of the classical parametrization models to later optimize it by minimizing the deviations between points in the original data and their corresponding pairs in the parametric domain.

In [Ma and Kruth, 1995] the parameterization is done through the projection of the point cloud to a baseline model, a rough approximation of the final model created from the boundary of the measured data. Although this method can be used even with an unordered point cloud, the baseline model computation can be a very difficult process if the measured data is noisy and has a complicated boundary.

As in the knots case, the development of new stochastic methods has allowed the researchers to deal with the curve and surface reconstruction in new ways built on the shoulders of classical approaches. What follows is just a small sample of the recent advances in the reverse engineering field where this work belongs.

Neural networks [Gu and Yan, 1995, Hoffmann and Varady, 1998] have been applied to find the parameterization of reverse-engineered curves and surfaces with the additional ability of re-arranging the data in the case of an unstructured point cloud.

Self-Organizing Map neural networks (SOM) are used in [Barhak and Fischer, 2001] to construct a parameterization from a 2D plane projection of the point cloud. This method improves on the previous ones by the SOM ability to infer a better topology of the measured points. However, it suffers from the same dependence on a good-enough boundary.

The full parameterization is recovered by means of evolutionary algorithms, such as differential evolution [Hasegawa et al., 2013], artificial immune system [Iglesias et al., 2013] and many others.

3.2.4 Other reverse engineering models

In this work we have put our attention into the spline reconstruction from a CAD point of view, but no literature review would be complete without mentioning other types of reconstructions which, in fact, are very good models for its use in other fields.

A staple in surface reconstruction was achieved by Eck and Hoppe in [Eck and Hoppe, 1996] with a schema for constructing a network of B-spline patches which is later optimized by an adaptive refinement of the patch network. In [Bajaj et al., 1995] a model is obtained from cross-sectional images, a very important and difficult problem in the medical science field. A two-step method is presented in [Guo, 1997] where a baseline model is constructed from 3D α-shapes and then a compact surface is constructed from the baseline structure, providing a suitable model for engineering analysis. Finally, Simulated Annealing is used in

[Sen and Zheng, 1992] to find a near optimal triangulation mesh of a given point cloud, a perfect fit for computer graphics rendering.

Part II

The Methodology

Chapter 4
Simulated Annealing

> There is a deep and useful connection between statistical mechanics and multivariate optimization
>
> S. Kirkpatrick, Optimization by Simulated Annealing

One of the major trends in global optimization during the last few decades has been to build algorithms trying to mimic certain efficient optimization patterns observed in natural processes. As a result, a series of very powerful nature-inspired optimization algorithms (e.g. particle swarm optimization, genetic algorithms, or ant colony optimization) have been devised. Very often, they provide better solutions than previous traditional mathematical algorithms to several hard optimization problems. Although they are very diverse, all of them share two common features: to be inspired by real-world observation and to search for solutions in a stochastic way. Most of them are also derivative-free, meaning that they can be applied to problems where it is not possible to compute the derivatives of the objective function (or they are very expensive computationally). In this work we apply the Simulated Annealing optimization algorithm to the surface and curve reconstruction problem.

This chapter is organized as follows: we start with a brief summary of the history behind the technique and its thermodynamic roots. Afterwards, the Simulated Annealing algorithm is presented with a concise discussion of its main parts and convergence properties. We conclude with the different SA proposals used in this work.

4.1 Background and history

Simulated Annealing (SA), introduced by Kirkpatrick et al. in the seminal paper *Optimization by Simulated Annealing* [Kirkpatrick et al., 1983], is regarded as one of the foundational algorithms of the nature-inspired computation field. It was presented as a method to resolve a very complex combinatorial problem (optimizing thousands of variables) which arises in optimal design of computers. The algorithm showcased the deep connections between statistical mechanics and computational optimization.

The thermodynamics metaphor behind the algorithm consists in mimicking the annealing process of a metal, the cooling and freezing strategy applied to the material so that it adopts a low-energy, crystalline state. During the process, atoms tend to move to configurations that minimize the system energy even if during such migration certain configurations rise the system overall energy (when it stabilizes for a fixed temperature, we call it thermal equilibrium). Such moves are more prominent at the beginning of the process than at the end, when the particles loose thermal mobility in order to polish the system inner structure to finally produce a better metal. As a result, the metals become stronger and with better properties, specially if the process is conducted several consecutive times (known as re-annealing).

The original SA algorithm is an advanced interpretation of the Metropolis-Hastings sampling method [Metropolis et al., 1953] to generate sample states of a thermodynamic system, showing the deep connections between statistical mechanics and combinatorial optimization. Given an initial (usually random) state in the solution domain, the algorithm iteratively perturbs it. Whenever a better solution is found, the change is always accepted; otherwise, it is accepted only with a certain probability. This probability is higher at the beginning (mimicking what happens in the thermodynamic process at high temperatures) than at the end. In other words, this idea of slow cooling is translated as a slow decrease of the probability of accepting such worse solutions. So essentially the system evolves from a free exploration of the search space at initial stages to a stochastic hill-climbing at latter stages.

4.1.1 A family of meta-heuristics

What follows is a general overview of the evolution of the SA algorithm. Even if the original implementation was proposed in the field of combinatorics optimization, i.e. the minimization of a function which states are drawn from a discrete set, the adaptation over continuous sets was formulated shortly afterwards in [Vanderbilt and Louie, 1984, Bohachevsky et al., 1986].

Although a major milestone in optimization, the original algorithm presents one major problem: an extremely slow convergence due to a slow cooling schedule coupled with a rigid generation function. See [Bertsimas and Tsitsiklis, 1993, Ingber, 1993b] for a more detailed analysis on its convergence properties.

To overcome these limitations the research developments in Simulated Annealing during the last decades have been centered on improving the convergence rate and the performance by showcasing the deep connection between the cooling schedule and the neighborhood function.

The first proof of convergence for a Simulated Annealing algorithm [Geman and Geman, 1984] was based on a logarithmic cooling schedule, also very slow, coupled with Gaussian perturbations. Later, a very promising generation function was proposed in [Corana et al., 1987] which tries to maintain a consistent number of accepted solutions during the algorithm life by computing an adaptive step-size for each dimension.

A proof of convergence was proposed in [Szu and Hartley, 1987] for the Fast Simulated Annealing which consists in coupling a fast cooling schedule with a Cauchy distribution for generating new solutions. The Very Fast Simulated Annealing [Ingber, 1993b] was built upon the previous method and it provided a proof of convergence for an even faster schedule. The Adaptive Simulated Annealing (ASA) [Ingber, 1993a] is the result of combining the Very Fast Annealing with a re-annealing technique (rise the temperature when certain conditions are met) and a very sophisticated neighborhood function which dynamically adjust a custom temperature for each decision variable that inter-operates with an adaptive step, thus taking into account the different sensitivities of the variables. Due to a continuous technical improvement, the ASA is considered, even today, a state of the art optimization algorithm.

Another trend of research arises from hybridizing the Simulated Annealing with either local search methods or population based meta-heuristics. Two common pitfalls in implementing major simulated annealing approaches are parameter tuning and the inability to obtain the desired precision (in finite time), see [Salamon et al., 2002, Suman and Kumar, 2005] for a survey on the topic. One way to overcome these drawbacks consists in coupling well known search heuristics, which exploit promising directions on the fitness landscape usually overlooked by global optimizers [Osman and Kelly, 2012, Rios and Sahinidis, 2012], with a Simulated Annealing implementation that gets a good enough approximation in a relative short time.

4.2 The algorithm

According to [van Laarhoven and Aarts, 1987] the Simulated Annealing family of meta-heuristics is composed of three core functions and the stop criterion:

- The neighborhood generation function, $\mathfrak{N}$, which generates a state based on a previously generated state and takes into account how advanced the algorithm is.
- An acceptance function, $\mathfrak{A}$, which sets the transition probability from one state to another.
- An annealing schedule, $\mathfrak{S}$, governing the algorithm flow by reducing an artificial parameter which mimics the annealing temperature.
- The stop criterion sets the condition that ends the algorithm.

In this work we are interested in the global optimization (minimization) of real-valued functions over a continuous domain:

$$\min_{\mathbf{x}\in X} f(\mathbf{x}) \tag{4.1}$$

where $f : X \subset \mathbb{R}^d \to \mathbb{R}$ is referred as the energy or cost function. In this context, the Simulated Annealing algorithm can be summarized as follows: given an initial feasible solution $\mathbf{x}_0 \in X$ and the system temperature $T_0 \in \mathbb{R}^+$, at each iteration k, the next candidate $\mathbf{y}_{k+1}$ is sampled from a neighborhood distribution function $\mathfrak{N}$ which takes into account the current solution $\mathbf{x}_k$, its history and the system temperature T_k. The new state is either accepted or ejected in accordance to an acceptance function $\mathfrak{A}$ which resembles a probability measure. Then, the system temperature is updated or not by following the rules given by a cooling schedule $\mathfrak{S}$. This cycle is repeated until some stopping criterion is met.

Algorithm 4.1 presents a general expression of the Simulated Annealing, where no assumptions about its parts are made (cooling schedule, neighborhood function, acceptance and stop criteria). Note that each choice of $\mathfrak{N}, \mathfrak{S}, \mathfrak{A}$ and the stop criterion leads to a different *interpretation* of the algorithm. More often than not, this choice is done in order to take into account certain characteristics of the problem being solved. Thus, the previous wording *the Simulated Annealing family of algorithms* is, therefore, fully justified.

Algorithm 4.1: General Simulated Annealing

Input: An initial guess $\mathbf{x}_0$ and temperature T_0. Cost function f
Output: The final solution $\mathbf{x}$
Initialization, $k \leftarrow 0$
while *The stop criterion is not met* **do**
  $\mathbf{z}_k \leftarrow \{\mathbf{x}_k\}$
  Get a sample point from the neighborhood distribution:

$$\mathbf{y}_{k+1} \ni \mathfrak{N}(\cdot, \{\mathbf{z}_k, T_k\})$$

  Get a sample from the uniform distribution $p \ni \mathcal{U}([0,1])$
  Accept or reject the trial solution in accordance to the acceptance function:

$$\mathbf{x}_{k+1} \leftarrow \begin{cases} \mathbf{y}_{k+1} & \text{if } p \leq \mathfrak{A}(\mathbf{x}_k, \mathbf{y}_{k+1}, T_k) \\ x_k & \text{otherwise} \end{cases}$$

  Sampling retrieval, $\mathbf{z}_{k+1} \leftarrow \{\mathbf{z}_k\} \cup \{\mathbf{y}_{k+1}\}$
  Apply the cooling schedule, $T_{k+1} \leftarrow \mathfrak{S}(\{\mathbf{z}_{k+1}\}, T_k)$
  Iterate, $k \leftarrow k+1$
end
return $\mathbf{x}_k$

The rest of this section is devoted to explain the main components of the Simulated Annealing algorithm.

4.2.1 The acceptance criterion

A major difference between the Simulated Annealing algorithm and other stochastic optimizers is the ability to accept worse transitions, i.e. continue the search from a solution which is fitness-wise worse than the preceding one. The underlying metaphor behind this ability comes from the statistical thermodynamics field: the probability of an atom to exist at an energy state $\mathcal{E}$ for a given temperature T can be modeled by the Boltzmann distribution.

$$\mathfrak{P}(\mathcal{E}) = \exp\left(\frac{\mathcal{E}}{\kappa T}\right) \tag{4.2}$$

where κ is the Boltzmann constant. Note that the probability of an atom to exist at a high energy state is greater at strong temperatures than at lower temperatures.

The law governing the probability of accepting a given transition follows the modified Metropolis criterion [Metropolis et al., 1953]:

$$\mathfrak{A} \leftarrow \min\left\{1, \exp\left(-\frac{\Delta f_k}{T_k}\right)\right\} \tag{4.3}$$

which is a computational model of the previous Boltzmann law: on the one hand, a positive transition is always accepted. On the other hand, the probability of accepting worse solutions is slowly decreased as the algorithm progresses (lower temperatures).

From now on, if it is not otherwise specified, any reference to an acceptance function must be interpreted as the one given by equation (4.3) which is, by far, the most used one [Suman and Kumar, 2005].

There is a theoretical framework that supports such a decision. Let $\mathfrak{A}(x, y, T)$ an acceptance function that can be expressed as $\mathfrak{A}^*\left(\Delta f_{x,y}, T\right)$, i.e. the criterion only depends on the energy difference of the solutions at a given temperature. Then, if the following conditions are met:

- $\mathfrak{A}^*\left(\Delta f_{x,y}, T\right) > 0$ holds $\forall T > 0$ and $\Delta f_{x,y} > 0$
- $\mathfrak{A}^*(1, T)$ is a strictly increasing function over $\mathbb{R}^+$

it has been proved [Schuur, 1997] that there exists a strictly increasing function $g(T) : \mathbb{R}^+ \to \mathbb{R}^+$ such that:

$$\mathfrak{A} = \min\left\{1, \exp\left(-\frac{\Delta f_k}{g(T_k)}\right)\right\}$$

in other words, given the aforementioned thermodynamic-driven constraints all acceptance functions are equivalent to the modified Metropolis criterion.

4.2.2 The next candidate distribution and cooling schedule

As it has been noted before, the cooling schedule and neighbor function are so interlinked that decoupling both terms leads to an unintuitive read. Thus, we will explain the most relevant choices in the literature by looking into how $\mathfrak{S}$ and $\mathfrak{N}$ interact. Before we get into the details of each method, it helps to explain some generalities about the cooling schedule and neighbor function.

On the one hand, the next candidate distribution, also known as the neighborhood function or visiting distribution, handles the way a new trial solution is proposed. An efficient candidate generator function $\mathfrak{N}$ must try to adhere to two optimization expectations derived from the SA thermodynamical principles:

- *Energy fluctuations:* as the system is cooled, the neighborhood function must produce candidates in such a way that their energy is lower than that of a random sample of the solution space. In other terms, after a certain number of iterations Δf_k must be comparable to T_k by an order of magnitude or less.
- *Flatland and deep basin escape:* as a general rule, the system must allow for occasional *big jumps* in the solution space, to explore new dimensions of the fitness landscape, in order to avoid being trapped, for a long set of iterations, in basin attractors or energy plateaus.

On the other hand, the annealing or cooling schedule specifies when and how the temperature is updated. Ideally, the cooling rate must be slow enough to let the system reach a thermal equilibrium, the stabilization of the accepted solutions energy, after each change of temperature. However, the number of iterations spent at the inner loop should be completely dependent on the fitness landscape and the new temperature. Thus, a successful SA implementation must take into account the coupling of the annealing schedule and the new candidate distribution, as the exploration of the solution space is essential in determining how long it takes to reach for the thermal equilibrium at a given temperature.

We conclude the cooling and candidate generation schemes dissertation with a summary of the most important concepts behind the principal schemes found in the literature.

Boltzmann annealing. The first proof of convergence [Geman and Geman, 1984] was proposed for the Classical or Boltzmann Annealing which consists in the pairing of a logarithmic cooling schedule with a Gaussian candidate distribution. More specifically:

$$T_{k+1} \leftarrow \frac{T_0}{\log(1+k)} \qquad \text{with} \quad T_0 > 0 \tag{4.4}$$

$$\mathbf{y}_{k+1} \leftarrow \mathbf{x}_k + \boldsymbol{\Delta}_k \qquad \text{with} \quad \boldsymbol{\Delta}_k \ni \mathcal{N}^d\left(0, \sqrt{T_k}\right) \tag{4.5}$$

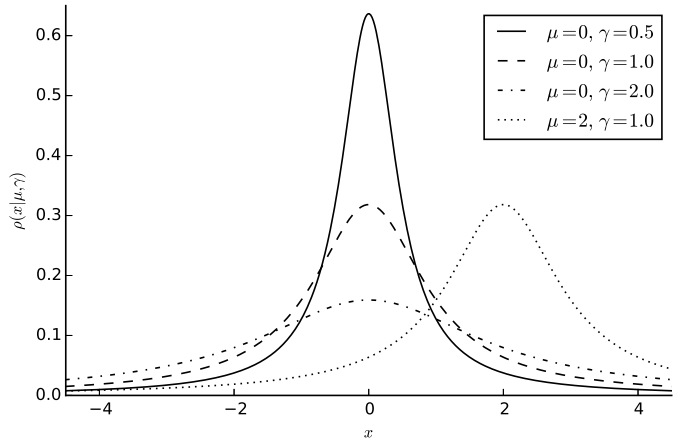


Figure 4.1: Cauchy distribution.

where the temperature is reduced every N_{inner} iterations for a total of N_{outer} iterations, which are problem-dependent. Note that the visiting distribution does not take into account the topology of the fitness landscape in the neighborhood of a given solution. To guarantee the full convergence of the algorithm it needs to spend thousands of iterations each temperature T_k and T_0 should be greater than the deepest local basin [Hajek, 1988]. Thus, although it is backed by a formal proof of convergence it may not happen in finite time.

Fast annealing. The Fast Simulated Annealing (FSA) [Szu and Hartley, 1987] consist in the pairing of an inverse logarithmic cooling rate with a D-dimensional Cauchy visiting distribution which allows for occasional long jumps, see Figure 4.1. It has been proven that the cooling schedule could be improved up to $T_{k+1} \leftarrow {}^{T_0}/_k$, which is faster than the inverse logarithmic rate. The neighborhood function is given by:

$$T_{k+1} \leftarrow {}^{T_0}/_k \tag{4.6}$$

$$\mathbf{y}_{k+1} \leftarrow \mathbf{x}_k + \Delta\mathbf{x}_k \quad \text{where} \quad \Delta\mathbf{x}_k \ э \frac{T_k}{\left(\|\Delta\mathbf{x}_k\|^2 + T_k^2\right)^{\frac{d+1}{2}}} \tag{4.7}$$

where T_0 must be chosen high enough to allow the algorithm to explore the solution landscape. As with the Boltzmann Annealing, the algorithm depends on two cycles, the general or outer cycle, ended with the stopping criterion, and an inner loop also called the thermal equilibrium criterion. The most common choice consists in selecting two integers N_{inner} and N_{outer} to control the behavior of the inner and outer cycle, respectively. In most FSA implementations the temperature is reduced when a certain ratio of accepted solutions has been met for a given temperature. Note that the FSA takes into account the desired thermodynamic-based criteria for a good neighborhood function and cooling schema.

Adaptive Simulated Annealing. The Adaptive Simulated Annealing [Ingber, 1993a] presents a sophisticated cooling schedule and neighbor function that can be summarized as follows: if the optimization process is not satisfied at a given temperature, by consulting the fitness variance, the cooling is *temporarily* stopped in order to take *long jumps*. By means of such an adaptive cooling scheme, the algorithm takes control of the probability distribution of the transitions during the stochastic search by taking into account both the acceptance rate and the variance of the visiting distribution.

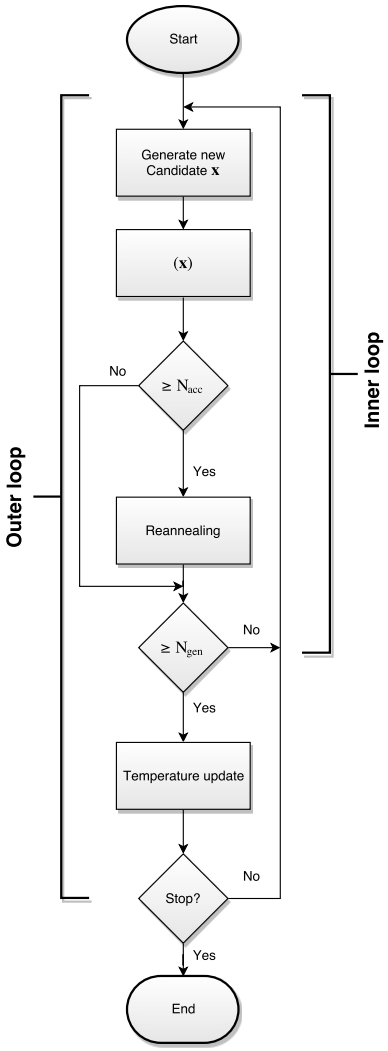


Figure 4.2: ASA.

The ASA algorithm has over 100 parameters that can be fine-tuned; it is beyond the scope of this work to explain with enough detail the internal workings of the method. Instead, we will introduce the cooling schedule and visiting distribution which play an essential role in the evolution of the Simulated Annealing and in our implementation.

One of the principal contributions of the ASA algorithm consists in the introduction of two set of temperatures, one for each decision variable, namely: the acceptance $T_{k,i}^{acc}$ and generation temperature $T_{k,i}^{gen}$ (the i index indicates the decision vector component).

The neighborhood function takes the form of

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \boldsymbol{\alpha} \odot (\mathbf{u} - \mathbf{l}) \quad \text{with} \quad \boldsymbol{\alpha} = \operatorname{sign}\left(\mathbf{p} - \tfrac{1}{2}\right) \mathbf{T}^{gen} \tag{4.8}$$

where $\mathbf{u}, \mathbf{l}$ are the upper and lower bounds for the decision vector, $\mathbf{p} \ni \mathcal{U}^d [0,1]$ and the generation temperature is updated according to:

$$T_{\mathbf{k}}^{gen} = \frac{\max(\mathbf{s})}{\mathbf{s}} T_{\mathbf{k}}^{gen} \quad \text{with} \quad \begin{cases} \mathbf{s} = \left\| \frac{f(\mathbf{x}_{best}+\boldsymbol{\delta}) - f(\mathbf{x}_{best})}{\boldsymbol{\delta}} \right\| \\ \mathbf{k} = \left(-\frac{1}{c} \log \left(\frac{\mathbf{T}_{\mathbf{k}}^{gen}}{\mathbf{T}_0^{gen}} \right) \right) \end{cases} \tag{4.9}$$

where $\mathbf{x}_{best}$ represents the best point so far and vector $\mathbf{s}$ is usually referred as the sensitivities, one for each decision variable. This temperature update takes place after N_{acc} points have been accepted, the so called re-annealing. Similarly the acceptance temperature vector, which is the control parameter used in equation (4.3), is updated according to:

$$k_{acc} = \left(-\frac{1}{c} \log \left(\frac{T_{k_{acc}}^{acc}}{T_0^{acc}} \right) \right)^d \tag{4.10}$$

Finally, after N_{gen} generated points, the temperatures and its indexes are updated:

$$\mathbf{k} \leftarrow \mathbf{k} + \mathbf{1} \quad \mathbf{T}^{gen} = \mathbf{T}_0^{gen} \exp\left(-c\mathbf{k}^{\frac{1}{n}}\right) \tag{4.11}$$

$$k_{acc} \leftarrow k_{acc} + 1 \quad T_{k_{acc}}^{acc} = T_0^{acc} \exp\left(-ck_{acc}^{\frac{1}{d}}\right) \tag{4.12}$$

The reasoning behind this highly sophisticated $\mathfrak{S} - \mathfrak{N}$ schema is to adapt the temperatures to the fitness landscape and to search for a steeper and more *sensitive* dimension where the visiting distribution should have a narrower shape than that of a component less sensitive to change. One advantage of these adaptiveness is that the initial set of temperatures plays a less significant role than in other approaches, as it is going to be recursively changed to adopt the variations on the fitness function.

The most critical choices are the parameters that decide when to update the temperatures N_{gen}, N_{acc} and the annealing rate control parameter c. For most problems, it is enough to set N_{acc} and N_{gen} in the order of hundreds and thousands, respectively.

We include a flowchart, Figure 4.2, of the algorithm in order to showcase the differences between the ASA and the other approaches, which are more akin to the classical version.

The Adaptive Simulated Annealing is without a doubt the most tested variant of the SA; we recommend the reading of [Ingber, 1996] for a better insight on the algorithm.

General Simulated Annealing. We conclude our discussion with the General Simulated Annealing (GSA) [Tsallis and Stariolo, 1996], a unified vision of the Simulated Annealing algorithm which parameterizes both, the next candidate distribution and the cooling schedule, in order to provide a generalized framework where the classical SA [Kirkpatrick et al., 1983] and the Fast Annealing [Szu and Hartley, 1987], correspond to a specific choice of the parameters.

Let $T_{q_v,k}$ be the temperature at iteration $k > 0$, then a next candidate $\mathbf{x}_k$ is found by altering it with a slight perturbation sampled from a generalized Cauchy-Lorentz distribution given by:

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \Delta(\mathbf{x}_k) \quad \text{with } \Delta(\mathbf{x}_k) \ э \frac{T_{q_v,k}^{-\frac{d}{3-q_v}}}{\left(1 + (q_v - 1)\frac{(\Delta \mathbf{x}_k)^2}{\left(T_{q_v,k}\right)^{\frac{2}{3-q_v}}}\right)^{\frac{1}{q_v-1}+\frac{d-1}{2}}} \tag{4.13}$$

where q_v is a parameter which controls the shape of the distribution and the temperature is updated from a starting value $T_{q_v,0}$ as per:

$$T_{q_v,k+1} = T_{q_v,0}\frac{2^{q_v-1} - 1}{(1+k)^{q_v-1} - 1} \tag{4.14}$$

If $q_v = 1$, the neighbor function given by equation (4.13) and the temperature update (4.14) get transformed into the Boltzmann distribution and its associated cooling schedule, i.e. the classical SA. Instead, the Fast Simulated Annealing is recovered by setting $q_v = 2$.

4.2.3 Stopping criterion

The stopping criterion, as its name would suggest, addresses the question about when to finalize the execution of the algorithm. As it happens in other parts of the method, it is impossible to give a universal rule that guarantees that the global optimum has been reached, due to the (often) unknown nature of the fitness landscape. Furthermore, most of the times there is not enough information to even assert that it has been reached with a desired probability. In order to circumvent these difficulties most SA implementations resort to a set of heuristic rules. The underlying idea behind almost every criteria consists in stopping the algorithm when not enough progress has been made during a certain number of iterations. Following is a brief overview of some relevant stopping criteria.

The natural principle of every major stochastic optimization algorithm is to run the method for a prescribed number of iterations. Most SA implementations let the method run for a massive number of iterations with an heuristic stopping rule checked at each iteration.

One common rule consists in stopping either when no new points has been accepted for a predefined number of iterations [Bohachevsky et al., 1986] or when the acceptance ratio falls below a parameter settled at start time [Ingber, 1993b].

In [Vanderbilt and Louie, 1984] the algorithm is stopped by looking into the fitness variance each ν iterations. Let X_ν the produced states at the last ν iterations, and f_ν, f_ν^{best} the mean and minimum of their associated fitness values, then the algorithm is stopped after its difference fall below a threshold adapted at each thermal cycle: i.e. $f_\nu - f_\nu^{best} \leq \epsilon f_\nu$, where $\epsilon < 1$ is a parameter chosen by the user.

Note that the previous rule can be generalized, to a more thermodynamics-driven implementation, if the rule is checked after each time the thermal equilibrium is reached (for a given temperature). See for example [Corana et al., 1987] where this rule is implemented by replacing f_ν with the best global fitness found by the algorithm.

A more convoluted variation of the above rules is presented in [Dekkers and Aarts, 1991]: the method is stopped when the temperature falls below a given small value and the variance of the cost function is small enough, but instead of using the fitness values directly, they are approximated with a smooth functional with a known closed derivative, so the variation can be expressed in terms of its gradient.

4.2.4 On the computation of the initial temperature

We conclude the Simulated Annealing section by addressing the role of the initial temperature, the artificial control parameter used to simulate the melting stage.

In theory, the start temperature must be high enough to let the system have sufficient energy to explore the entire solution space. A very high T_0 can lead to a system with vast amounts of energy, thus being a pure random walk, which is neither intuitive in a thermodynamic sense nor useful in a computational way: a lot of iterations are going to be wasted until the visiting distributions provides a more guided approach. Instead, if the initial temperature is not high enough, the system can be led to a premature convergence as the exploration resembles a gritty search too early, i.e. the system does not have enough energy to let the particles move freely.

Note that the initial temperature is extremely problem-dependent, as we are talking about the system energy which is directly linked to the fitness function. To control the behavior of the system at the first stages several authors have proposed a set of heuristics for the determination of the start temperature. What follows is a brief summary of the most important ones.

The Classical Simulated Annealing [Kirkpatrick et al., 1983] relies on a very intuitive and thermodynamics-driven heuristic: before the start of the algorithm choose a very high initial temperature, then perform a random walk over the solution space recording each transition acceptance status. If the ratio of accepted solutions is not high enough (usually 80%) increase the temperature by a factor of 2. This heuristic is repeated until the ratio is beyond the desired acceptance threshold. In case the ratio is too high (usually 90%), divide the temperature by 2.

In [van Laarhoven and Aarts, 1987, Dekkers and Aarts, 1991] it is suggested that the fitness distribution must be correctly approximated during the initial stages. To this end, the method estimates the energy distribution when $T_0 \to \infty$ by performing random walks on the solution space and computing their acceptance status, thus the expected value $E_\infty(\Delta f)$ is computed in order to set the start temperature as $T_0 = \kappa E_\infty$ (where κ ranges between 5 and 10).

In the case of combinatorial optimization, also applicable to the continuous case, one of the most used techniques for the initial temperature estimation was proposed in [Johnson et al., 1989, Johnson et al., 1991]. Let $X_0^+ = \{\Delta(\mathbf{x}_k)\}_k$ a set of randomly chosen positive transitions, and χ_0 the desired acceptance ratio, typically 0.8. Then, the initial temperature is given by:

$$T_0 = -\frac{\text{mean}(X_\Delta)}{\log(\chi_0)}$$

which can be summarized as the average of the temperatures needed to raise the fitness with a probability of χ_0.

In [Ben-Ameur, 2004] an algorithm is provided in order to compute an initial temperature which is compatible with a given acceptance ratio. This procedure is detailed in Algorithm 4.2 as it is used in our Simulated Annealing implementation for computing the parameters of a Rational Bézier Curve.

Let $\chi_0 \in [0,1]$ be the desired acceptance probability, $X_0^+ = \{\Delta(\mathbf{x}_n)\}_n$ a random set of positive transitions, we define $\hat{\chi}(T_k)$ as:

$$\hat{\chi}(T) = \frac{\sum_n \exp\left(-\frac{f_n^+}{T}\right)}{\sum_n \exp\left(-\frac{f_n^-}{T}\right)} \tag{4.15}$$

where f_n^+ and f_n^- denote the state after and before the transition Δ_n, respectively.

Algorithm 4.2: Computing the initial temperature

Input: Desired acceptance probability χ_0, cost function f, $\epsilon_0 \ll 1$, $p > 1$
Output: Initial temperature T_0
Sample the solution space and store the positive transitions X_0^+
$i \leftarrow 1$
$T_i \ni \mathcal{U}(0,1)$
do
  Compute $val \leftarrow \hat{\chi}(T_i)$ by following Equation (4.15)
  $T_0 \leftarrow T_i$
  $T_{i+1} \leftarrow T_i \left(\frac{\log(\hat{\chi}(T_i))}{\log \chi_0}\right)^{1/p}$
while $|val - \chi_0| \leq \epsilon_0$
return T_0

Chapter 5
The Method

> All models are wrong, but some are useful.
>
> George Box

The aim of this Thesis is to find a free-form spline model from a point cloud which does not require any subjective parameter for its computation. The resulting model must provide a good trade-off between data-fidelity and complexity in a fully automated way.

The rest of this chapter is organized as follows: we begin with a brief outline of the underlying ideas behind the methodology. Then, related works on spline fitting model selection are discussed. Finally, we conclude the chapter with a discussion of the different approaches followed.

5.1 Outline

In its most general form our methodology, as outlined in Figure 5.1, can be summarized as follows: given a point cloud $\{\mathbf{Q}_i\}_i$ in $\mathbb{R}^d$, select a set of spline models to approximate the data, construct a fitness function that takes into account the model parameters and contains enough information about the model data-fidelity, optimize it with the Simulated Annealing algorithm to get a final fitness value, and the associated spline parameters, for each model. Finally, compare all the models through the use of an information sciences criteria, thus producing one new fitness value for each model, then select the model with the best transformed fitness.

5.1.1 Model selection

As discussed above, our problem consists of reconstructing the underlying shape of a given set of noisy data points by using a spline model. The model is fully determined by the spline choice and the problem being solved, see Table 5.1 for a summary of each problem-model complexity.

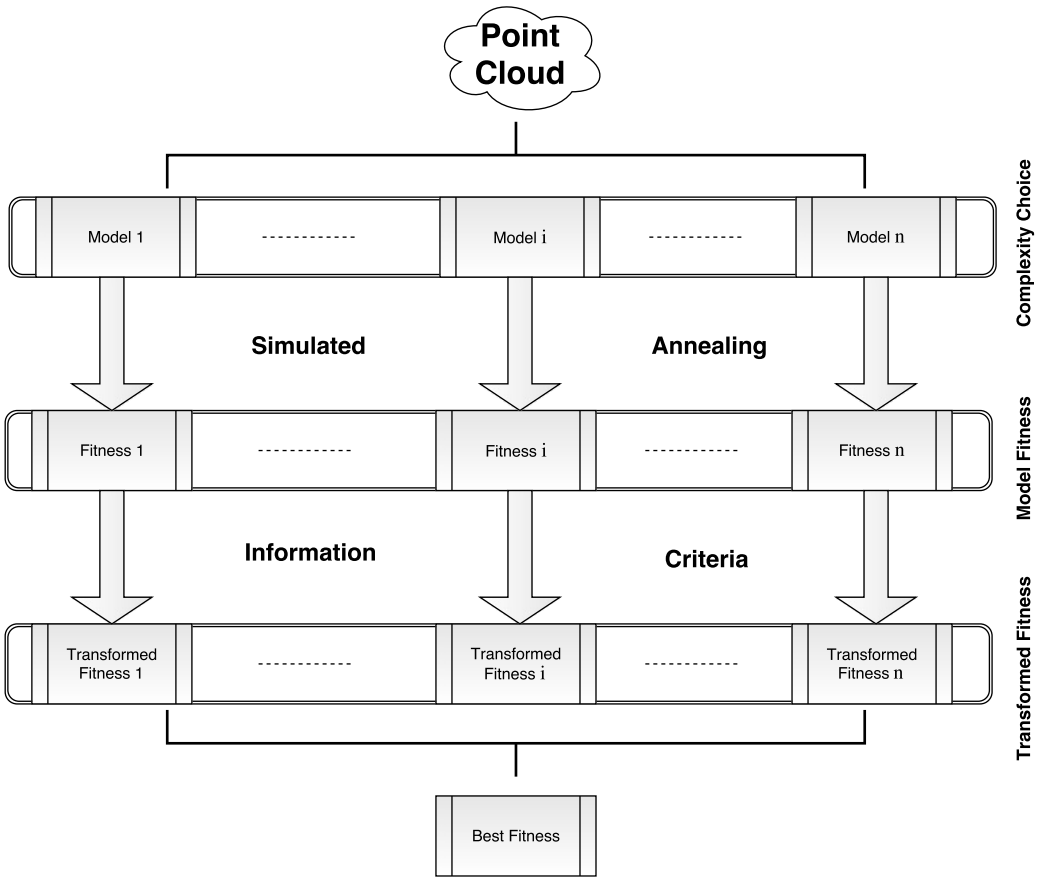


Figure 5.1: General diagram of the proposed methodology.

The complexity of each model is unequivocally determined by the number of free parameters required to define the spline being used. For example, the number of free parameters needed to define a NURBS was constructed in equation 3.2. Note that, the same procedure can be formulated for each of the spline models considered in this work, namely: rational and non-rational Bézier/B-spline curves and surfaces.

5.1.2 The fitness function

Once an underlying spline model is chosen, we need a measure of the quality of the model. At this stage, there is no need for the fidelity metric, also known as fitness or cost function, to capture any information about the complexity or other characteristics of the data to be fitted.

In all our implementations, the residual sum of squares RSS is used as the performance metric to measure the data-fidelity. Thus, for a given spline model, we use the RSS as the fitness function for the Simulated Annealing. Note that is choice is compatible with the proposed AIC (5.1) and BIC (5.2) criteria.

Note that, as the complexity increases so does the non-linearity of the objective function, as per equation 3.1. Thus, although a NURBS is more feature complete than

Model	Control Points	Parameters	Weights	Knots
Bézier	✓	✓		
Rational Bézier	✓	✓	✓	
Free-knot spline	✓			✓
Parametric B-spline	✓	✓		✓
NURBS	✓	✓	✓	✓

Table 5.1: Model complexity summary table.

any other spline model, it is still needed a general method to obtain less complex models that can be numerically more stable, faster and easier to understand and manipulate.

5.1.3 Information Sciences Criteria

The Akaike Information Criterion (AIC) [Akaike, 1974, Akaike, 1998] and the Bayesian Information Criterion [Schwarz, 1978] are penalized information-theoretical criteria which involve the creation of a new fitness function with the aim of providing a fair trade-off between data-fidelity and complexity. The new measure is comprised of two competing terms: on the one hand the residual sum of squares (RSS) is used to provide an estimation of the model fidelity, on the other hand a penalty term is introduced in order to take into account the complexity via the number of free parameters of the model, η. The AIC is given by:

$$\mathrm{AIC} = N \log(\mathrm{RSS}) + 2\eta \tag{5.1}$$

whereas the BIC is given by:

$$\mathrm{BIC} = N \log(\mathrm{RSS}) + \eta \log(N) \tag{5.2}$$

where N is the number of parameters to be fitted and log is the natural logarithm.

In the particular case of data fitting, the main advantages of AIC and BIC criteria is that they allow us to select a model in a fully automated way, without the need of any subjective decision or parameter: select the model with the lower transformed fitness. Note that for a given problem, i.e. N is fixed, if we fix the number of free parameters η, both criteria behave like the original fitness. Instead, for a given error, RSS, the transformed fitness increases along the model complexity.

Although both criteria produce similar models, it is important to note that, due to the penalizing term expression, the BIC tends to produce simpler models than the AIC. This simple fact has an important ramification in the case of data fitting with splines: the BIC, thanks to yielding less unnecessary parameters than the AIC, provides a more robust model when the underlying shape of the data exhibits sharp elbows, discontinuities or rough localized areas. The problem arises especially in the case of free-knot splines, as evidenced in [Yoshimoto et al., 2003, Gálvez and Iglesias, 2011, Gálvez and Iglesias, 2013, Gálvez et al., 2015] and our take on the problem in Section 7.1, where the issue gets a more detailed explanation.

5.1.4 Other methodologies

Although our method is in accordance with other state of the art nature-inspired methodologies for spline fitting [Yoshimoto et al., 2003, Gálvez and Iglesias, 2011, Gálvez et al., 2015] here we will discuss other model selection techniques.

The most common methods for spline approximation [Farin, 2002] are based on one of two opposite procedures. On the one hand, a class of methods begins with a small set of knots, or control points in the case of Bézier models, and then iteratively increase its numbers after a certain condition is met (usually a desired precision). On the other hand, the reverse procedure starts from a large number of knots, or control points, and then the set is reduced until a threshold is no longer met. Both procedures rely on a good initial set of breakpoints or control points, the problem itself is a difficult one and it is extremely complicated in the case of free-knot splines [Jupp, 1978]. Furthermore, both methodologies tend to fail in the case of noisy or large point clouds.

In the case of Bézier curves and surfaces the typical use case does not involve a model selection procedure, as the data fitting is made by the smooth connection of cubic Bézier curves or surface patches. These methods are compatible with the requirements of certain industries, such as font design and storage[1] [Yannis Haralambous, 2007]. Note however, that our methodology can indeed meet those requirements by making use of the Bézier subdivision capabilities. Therefore, our final model could be subdivided into a path of cubic Bézier curves through one of the many robust methods found in the literature [Warren and Weimer, 2001].

5.2 Scattered data

Let us assume that a given point cloud is composed of samples of a real-valued function of one or two variables with an unknown topology, i.e. we do not know the connectivity between the data points. The construction of a digital model of such a dataset is of extreme importance in such areas as high-resolution terrain maps, laser scans or medical data [Nielson, 1993].

A dataset with an unknown ordering is commonly referred in the literature as a *scattered point cloud*. Although it has not been explicitly pointed out, the proposed methodology assumes a point cloud with a known topology. Our approach to finding a suitable connectivity for the dataset consists in constructing a baseline surface, with a known topology, that approximates the data. By evaluating this baseline surface we get a transformed point cloud with a known topology as close to the original point cloud as we want, so we can now apply the previously presented methodology (Figure 5.1). The baseline surface is obtained by means of the natural neighbor interpolation method.

[1] For example, cubic Bézier paths are used as the basis of the Compact Font Format (CFF), part of the OpenType Standard

5.2.1 The Sibson method

The natural neighbor interpolation method [Sibson, 1981], also known as the Sibson method, is a powerful general-purpose spatial-data estimation technique based on combining a local weighted-average interpolation over a query point via its neighbors (in a Voronoi sense). The resulting transformed data has some useful properties, such as $\mathcal{C}^1$ continuity, locality and a good fitting error. The major drawbacks of the method are its computational inefficiency and difficult implementation.

However, many methods have been proposed to accelerate the procedure. A Vornoi diagram can be efficiently discretized and computed through the use of graphics hardware [Hoff III et al., 1999], a shortcut which can be used to accelerate the Sibson method. In [Park et al., 2006] a method is presented in order to avoid the computation of the full Voronoi diagram, a very demanding task, via the use of *kd-trees* to find the nearest neighbors of only a subset of the point cloud.

5.2.2 Constructing the natural neighbor interpolant

A full discussion of the method is well beyond the aim of this dissertation; however we will provide a brief outline of the method. First we will introduce the necessary mathematical concepts behind the interpolation, then we conclude by examining the Sibson interpolation technique.

Voronoi diagram. A Voronoi diagram of a convex domain $D \subset \mathbb{R}^d$ is a partitioning of D into a series of tiles (cells) which are based on a finite set of points (sites). A Voronoi diagram of the set $X = \mathbf{x}_1^N$ over D, denoted as $\blacktriangle(X)$, parts the domain into regions knows as cells $\upsilon_{\mathbf{x}_i} \subset X$, in a way that any given point in the cell is closer to $\mathbf{x}_i$ than to any other point in X.

Natural neighbors. Given a set of sites and its associated diagram, the natural neighbors of a query point $\mathbf{y}$ are computed as follows. If we insert the query point into the Voronoi diagram, the Voronoi cell $\upsilon_{\mathbf{y}}$ has k neighboring cells $\{\upsilon_{\mathbf{x}_1}, \ldots, \upsilon_{\mathbf{x}_k}\}$. The k associated sites are known as the natural neighbors of $\mathbf{y}$.

Natural Neighbor Interpolation. In its most basic form the interpolant can be written as $\mathbf{I}(u_0, v_0) = \sum_{\mu=1}^{D} \xi_{\mu,0}\mathbf{Q}_u$. For a given (u_0, v_0) the method computes a Delauney triangulation $\blacktriangle(\mathbf{Q})$ of the data in order to find the closest nodes that form a convex hull around the query point, then the associated weights $\xi_{\mu,0}$ are calculated by finding how much area could be *stolen* when inserting the point into $\blacktriangle$.

To be more precise, let say that $\mathbf{y}$ is the point where we want the interpolation to be done. The first step consists in constructing a Voronoi diagram of our point with and without $\mathbf{y}$: $\blacktriangle^+$ and $\blacktriangle^-$ respectively. If a Voronoi Cell of a given point $\mathbf{x}$ lies in $\blacktriangle^+$, we label it as $\upsilon_{\mathbf{x}}^+$, in the other case $\upsilon_{\mathbf{x}}^-$. Now, we construct the discrete set of natural neighbors of $\mathbf{y}$, $N_{\mathbf{y}} = \left\{ n \in \mathbb{N} : \ \mathbf{y}_n \text{ is a natural neighbor of } \upsilon_{\mathbf{y}}^- \right\}$. The

weights are computed by finding the *stolen* volumes from the Voronoi diagram when $\mathbf{y}$ is added by following:

$$w_n = \begin{cases} \frac{\mathrm{vol}(v_n^+)}{\mathrm{vol}(v_n^-)} & \text{if } n \in \mathbb{N} \\ 0 \quad \text{otherwise} \end{cases} \tag{5.3}$$

Finally, the estimated value of a point is computed by the resulting weighted sum: $\sum_n w_n \mathbf{y}_n$.

Part III

Reverse Engineering Applications

Chapter 6
Curve and Surface fitting with Bézier models

If your system were that good, the Americans would have invented it first!

A short-sighted manager to Bézier.

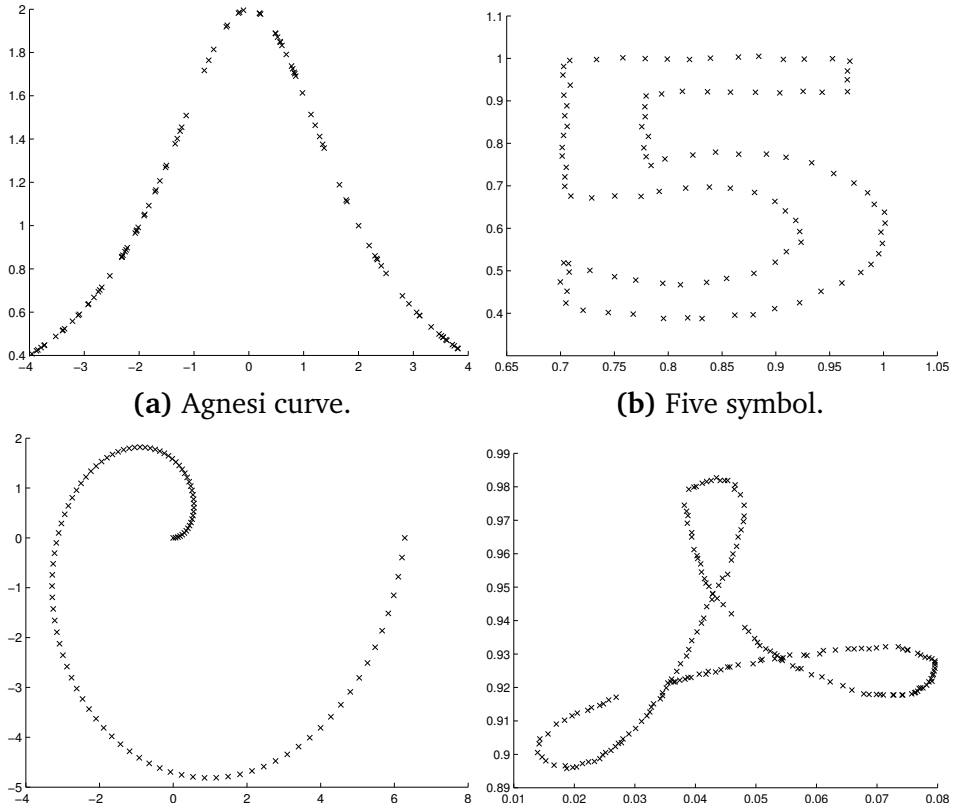
6.1 Non-rational Bézier Curves

This section is devoted to present the results of *Simulated Annealing Algorithm for Bézier Curve Approximation* presented in the *Cyberworlds 2014* [Loucera et al., 2014].

6.1.1 Introduction

Curve approximation is a very important topic in many industrial and applied fields. The typical input in real-world applications is a set of sampled data points for which a fitting curve is to be obtained. This work addresses this problem by using Bézier curves as the approximating functions. This formulation leads to a continuous multivariate nonlinear optimization problem. Unfortunately, this is very difficult problem that cannot be solved with classical mathematical optimization techniques. In this chapter, we solve the problem through a hybrid strategy combining classical methods (linear least-squares minimization), modern stochastic methods (Simulated Annealing) and information science metrics. For a given degree n, our method computes a near-to-optimal parameterization of data points by using Simulated Annealing for global search and a local search optimizer for further refinement of the global solution. Then, we compute the control points by least-squares minimization. Finally, we determine the best value for the degree of the curve by using two information science metrics that represent an adequate compromise between data-fidelity and

model-complexity. Our method is applied to four illustrative examples of mathematical curves and noisy scanned data and different configurations. Our experimental results show that the method performs well for all examples.



(a) Agnesi curve. **(b)** Five symbol.

(c) Archimedes. **(d)** Famous logo.

Figure 6.1: Original point clouds.

6.1.2 The problem

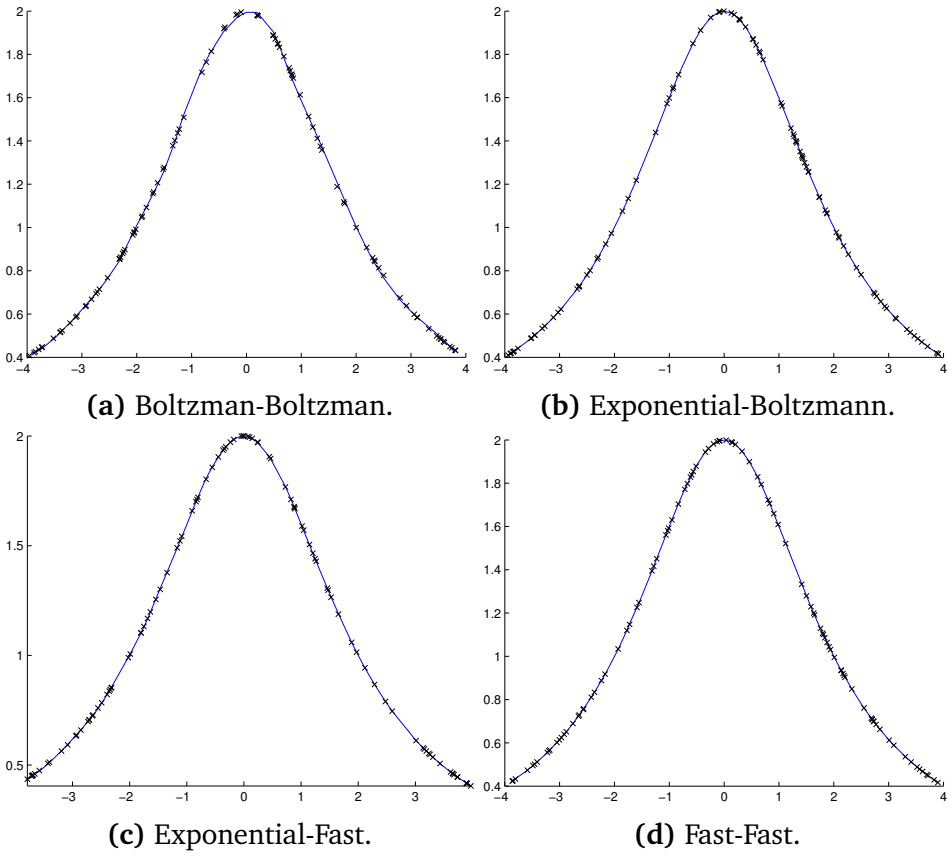
The problem being solved in this section is a smaller subset of the fitting scenario outlined in Chapter 3. The problem can be summarized as follows: let $\{\mathbf{Q}_i\}_{i=1}^m$ be a set of points in $\mathbb{R}^d$, our aim is to find $\mathbf{s}$, the best fitting non-rational Bézier curve of degree n, as defined per equation (2.1).

The fitting problem gets transformed into the following least-squares minimization:

$$E = \sum_{i=1}^{m} \mu_i \left\| \mathbf{Q}_i - \sum_{j=0}^{n} \mathbf{P}_j B_j^n(t_i) \right\|_2^2 \tag{6.1}$$

where $\{\mu_i\}_{i=1,\ldots,m}$ are scalar numbers used in situations when it may not be reasonable to assume that every sampled data should be treated equally. Note that we need to associate a parameter t_i to each data point $\{\mathbf{Q}_i\}$. It is obvious that, since each

blending functions is non-linear in t, the system (6.1) is also non-linear. Therefore, we are dealing with a continuous nonlinear minimization problem that involves a large number of variables for large sets of data points, a typical case in many real-world technological problems (e.g., the Michelangelo project).



(a) Boltzman-Boltzman. **(b)** Exponential-Boltzmann.

(c) Exponential-Fast. **(d)** Fast-Fast.

Figure 6.2: Agnesi curve: Best fit (BIC-sense) after local search.

6.1.3 Method implementation

Our method consists of three major steps: data parameterization, least-squares minimization and error evaluation. The last step is not only aimed at computing the value of the fitness function (6.1) required for second step, but also at determining the best value for the degree n of the curve. This is a critical task: to decrease the complexity of our problem, it is convenient to choose a low value for the degree of the curve; however, a very low value of n also means less degrees of freedom and hence, more difficulties to describe the underlying shape of data. On the other hand, the curve degree is the only available parameter to accomplish this goal, since we have no control over the size of the point cloud. We compare the Akaike and Bayesian Information Criteria for the model selection.

Simulated Annealing. Data parameterization is computed by means of the minimization of equation (6.1) by making use of the Simulated Annealing algorithm. The implementation used here follows the classical SA schema presented in Algorithm 4.1. We consider a *memory effect* in the form of *elitism*: we allow the best state from the current generation to carry over to the next, unaltered. This feature provides faster convergence rates with respect to the non-elitist variant and improves the memory capacity of the approach. Finally we test a set of different neighbor functions and cooling schemes:

Boltzmann annealing. It performs a step-size equal to the square root of the current temperature and the direction vector is uniformly random.

Fast annealing approximation. It consists of performing steps of length proportional to the current temperature, and the direction is uniformly random.

Power schedule. $T_{k+1} = 0.95^k T_0$

Boltzman schedule. $T_{k+1} = \dfrac{T_0}{\log(k)}$

Fast schedule. $T_{k+1} = \dfrac{T_0}{k}$

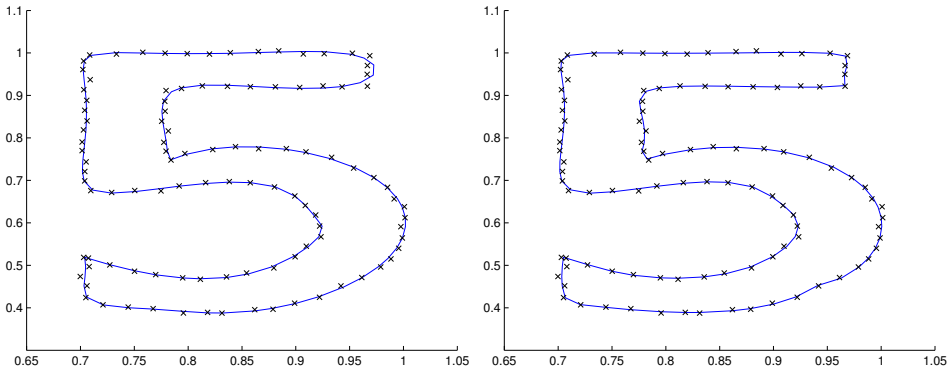
6.1.4 Experimental Results

To evaluate the performance of our approach, it has been applied to different examples with varying difficulty levels. In this section, we analyze four of them. Our examples include two scanned figures containing a significant amount of noise (examples 1 and 4, corresponding to the five character and a famous logo, respectively), and two noise-free mathematical curves (examples 2 and 3, corresponding to Agnesi's curve and Archimedes spiral curve respectively). They are represented graphically in Figure 6.1.

Parameter selection and general test configuration

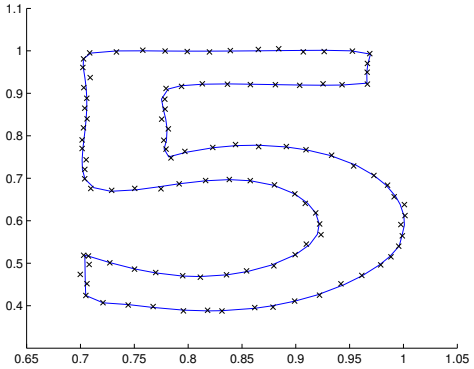
For each example we fit the data to a Bézier curve of degree n, with n ranging from 4 to 90. To remove the spurious effects of stochasticity, each example is executed 20 times for each value of n and for each simulated annealing parameter configuration. In this paper we consider the following four configurations:

- Boltzmann's cooling schedule & neighborhood function.
- Exponential cooling schedule & Boltzmann's neighborhood function.
- Exponential cooling schedule & Fast Annealing neighborhood function.
- Fast Annealing cooling schedule & neighborhood function.

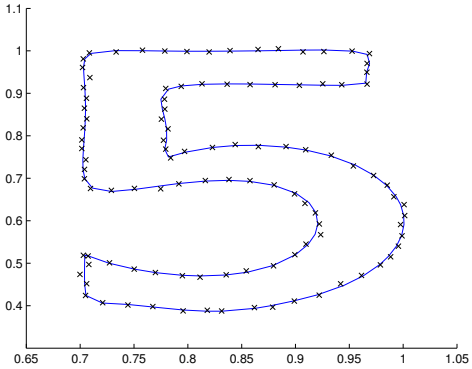


(a) Boltzman-Boltzman.

(b) Exponential-Boltzmann.



(c) Exponential-Fast.



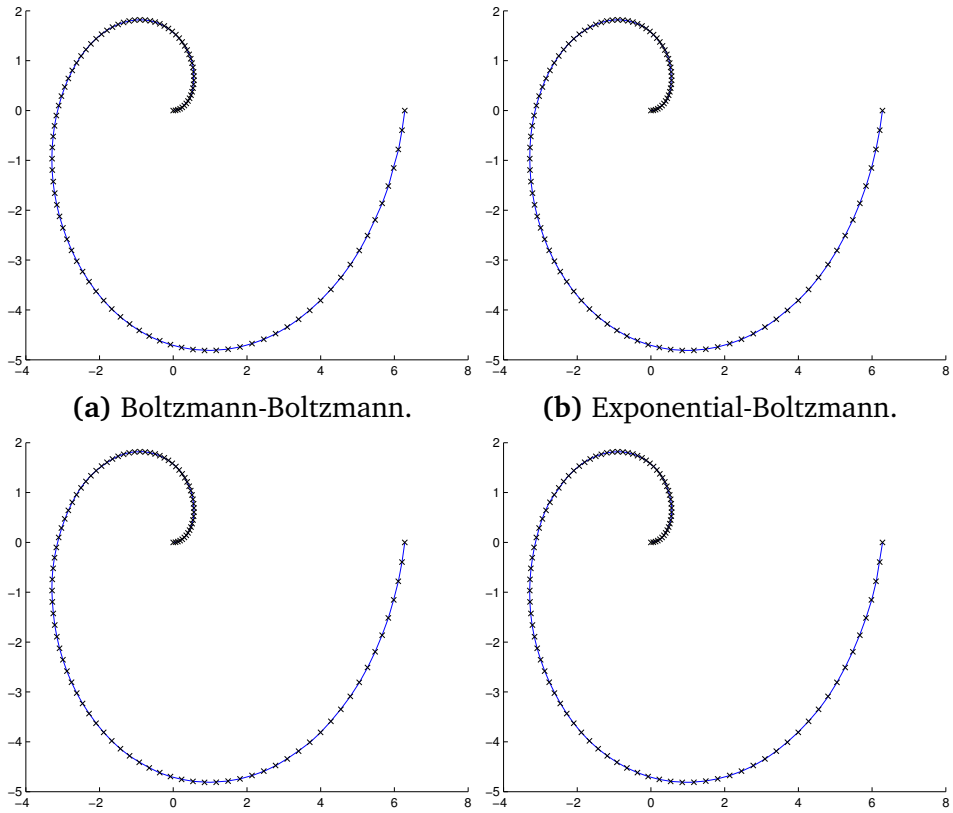
(d) Fast-Fast.

Figure 6.3: Five symbol: Best fit (BIC-sense) after local search.

Performance discussion

Our experimental results for the four examples are displayed in Figures 6.3 to 6.5. Each figure shows the best fitting curve according to the BIC criterion after local search. Table 6.1 summarizes our main results. The corresponding examples are listed in rows, for each of the configurations indicated above (where B, E, FA, T and N stand for Boltzman, exponential, fast annealing, cooling schedule, and neighborhood function, respectively). The table shows the best and average values of AIC and BIC computed in terms of the Bézier curve degree n, along with the optimal value for this parameter n. The table also reports such results before and after the local search stage. Note the substantial improvement of our results for the latter case. Finally, mean errors after local search for coordinates x and y are reported.

From our experimental results we conclude that the method performs pretty well, as we have been able to reconstruct the underlying shape of data in all cases. As expected, the method performs better in the cases of mathematical curves, because they are unaffected by noise. In such cases, we obtain an excellent visual quality, as shown in Figures 6.2 and 6.4. The method also performs well for noisy scanned data, but the quality of reconstruction is visually less appealing. Also, in those cases our method is more sensitive to the configuration used for each example. This fact is clearly visible in Figure 6.5, where some configurations find troubles to reconstruct



(a) Boltzmann-Boltzmann. **(b)** Exponential-Boltzmann.

(c) Exponential-Fast. **(d)** Fast-Fast.

Figure 6.4: Archimedes spiral: Best fit (BIC-sense) after local search.

some parts of the image, remarkably the leftmost loop. Note, however, that this example is particularly challenging because it is too noisy and contains several self-intersections and changes of concavity. Yet, the method captures well the overall shape of data for all configurations.

All our experiments have been done on an AMD-FX^{tm}-4100 Quad-Core Processor at 3600 *Mhz* with 8GB DDR3 RAM running *Debian 7.1* and *MATLAB 2012a*.

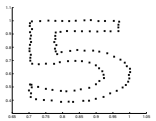
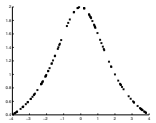
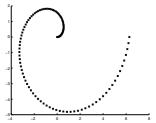
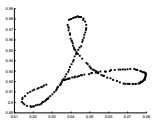
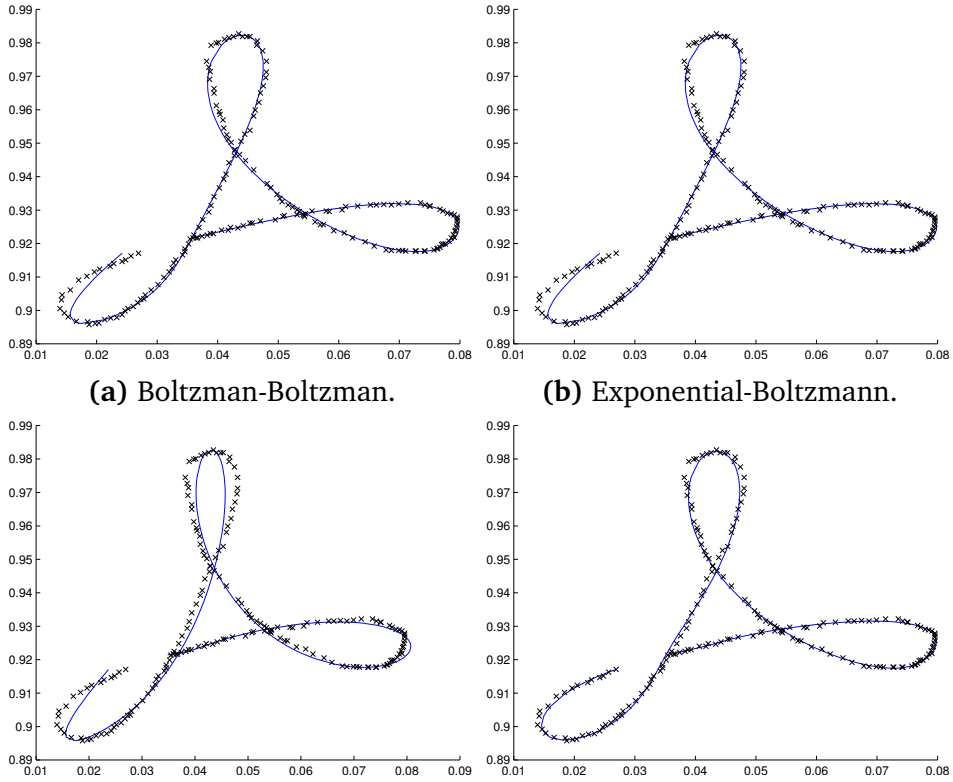
			global						local							
example	T	N	bic	bic_avg	o_bic	aic	aic_avg	o_aic	l_bic	l_bic_avg	l_o_bic	l_aic	l_aic_avg	l_o_aic	l_xmean	l_ymean
	B	B	267.2796	289.5752	21	-122.8525	-100.5570	32	-63.7532	-62.5	21	-453.8853	-452.0737	30	6.96e-03	7.07e-03
	E	B	252.5929	289.1011	16	-110.9996	-74.4915	32	-88.5238	-87.625	25	-499.8877	-469.77	25	6.17e-03	6.72e-03
	E	FA	278.8890	303.2310	20	-105.9353	-81.5933	43	-102.0414	-101.001	23	-502.7894	-501.94	25	5.85e-03	8.94e-03
	FA	FA	258.9516	294.1581	15	-99.3330	-64.1266	37	-101.9477	-101.90	21	-492.0799	-492.01	29	6.07e-03	7.36e-03
	B	B	199.29	285.41	48	-232.16	-67.612	77	-881.63	-881.22	15	-1181.2	-1180.5	15	0.0096324	6.3645e-05
	E	B	247.19	296.53	28	-182.17	-3.0624	71	-853.24	-853.01	25	-1182.3	-1181.5	30	0.00058944	6.2241e-05
	E	FA	229.89	278.76	17	-195.77	-53.054	78	-888.33	-887.57	19	-1198.3	-1198	19	0.0041865	5.9841e-05
	FA	FA	196.8	261.01	19	-209.77	-49.004	89	-799.87	-799.64	33	-1146.4	-1145.6	33	0.0024992	6.4121e-05
	B	B	447.18	499.91	42	48.771	153.42	88	-805.33	-804.93	24	-1128.4	-1127.4	24	0.0086604	2.9765e-05
	E	B	360.61	476.24	11	25.955	153.2	44	-804.93	-804.28	20	-1117.6	-1117.4	20	0.0062883	7.4593e-05
	E	FA	402.61	474.31	12	39.945	161.69	80	-669.59	-668.81	33	-1016.1	-1015.4	33	0.0083015	0.00010426
	FA	FA	446.88	500.39	23	50.29	153.9	90	-660.7	-660.59	24	-983.74	-983.54	24	0.0079914	0.00013109
	B	B	-322.67	-106.42	18	-683.51	-596.19	49	-723.81	-723.77	12	-1015.6	-1015.2	12	0.00069641	0.00010515
	E	B	-322.67	-265.82	18	-683.51	-557.6	49	-723.81	-723.49	12	-1015.6	-1015.2	12	0.00069641	0.00010515
	E	FA	-320.62	-257.52	16	-666.3	-549.3	39	-714.07	-714.02	10	-1015.3	-1014.5	35	0.0011091	0.00010602
	FA	FA	-320.25	-247.88	29	-663.07	-623.41	45	-691.91	-691.83	13	-1038	-1037.7	35	0.00052515	0.00010989

Table 6.1: Experimental results.



(a) Boltzman-Boltzman. **(b)** Exponential-Boltzmann.

(c) Exponential-Fast. **(d)** Fast-Fast.

Figure 6.5: Famous logo: Best fit (BIC-sense) after local search.

6.2 Bézier Surfaces

This section is devoted to the research article titled *A Simulated Annealing Approach for Data Fitting with Bézier Surfaces* as found in the *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication - IPAC '15* [Iglesias et al., 2015b]

6.2.1 Introduction

This work presents a new method for reconstructing a cloud of noisy data points through Bézier surfaces. Our approach is based on the combination of a single-particle metaheuristic technique called simulated annealing, the Bayesian information criterion, and classical fitting techniques (least-squares). Three illustrative examples are used to show the good performance of this approach.

6.2.2 The problem

A polynomial Bézier surface of degree (n, m) is given by:

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{i,j}\, B_i^n(u)\, B_j^m(v) \tag{6.2}$$

where $u, v \in [0,1]$ are the parameters of the surface, and $\{\mathbf{P}_{i,j}\}$ the control net. See chapter 2 for a more detailed discussion on the topic.

Let now $\{\mathbf{Q}_{k,l}\}_{k=1,\ldots,p;l=1,\ldots,q}$ be a set of data points in $\mathbb{R}^3$. Our objective is to reconstruct the Bézier surface $\mathbf{S}(u,v)$ that fits the data in the discrete least-squares sense, i.e. we want to minimize the sum of squares of residuals:

$$E = \sum_{k=1}^{p}\sum_{l=1}^{q} \left\| \mathbf{Q}_{k,l} - \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{i,j}\, B_i^n(u_k)\, B_j^m(v_l) \right\|_2^2 \tag{6.3}$$

Note that we are dealing with an over-constrained system of equations. Furthermore, it also remains the problem of computing a suitable parameterization of data points. Note that since the blending functions are nonlinear in u and v, the least-squares minimization is a strong nonlinear problem. Our strategy to solve this issue consists of applying the simulated annealing algorithm, explained in next section, for data fitting and the Bayesian Information Criterion for model selection.

6.2.3 Modified Simulated Annealing Method

In the following we will discuss the modifications we have carried out to adapt the classical Simulated Annealing, as outlined in chapter 4 and section 6.1, for Bézier surface reconstruction.

Let $\mathbf{x}^{old}, \mathbf{x}^{new}, \mathbf{x}^{best} \in \mathcal{D} \subseteq \mathbb{R}^d$ be a set of three solutions, at a certain iteration, corresponding to the previous, new and best solutions reached so far. We also have the temperatures $T_0, T^{old}, T^{new} \in \mathbb{R}^+$, and, following the same notation, the system energies $f^{old}, f^{new}, f^{best}$. In section 6.1 four SA schemes were discussed: the Boltzmann-Boltzmann, $\mathcal{B}-\mathcal{B}$, Exponential-Boltzmann, $\mathcal{E}-\mathcal{B}$, Exponential-Fast, $\mathcal{E}-\mathcal{F}$, and Fast-Fast, $\mathcal{F}-\mathcal{F}$. The first and second part of these names refer to the cooling schedule and the neighborhood function, respectively. From them, we have finally opted for the Fast-Fast $\mathcal{F}-\mathcal{F}$ schema, which offers more visually appealing results and is generally faster. In fact, as we have empirically tested, the other schemes typically waste too many iterations without further improvement of the solution. They are also very sensitive to the initial choice of T_0 and the total number of evaluations of the objective function, n_{feval}. The Fast cooling schedule updates the temperature according to the expression T_0/k. In this work, we have also hybridized the SA approach with a simplex method to further strengthen the local search capabilities of our method. The resulting neighborhood function $\mathfrak{N}$ is actually the composition of three functions:

$$\mathfrak{N}_L \circ \mathfrak{N}_{\mathcal{D}} \circ \mathfrak{N}_{\mathcal{F}} \tag{6.4}$$

where: $\mathfrak{N}_F$ is used to obtain a step proportional to the current system temperature, which is subsequently perturbed according to a Gaussian random law; $\mathfrak{N}_D$ is used to ensure that the new generated state is kept within the search space domain $\mathcal{D}$, by sending it back into $\mathcal{D}$ whenever it goes away; and $\mathfrak{N}_L$ performs a very fast local search by making use of a smaller number of evaluations of the target function. This local search is performed by a bounded simplex algorithm.

The acceptance criterion is an adaptation of the Metropolis-Hastings algorithm [Metropolis et al., 1953]. For a minimization problem, it can be described through the following equation:

$$\mathfrak{A}(\varphi) = \begin{cases} 1 & \text{if} \quad f(\varphi) \leq 0 \\ e^{-\varphi/T} & \text{otherwise} \end{cases} \tag{6.5}$$

where φ denotes the transition from $\mathbf{x}^{old}$ to $\mathbf{x}^{new}$, and $f(\varphi) = f^{new} - f^{old}$. Note that equation (6.5) meets the conditions indicated in section 4.2.1.

In addition to these modifications, we have introduced three new improvements into the basic algorithm:

I *Restart strategy.* To make the technique less sensitive to parameter tuning, we restart the annealing until the ratio of rejected transitions versus the accepted ones tends towards 0.8. The primary goal of this restart strategy is to select a good starting temperature even if it is initially overestimated. Another beneficial collateral effect is that of improving the convergence of the $\mathcal{F} - \mathcal{F}$ schema.

II *Memory effect.* We also provide the algorithm with a *two-side memory effect* by storing the best overall solution, $(\mathbf{x}^{best}, f^{best})$, and the best solution for each time the system reaches a thermal equilibrium state.

III *Local search strategy.* At the end of our algorithm, we perform a local search for each of these solutions with a higher number of evaluations of the energy function.

With these additional improvements, our approach outperforms the original SA method described in Sect. 6.1 for the surface reconstruction problem addressed in this paper.

6.2.4 Method implementation

Our method is comprised of three main steps: data parameterization, data fitting, and degree determination. In first step we apply the SA algorithm described above to create a suitable one-to-one mathematical association between the data points $\{\mathbf{Q}_{k,l}\}_{k=1,\ldots,p;l=1,\ldots,q}$ and their corresponding surface parameters (u_k, v_l). In our setting, the states correspond to different associations (parameterizations) of data points, while the energy function is given by equation (6.3). Given a random initial

state, we select the different components of the Simulated Annealing method previously described. As discussed in Section 6.2.3, we selected the Fast cooling schedule and the neighborhood function given by equation (6.4), thus embedding a bounded simplex method for fast local search in our approach. We also select the acceptance criterion given by equation (6.5). The resulting Fast-Fast schema is enriched by its hybridization with a local search method with a higher number of function evaluations, the two-side memory effect (a kind of elitism to keep the best solution achieved so far) and the restart policy described above. Finally, the stopping criteria are given by the three following conditions: (1) $n_{feval} > 1000 * \xi$; (2) $\#[\partial(\Theta)] = 10 * \xi$, and (3) $\#[\partial(\mathbf{x}^{best})] = 10 * \xi$, where n_{feval} means the total number of evaluations of the energy function, ξ denotes the number of free parameters of the problem, and the notation $\partial(.)$ is used to indicate the lack of changes of the mean variation of the corresponding input function for a given number of function evaluations. Thus, the second rule means that the procedure stops if the mean value of the energy function E does not change during $10 * \xi$ function evaluations. Similarly, the procedure stops if the mean value of the best state $\mathbf{x}^{best}$ does not change for the same number of function evaluations. In both rules, the number of function evaluations does not include those performed during the local search phase. The function $\mathfrak{N}_L$ performs local search by using a smaller number of evaluations, set to $\xi/2$ in this work.

The data fitting step consists of the least-squares minimization of kernel (6.3) to compute the Bézier surface coefficients. This process can be achieved by either LU decomposition or singular value decomposition (SVD), which provides the best numerical answer for the Moore-Penrose pseudo-inverse of the least-squares problem. Finally, the optimal degree value (n^*, m^*) for the Bézier surface is determined by using the Bayesian Information Criterion (BIC). This is achieved by introducing a penalty term on the number of free parameters of the model, described by: $BIC = \eta \log(E) + \xi \log(\eta)$ where log denotes the natural logarithm and η is the number of sampled points. We have applied this criterion in order to obtain a Bézier surface fitting the data points with high accuracy but with a minimal degree. In other words, our fitting surface is not obtained at the expense of an unnecessarily large value for the surface degree, thus preventing overfitting to happen.

6.2.5 Experimental results

Our surface reconstruction approach has been applied to three illustrative examples: a random surface (Figure 6.6), a noisy digital representation of a heart (Figure 6.7) and a noisy parametric surface representing a seashell (Figure 6.8).

First example corresponds to a Bézier patch generated from a collection of 4×4 randomly chosen control points. From this surface, a collection of 900 points were sampled and then perturbed at each component independently by a Gaussian additive noise of mean zero and standard variation $\sigma = 0.2$. Then, it was fitted with our approach by using a Bézier surface with degree varying from 3 to 10 for each component. Figure 6.6 shows the noisy data points of this example along with the best fitting surface, obtained for degree $(3, 3)$ according to the BIC measure. The corre-

sponding mean value of the fitting error between the original and the reconstructed data points for each coordinate are reported in Table 6.2. We have performed 10 simulations for each choice of (n,m) to remove any undesirable spurious effect derived from the randomness of the process. The table reports the mean value (left column) and the best value (right column) of these 10 simulations. The table also reports the number of data points and the value of σ used in this example (first and second rows of the table, respectively). This table also provides similar stats for each example, they are differentiated by the accompanying graphical representation. Second example corresponds to a collection of 1891 data points sampled from a computer model of a human heart. These data points have been obtained from [Malchenko, 2005] and then perturbed at each component by a Gaussian additive noise of mean zero and standard variation $\sigma = 0.1$. Then, we applied our approach with degrees varying from 3 to 40 for each component. Results for the optimal value, obtained for $(n,m) = (11,17)$, are reported in Table 6.3. Last example corresponds to a collection of 3600 data points sampled from a popular parametric surface usually called seashell and subjected to Gaussian additive noise of mean zero and standard variation $\sigma = 0.05$. Once again, we applied our approach with degrees varying from 3 to 40 for each component. Results for the optimal value are reported in Table 6.3. As the reader can see from Table 6.2, we obtain a very good fitting of data points, with mean errors ranging from order $10^{-5} \sim 10^{-6}$ for the first example to order 10^{-3} for the heart and seashell examples. These values are obtained even in presence of additive white noise of low-medium intensity.

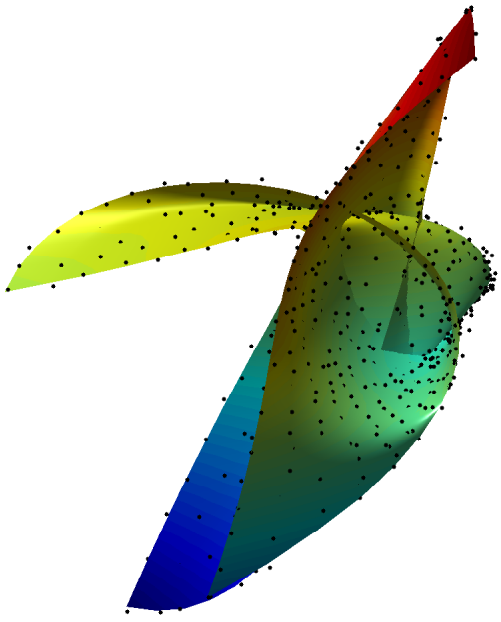


Figure 6.6: Random Bézier surface of degree (3,3).

Table 6.3 reports the main results for the best fitting Bézier surfaces of the three examples (arranged in rows). The following data are reported (in columns): optimal degree in u and v, mean and best value of n_{feval}, mean and best value of the

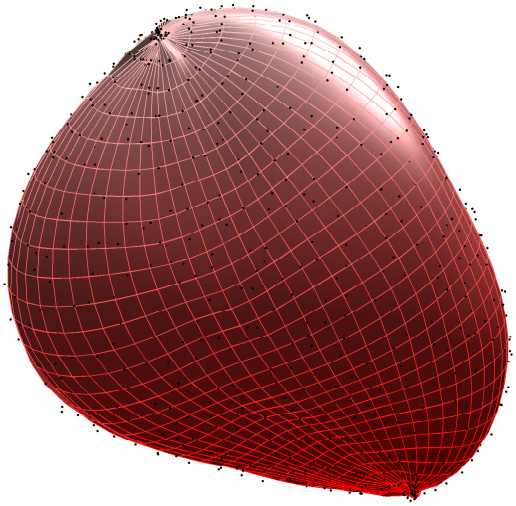


Figure 6.7: Heart shape: Bézier surface of degree (11,17).

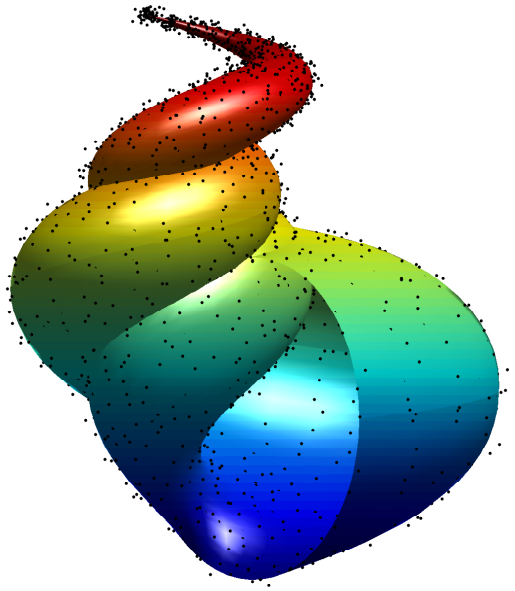
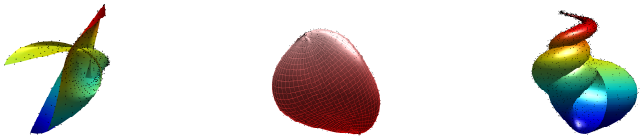


Figure 6.8: Seashell: Bézier surface of degree (30,30).



stat	mean	best	mean	best	mean	best
points	900		1891		3600	
noise	0.2		0.1		0.05	
x_mean	1.44e-5	7.49e-6	0.0052	0.0019	0.0029	0.0026
y_mean	1.29e-5	5.88e-6	0.0029	0.0013	0.0029	0.0026
z_mean	1.32e-5	5.65e-6	0.0055	0.0022	0.0029	0.0028

Table 6.2: Results for the random, the heart and the seashell surfaces examples.

	degree_u	degree_v	nfeval_mean	nfeval_best	mse_mean	mse_best	bic_mean	bic_best
	3	3	2871	2743	6.36e-07	1.30e-07	-3.7670e+04	-4.1957e+04
	11	17	11429	11308	0.2638	0.0172	-5942.8719	-16636.7217
	29	29	39520	35120	0.0892	0.0828	87.9482	-708.1069

Table 6.3: Results for the best fitting Bézier surfaces of the three examples.

mean square error (MSE), and the mean and best value of BIC. The reported values confirm the good behavior of the method and its resilience against noise of moderate intensity.

6.3 Rational Bézier curves

This section is devoted to present the results of *Two simulated annealing optimization schemas for rational Bézier curve fitting in the presence of noise*, a paper published in the journal *Mathematical Problems in Engineering* [Iglesias et al., 2016b]

6.3.1 Introduction

Fitting curves to noisy data points is a difficult problem arising in many scientific and industrial domains. Although polynomial functions are usually applied to this task, there are many shapes that cannot be properly fitted by using this approach. In this paper, we tackle this issue by using rational Bézier curves. This is a very difficult problem that requires computing four different sets of unknowns (data parameters, poles, weights, and the curve degree) strongly related to each other in a highly nonlinear way. This leads to a difficult continuous nonlinear optimization problem. In this paper, we propose two simulated annealing schemes (the all-in-one

schema and the sequential schema) to determine the data parameterization and the weights of the poles of the fitting curve. These schemas are combined with least-squares minimization and the Bayesian Information Criterion to calculate the poles and the optimal degree of the best fitting Bézier rational curve, respectively. We apply our methods to a benchmark of three carefully chosen examples of 2D and 3D noisy data points. Our experimental results show that this methodology (particularly, the sequential schema) outperforms previous polynomial-based approaches for our data fitting problem, even in the presence of noise of low-medium intensity.

6.3.2 The problem

A rational Bézier curve of degree n (as defined in chapter 2) can be expressed as:

$$\mathbf{s}(t) = \sum_{j=0}^{n} R_{j,n}\mathbf{b}_j \quad \text{with } t \in [0,1] \tag{6.6}$$

where $\mathbf{b}_j$ are the control points and $R_{j,n}$ are the rational blending functions with weights $\{w_j\}_{j=0}^{n}$ given by:

$$R_{j,n} = \frac{w_j B_{j,n}(t)}{\sum_{\ell=0}^{n} w_\ell B_{\ell,n}} \quad \text{with } B_{i,k}(t) = \binom{k}{i} t^i (1-t)^{k-i} \tag{6.7}$$

Let now $\{\mathbf{Q}\}_{i=1}^{M}$ be a set of data points in $\mathbb{R}^d$. The problem consists of obtaining the rational Bézier curve, $\mathbf{s}(t)$, of a certain degree n providing the best least-squares fitting of the data points. This leads to a minimization problem of the least-squares error E defined as the weighted sum of squares of the residuals:

$$\Theta = \sum_{i=1}^{M} \mu_i \left\| \mathbf{Q}_i - \frac{\sum_{j=0}^{n} w_j B_{j,n}(t)\,\mathbf{b}_j}{\sum_{\ell=0}^{n} w_\ell B_{\ell,n}} \right\|_2^2 \tag{6.8}$$

where μ_i are scalar numbers used in situations when it may be reasonable to assume that sampled data should not be treated equally. In order to reflect faithfully the most common situation in real-world problems, in this paper we will assume that no information about the problem is available beyond the data points. This means that all data points must be treated equally; that is, $\mu_i = 1$, for all i. Note, however, that our method is independent on the values of μ_i. To represent the geometrical distribution of the data we need to associate a parameter T_i for each input point $\mathbf{Q}_i$. Therefore, our goal is to obtain the three sets of parameters $\{t_i\}_{i=1}^{M}$, $\{w_j\}_{j=0}^{n}$, and $\{\mathbf{b}_j\}_{j=0}^{n}$. It is obvious that since each blending function in (6.7) is nonlinear in t, system (6.8) is also non-linear. As a consequence, we have to deal with a multivariate continuous nonlinear minimization problem. In this paper we solve this problem by applying two different schemas of the simulated annealing optimization method, which is described in the next section.

Note, that this functional 6.8 is an special case of the more general problem formulated in chapter 3. Therefore, once the weights and parameters are known, it can be solved by means of the Moore-Pensrose pseudo-inverse.

6.3.3 Method implementation

As discussed above, our problem consists of reconstructing the underlying shape of a given set of noisy data points by using a rational Bézier curve. This implies solving a nonlinear least-squares minimization problem while simultaneously minimizing the required number of free parameters. Solving this problem requires computing four different sets of unknowns: data parameters, poles, weights (represented in this section by vectors $\mathbf{p}$, $\mathbf{b}$, and $\mathbf{w}$, resp.), and the curve degree, n. Our approach to tackle this issue is a hybrid strategy combining classical methods (least-squares minimization), modern stochastic methods (simulated annealing), and information science metrics (Bayesian Information Criterion (BIC)).

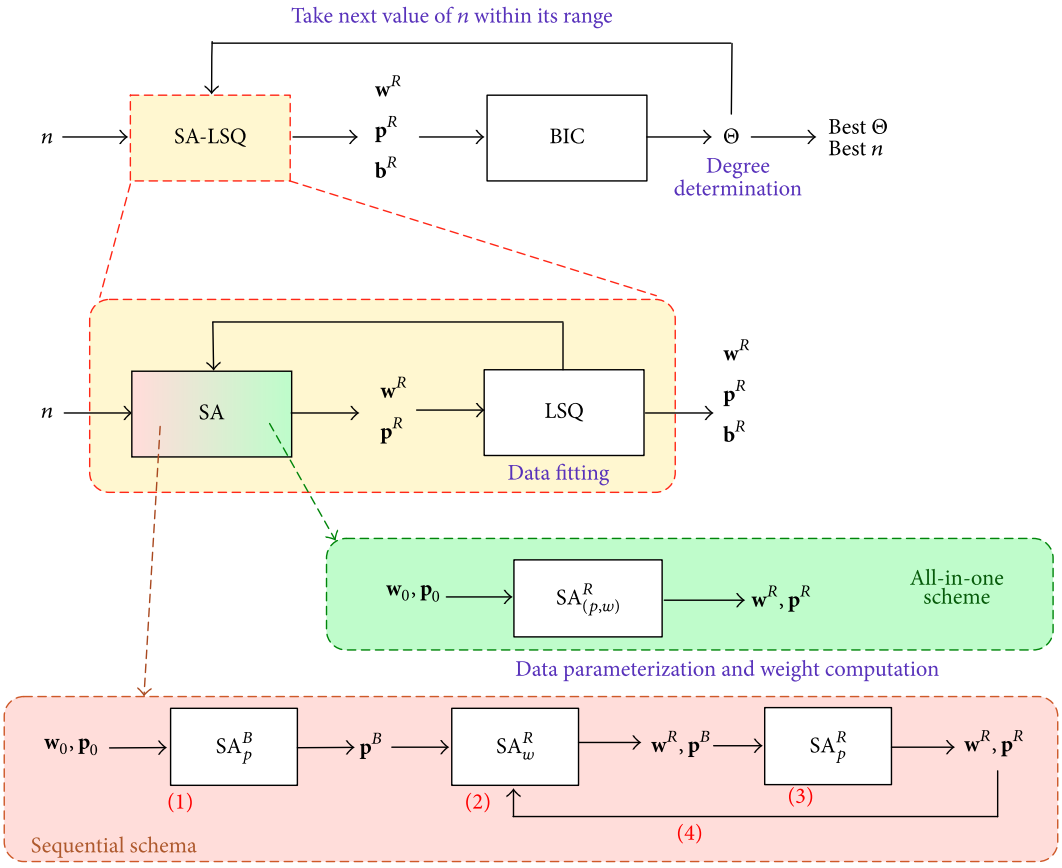


Figure 6.9: Workflow of the proposed method described in three layers (from upper to lower layer): the general workflow; decomposition of the SA-LSQ procedure; the two different SA schemas introduced in this paper: all-in-one schema (top) and sequential schema (bottom).

Before explaining how our method works, let us introduce the following notation: from now on, we will use the subindex $(\cdot)_w$ when searching for the curve weights, $(\cdot)_p$ when searching for the curve parameters, and $(\cdot)_{w,p}$ when searching for both, parameters and weights. Thus, SA_p stands for simulated annealing applied to parameter search, $f_{w,p}$ means the objective function with domain of definition $\mathbf{y} = (\mathbf{w}, \mathbf{p}) \in \mathcal{W} \times \mathcal{D} \subset \mathbb{R}^{n+1} \times [0,1]^M$, and so on. The superindex $(\cdot)^B$ denotes the search for a non-rational Bézier curve and $(\cdot)^R$ stands for a rational one. Hereafter, consider $\mathcal{W} = (0,100]^{n+1}$ and $\mathcal{D} = [0,1]^M$. Without loss of generality, we can also assume that $w_0 = w_n = 1$, $t_1 = 0$, and $t_m = 1$.

Figure 6.9 shows the main steps of our method. Basically, it consists of four major tasks: data parametrization, weight computation, data fitting, and degree determination. Upper part of this figure summarizes the method: we initially set a range for the curve degree n; then, for each value of this parameter n within that range, we apply a combination of simulated annealing and least-squares optimization (box SA-LSQ) to perform the first three tasks and compute the data parameters, weights, and poles of the best fitting rational Bézier curve for this value of n. Then, the BIC value of the resulting curve (corresponding to the last task, degree determination) is obtained (box BIC). At its turn, the SA-LSQ can be decomposed into two steps (middle layer of Figure 6.9): SA performs data parameterization and weight computation, while LSQ is used to compute the poles. This combination of SA and LSQ is repeated iteratively until no further improvement is reached. In this paper, we introduce two different SA schemas, shown graphically in the lower layer of Figure 6.9: the sequential schema and the all-in-one schema. They are explained in detail in the next section.

6.3.4 Simulated Annealing for rational Bézier curves

In this section we discuss our two Simulated annealing optimization schemata specifically tailored for data fitting with rational Bézier curves. The core of our SA implementation is based on the classical Simulated Annealing as explained in chapter 4 with the improvements exposed in sections 6.1 and 6.2 for non-rational Bézier curve and surface fitting, respectively. Thus, emphasis is placed on the new method discrepancies with respect the previous ones.

Simulated Annealing for data parameterization and weight computation

In this step we perform two different (but intertwined) tasks: data parameterization and weight computation. The former consists of obtaining a discrete association between the set of parameters $\{t_i\}_{i=1}^M$ and the noisy data points $\{\mathbf{Q}\}_{i=1}^M$ to be fitted, while the latter computes the weights. Both tasks are performed simultaneously by using the simulated annealing approach described in the previous section. The input for the SA method is given by the following:

I *Model complexity.* The curve degree, n.

II *Initial solution.* Initial random parameter vectors $\mathbf{p}_0$, $\mathbf{w}_0$, and $\mathbf{b}_0$.

III *Energy function.* The energy function, given by equation (6.8).

IV *Neighborhood function.* A next candidate generator function $\mathfrak{N}$.

V *Cooling schedule.* A cooling schedule $\mathfrak{S}$: how to decrease the SA control parameter.

VI *Stopping Criterion.* When to finish the algorithm (frozen state).

In this work, two different simulate annealing schemas are considered: the *sequential schema* and the *all-in-one schema*. Basically, the former calculates the different sets of unknowns of our optimization problem in a sequential way (i.e., only some parameters are initially computed and subsequently used to compute the remaining parameters), while the latter computes all unknowns at once. Let us analyze them in detail.

Using the notation introduced above, the sequential schema (SEQ) begins by finding the best non-rational Bézier fitting curve through SA_p^B, then it performs the following sequence iteratively:

$$\mathrm{SA}_w^R \rightarrow \mathrm{SA}_p^R \tag{6.9}$$

until there is no further improvement in the final solution of either SA procedure in comparison with the previous one. Each routine takes the preceding solution as an input parameter. The schema can be summarized as follows (see also the lower layer of Figure 6.9, where the numbers in red indicate the different steps of the algorithm):

(1) Apply the SA_p^B with a random initial guess $\mathbf{p}_0$ to find a non-rational Bézier curve that fits the data better. Let $\mathbf{p}^B$ be the resulting solution.

(2) Search for a rational Bézier curve through SA_w^R with a random initial guess $\mathbf{w}_0 \in \mathcal{W}$ and fixed parameters $(\mathbf{p}^B)$. Let $(\mathbf{w}^R, \mathbf{p}^B)$ be the resulting solution.

(3) Apply the SA_p^R optimization with $(\mathbf{p}^B)$ as the initial guess and fixed weights $\mathbf{w}^R$. Let $(\mathbf{w}^R, \mathbf{p}^R)$ be the resulting solution.

(4) Repeat (2)-(3) iteratively until there is no improvement in the resulting solution.

In general, the energy function for the simulated annealing procedures SA^R in steps (2) and (3) above is that of equation (6.8). However, a different energy function is required for the non-rational case (i.e. $w_j = 1$ for all j) in step (1), SA^B, given by:

$$\Theta = \sum_{i=1}^{M} \mu_i \left\| \mathbf{Q}_i - \sum_{j=0}^{n} B_{j,n} \mathbf{b}_j \right\|_2^2 \tag{6.10}$$

On the other hand, the *all-in-one schema* (AIO) corresponds to the minimization of functional

$$f_{w,p} : \mathcal{W} \times \mathcal{D} \rightarrow \mathbb{R} \tag{6.11}$$

where for each state vector $(\mathbf{w}, \mathbf{p})$ we compute $f(\mathbf{w}, \mathbf{p})$ as the least-squares solution of equation 6.10.

Neighbor function

The neighborhood function is one of the key components of the SA algorithm. Furthermore, its role becomes even more important for multi-modal optimization problems, where the objective function is of the *many local peaks surrounded by deep valleys* type. This is exactly what happens in this paper. There are several alternatives for the neighborhood function. A very popular choice is the *fast next candidate generator*, which builds the new solution by modifying the previous one in steps proportional to the system temperature:

$$\mathbf{x}_{new} = \mathbf{x}_{old} + T\mathbf{v} \tag{6.12}$$

where is a random vector holding $\|\mathbf{v}\|_2^2 = 1$. This is an approximation of FSA schema, one of the tested functions in section 6.2, but it required another support function (a local search method) in order to exploit the neighborhood of a solution. In this work we remove the supporting local method by maintaining two sets of controlling parameters: firstly, the global *real* temperature following the classical SA temperature. Then, the algorithm builds a second set of *virtual* temperatures (one for each spatial dimension) that are restarted at the beginning of each inner cycle. The resulting equation becomes

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \mathbf{T} \odot \mathbf{v} \tag{6.13}$$

where $\mathbf{T}$ has all its components set to the current system temperature at the beginning of each outer cycle. Thus, whenever $\mathbf{x}_{new}$ is accepted through the acceptance criterion, we compute the absolute difference between the *old* and *new* solutions. For each component that behaves better than the previous one in the comparison, we update the corresponding component on $\mathbf{T}$ according to the cooling schedule. The proposed method is based on the Adaptive Simulated Annealing (ASA), see section 4.2.2.

Cooling schedule.

By the cooling schedule we refer to a triplet $\mathfrak{S} = \{T_0, T, \mathfrak{T}\}$ accounting for the selection of the initial temperature, the temperature parameter, and the cooling function ($\mathfrak{T}$), respectively, along with the *thermal equilibrium criterion*. The cooling schedule governs the pace at which the temperature is updated during the execution of the SA. Therefore, its choice is of primary importance for the good performance of the algorithm. In this work we considered initially the power, Boltzmann, and fast

schedules. However, our computational experiments showed that the fast schedule provides more visually appealing results and runs faster than the other two alternatives. Since these are two particularly valuable features in the context of data fitting, we eventually selected the fast schedule as the best choice for rational Bézier curve fitting. The fast schedule is governed by the law $T_{new} \leftarrow {}^{T_0}/_k$, which provided a good balance among simplicity, speed, and good performance for other data fitting problems as has been exposed in our previous works (sections 6.1 and 6.2).

In order to select the initial temperature we have discarded the time-consuming restarting strategy used in our previous work in favor of the method presented in [Ben-Ameur, 2004]. It consists of setting an acceptance ratio, χ_0, and determining, through an iterative algorithm, a compatible starting temperature. Among the several possible ways to define χ_0, in this work we choose the classical one of the quotient between the number of bad transitions accepted and the attempted ones, already proposed in [Aarts et al., 1985]. According to this definition, we set $\chi_0 = 0.8$, a typical value in previous literature in the field. For further details on the proposed T_0 initialization schema see section 4.2.4, more specifically Algorithm 4.2.

Stopping criterion

Similar to the choice of the initial temperature, there is not a general set of stopping conditions suitable for all problems. There are, however, two common practices: the first one is to set a maximum number of iterations; the second one is to stop the execution of the algorithm when the system is *frozen*, that is, when no new solutions (either better or worse states) are accepted for a predefined number of iterations. Since the former can waste a lot of computation time with no further improvements on the solution, in practice the stopping criterion is often a combination of the two. In our implementation, the algorithm stops whenever one of the following conditions is met: either $n_{feval} > 1e^3 \cdot \xi$ or $\sharp\left[\partial\Delta_global\right] = 10 \cdot \xi$, where n_{feval} denotes the total number of evaluations of the energy function, ξ denotes the number of free parameters of the system, and $\partial(\cdot)$ denotes the lack of changes of the mean variation of the energy function.

Memory effect

We improve the memory capacities of the method through elitism: the best state from the current iteration is encoded as a vector $(\mathbf{x}^k_{best}, f^k_{best})$ and stored in a temporal buffer. Obviously, this *best so far* solution is updated whenever a better solution is achieved during SA execution. We remark however that this $(\mathbf{x}^k_{best}, f^k_{best})$ vector is not used to drive the SA execution; instead, it is only used as a memory effect, with the role to (possibly) improve the convergence rate with respect to the standard (non-elitist) version of SA.

Cast back operator

We add a new operator, related to the domain of the problem, to work in combination with the neighborhood function. This extra functionality checks whether a new generated solution goes outside the search domain of the problem and sends it back into the search space whenever it goes away. To this purpose, we apply the classical cast *back operator*, a widely accepted routine in numerical methods. Suppose that the SA returns a new solution $\mathbf{x}_{new}$ outside the problem domain $\mathcal{D}$, obtained from a previous solution $\mathbf{x}_{old}$ within $\mathcal{D}$. The cast back procedure replaces $\mathbf{x}_{new}$ by a new value $\mathbf{x}_{cb}$ given by the convex combination: $\mathbf{x}_{cb} = \alpha \cdot \text{Proj}(\mathbf{x}_{new}) + (1-\alpha)\mathbf{x}_{old}$, where $\text{Proj}(\cdot)$ is the operator that projects any point outside the domain onto its closest point on the boundary of $\mathcal{D}$ and $\alpha \ni U((0,1))$ is a uniform random number in the interval $(0,1)$. This procedure returns a new point that is well within the search domain while simultaneously ensuring that the probability of the boundary is not increased by this operator.

Adaptive thermal equilibrium

We improve the cooling schedule ($\mathfrak{S}$) with extra conditions for the thermal equilibrium. In our implementation, the inner cycle stops if the value of χ_0 is reached after ξ iterations.

6.3.5 Experimental results

In this section, we analyze the performance of our method by applying it to a benchmark of three illustrative examples of 2D and 3D noisy data points. These examples have been carefully chosen so that they exhibit challenging features such as self-intersections or strong changes of slopes and curvatures. The first two examples correspond to real-world instances so that we can replicate the usual conditions of real-world applications, including the presence of noise of low-medium intensity. The last one is an academic example designed to analyze the effect of different levels of noise on our method. For each example, we report the results of three different schemas used for comparative purposes: non-rational, rational *all-in-one*, and rational *sequential*. Finally, we analyze the robustness of the method in the presence of noise by comparing each schema against the same data perturbed with different levels of noise for the last example.

Regarding the implementation issues, all the experiments were run on a AMD-FX^{tm}-4100 Quad-Core Processor at 3600 *Mhz* with 8GB DDR3 RAM running *Linux 3.14.x LTS kernel* and *MATLAB 2012a*. For each dataset and schema, the experiment has been executed 26 times. Then, 20 executions are finally selected (after removing the three best and three worst executions) to provide statistical evidence for the results presented and assert the experiment reproducibility.

Table B.1 reports our results for each dataset and experiment. The following items are arranged (in columns): dataset examined, the type of curve reconstructed

(NR: non-rational; R: rational), the schema executed for rational curves (AIO: *all-in-one*; SEQ: *sequential*), the total number of calls to the energy function in the best case (represented by n_{feval}), the best and average BIC, the number of poles for the best BIC (represented by n_{pol}), and the relative mean error for each component (x_{mean}, y_{mean}, and z_{mean}). When used, the acronym N.A. stands for not applicable. The *logo*, *shoe*, and *torus* names refer to the *scanned logo*, *shoe profile*, and *curve on a torus* datasets described below, respectively. The number after *torus* refers to the signal-to-noise ratio (SNR) applied. For the non-rational examples, a local search was performed in order to refine the SA solution. Our results show the good performance of the method even in highly noisy situations, which are the common case in real-world applications.

A Scanned Logo

The first example corresponds to the shape of a digitally scanned logo. The dataset consists of a set of noisy 2D data points, represented by black symbols in Figure 2. The figure shows our experimental results for the three cases analyzed: non-rational case (top), the all-in-one rational case (middle), and the sequential rational case (bottom). The figures on the left show the reconstructed points, represented as red empty circles. On the right, the best fitting curve is displayed as a red solid line. This example has been chosen because it represents a common real-world scenario: a scanned figure with the typical noise introduced during the scanning process. In addition to the high-intensity noise (clearly visible in all instances of Figure 6.10), this shape is also challenging because it includes difficult geometric features, such as several self-intersections and strong changes of slope and curvature. This example was first reconstructed in section 6.1 by making use of (non-rational) Bézier curves, both results are presented here, side by side, to have a better grasp of our new methodology added value over the old one.

As the reader can see from the figures, the method is able to recover the general shape of the data points with good accuracy. This is a very remarkable result because the original data points are highly noisy. Best results correspond to the sequential rational schema, while the non-rational and the *AIO* rational schemas perform almost similarly in this case. This fact is clearly visible in the topmost loop of the figures in right column, best fitted through the sequential schema (bottom figure). These visual results are in good agreement with the numerical results reported in Table B.1. The best and average BIC for the sequential schema are approximately -770 and -701, respectively, while they are both -691 for the non-rational schema and -661 and -658 for the AIO rational schema. Note also that the three methods obtain the same optimal number of poles $n_{pol} = 13$. In other words, the good accuracy of our method is not at the expense of a very large number of free variables.

Shoe profile

The second example corresponds to a shoe profile obtained from a pressure-mechanical method without filtering, leading to a set of 400 three-dimensional noisy

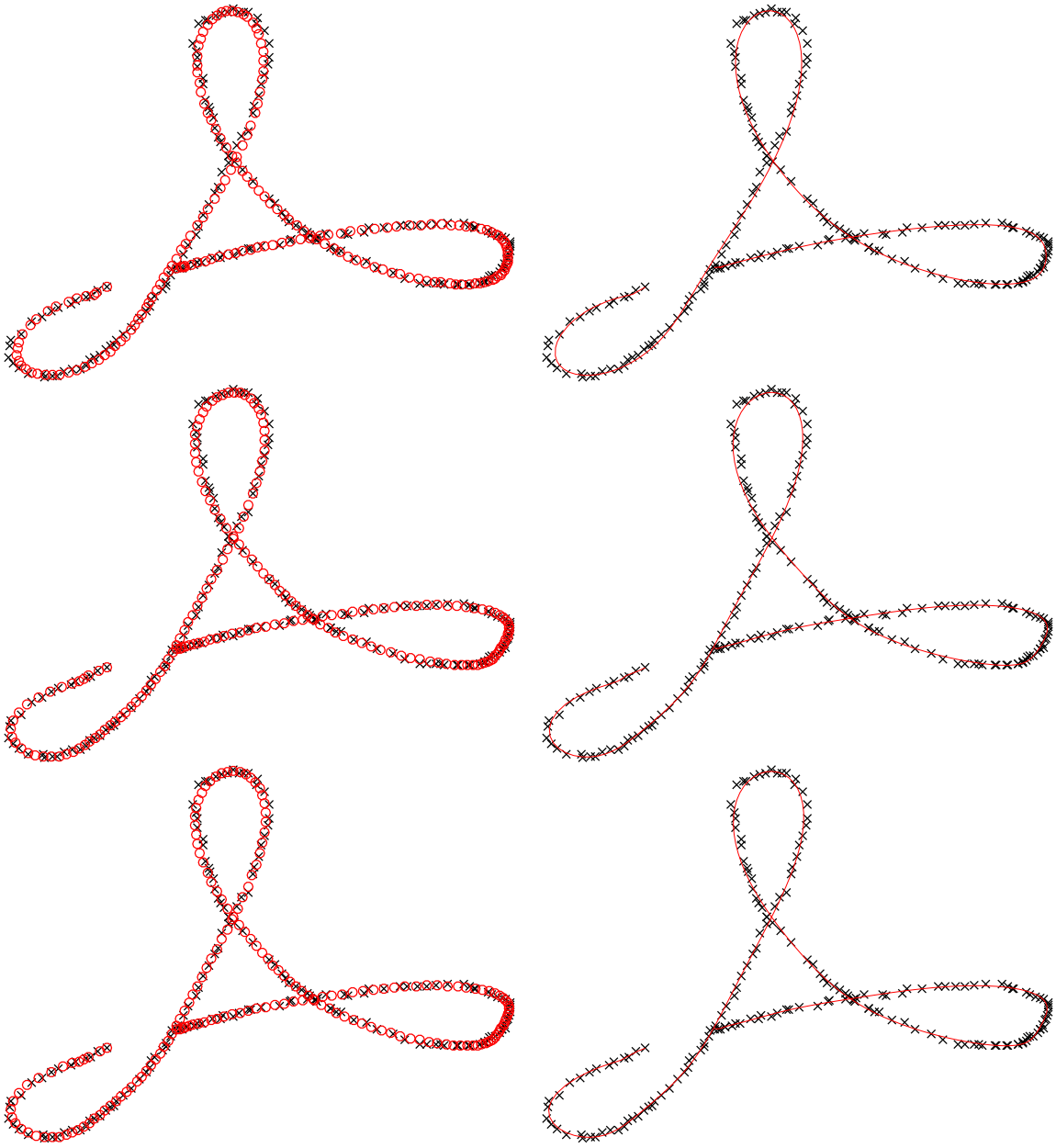


Figure 6.10: Experimental results for Example 1: left: reconstructed points (red circles); right: best fitting curve (red solid line); top: non-rational case; middle: *AIO* rational case; bottom: sequential rational case. In all cases, $n_{pol} = 13$.

data points. Figure 6.11 shows our experimental results. The interpretation of this figure is similar to that of previous example and, hence, it is omitted here to avoid redundant information. Once again, the best fitting is obtained with the sequential rational schema, although in this case, the visual and numerical results are closer for the two rational schemas and significantly worse for the non-rational one. Note also that we obtained a similar parameter value, $n_{pol} = 24$, for the number of poles in all cases.

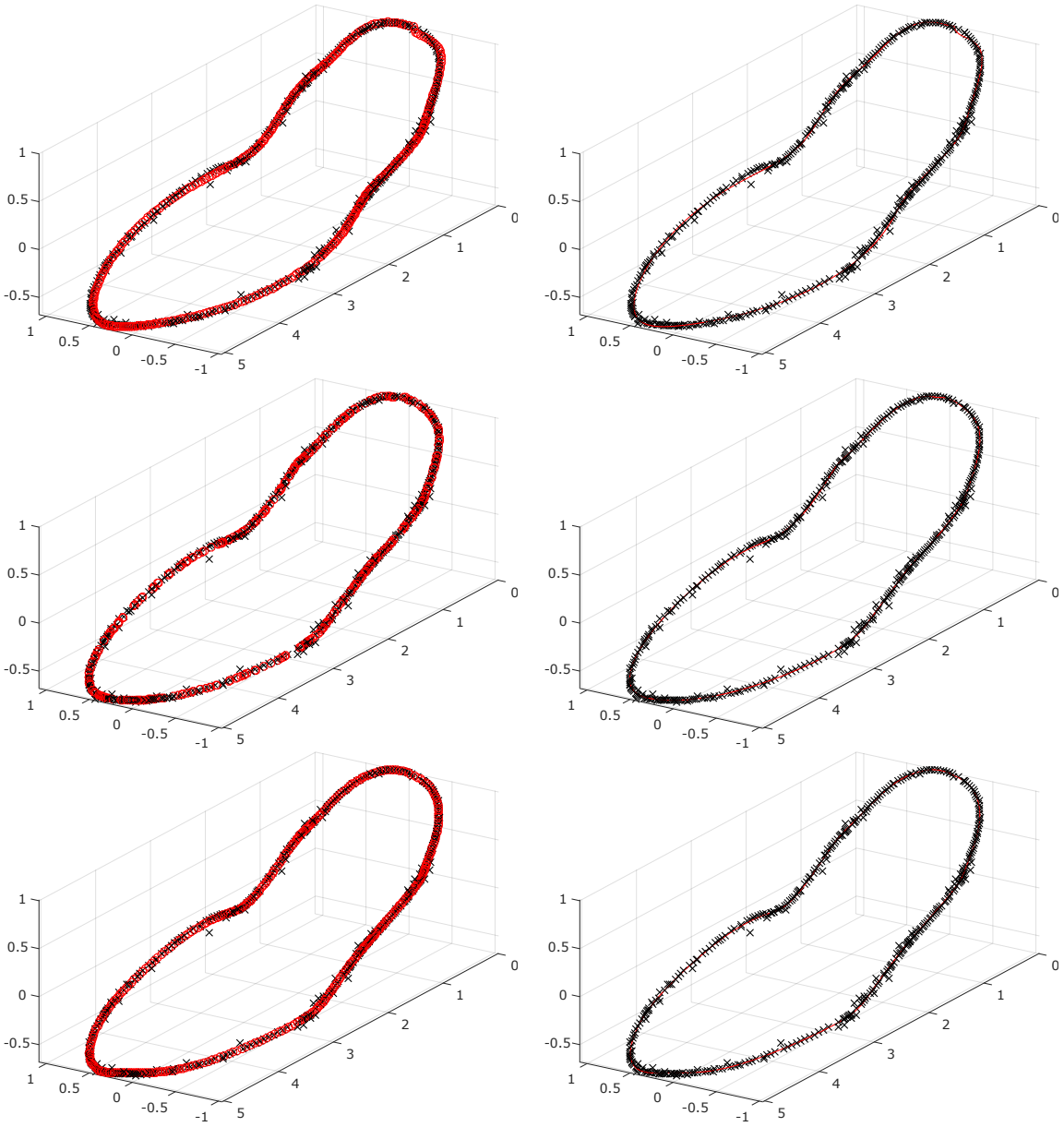


Figure 6.11: Experimental results for Example 2: left: reconstructed points (red circles); right: best fitting curve (red solid line); top: non-rational case; middle: AIO rational case; bottom: sequential rational case. In all cases, $n_{pol} = 24$.

Example 3: Curve over a Torus

Last instance in our benchmark corresponds to an academic example. It has been carefully designed to analyze the performance of our method against noise of different intensities. To this aim, we consider the parametric curve given by:

$$\begin{cases} x(t) = (a + b\cos(5t)\cos(2t) \\ y(t) = (a + b\cos(5t)\sin(2t) \qquad t \in [0, 2\pi] \\ z(t) = c\sin(5t) \end{cases} \tag{6.14}$$

which corresponds to a curve on a Torus. We consider a set of three-dimensional data points with uniform sampling in the interval domain $[0, 2\pi]$. This dataset, labeled as *torus* in Table B.1, is then perturbed with additive white noise of different intensities, modulated by a signal-to-noise ratio (SNR) ranging from 10 (very high intensity) to 90 (low intensity), with step-size 10. The corresponding datasets are labeled as *torusN*, where N indicates the SNR intensity. The simulation results with our method for the resulting 10 datasets are reported in the last 10 horizontal blocks of TableB.1. Some important observations can be obtained from the numerical data in the table. The most important one is that the sequential rational schema outperforms the others in terms of BIC value, meaning that it provides the best trade-off between accuracy and complexity for all instances in this example. According to our results, the optimal number of poles for this example is $n_{pol} = 20$ in all cases, except for the instance *torus10*, which corresponds to a case of very high noise intensity. In other words, this schema is able to capture the optimal number of poles in cases of noise of low and medium intensity. Furthermore, the method fits the data points very accurately. For instance, the relative fitting error for the noiseless case is as good as 10^{-6} for each coordinate. These striking results are visually confirmed in Figure 6.12. Note, for instance, the very good fitting for the sequential rational schema (bottom row).

On the other hand, as expected, the BIC increases as the noise level increases, meaning that the method is affected by the noise intensity, but not drastically. In fact, the method is very resilient against noise, as it still yields very reasonable relative fitting errors of order 10^{-2} for high-intensity noise (for instance, of SNR $= 10$) and 10^{-3} for SNR $= 30$. For example, the visual quality of the fitting is clearly visible for the case SNR $= 50$, as shown in Figure 6.13. We remark, however, that in this case, the non-rational and AIO rational schemas require extra parameters to obtain their best fitting. Note, for instance, that $n_{pol} = 25$ for the non-rational schema in this example. This effect can be explained by the fact that the non-rational curve has less degrees of freedom because no weights are available. As a consequence, more poles are usually required to compensate this limitation. But even in this case, the value of this parameter is lower or equal to 25. This result is a clear indication of the effectiveness of our proposal to use BIC to keep the dimension of the problem as low as possible and to prevent over-fitting.

This third example has also been used to illustrate the good performance of our neighborhood function, described during the Simulated Annealing implementation section. Figure 6.14 shows two graphical examples of the evolution of the BIC for the rational *all-in-one* schema *versus* the number of evaluations of the fitness function, given by the parameter n_{feval}. The pictures display the examples *torus* and *torus50* from Table B.1, corresponding, respectively, to the noiseless case (left) and the noisy case with SNR $= 50$ (right). Both pictures show the evolution of the maximum, mean, and minimum BIC in a color-coded representation (in blue, green, and red,

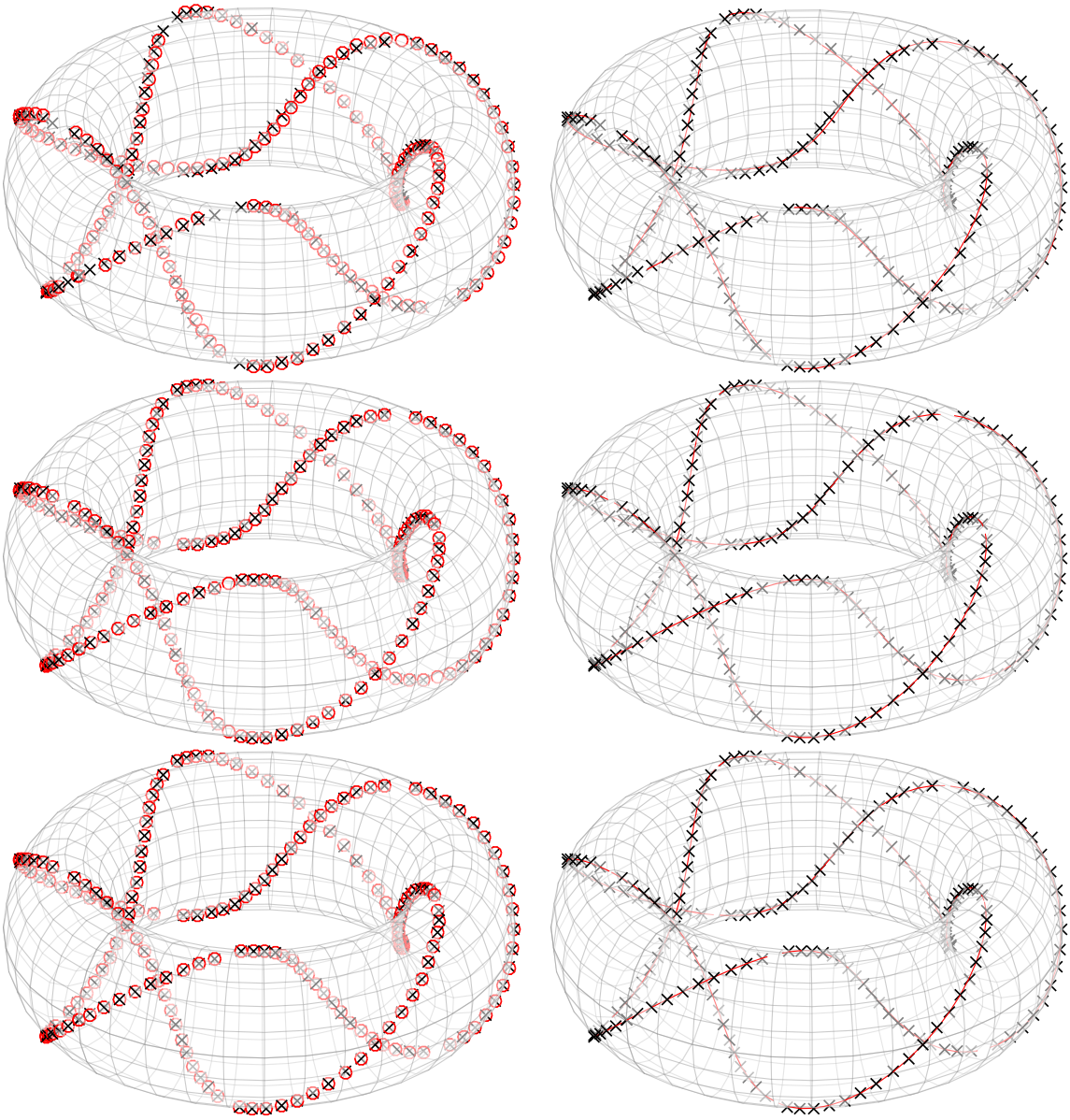


Figure 6.12: Experimental results for Example 3 (without noise): left: reconstructed points (red circles); right: best fitting curve (red solid line); top: non-rational case; middle: AIO rational case; bottom: sequential rational case.

resp.). These BIC values have been obtained with our method from 20 executions out of 26 executions after removing the three best and three worst results for each case. As the reader can see, our neighborhood function allows the method to escape from local minima, a situation that happens particularly at earlier stages of the evolution, associated with an intensive exploration of the search space. Two temporal windows have also been included in the pictures to enlarge these initial stages by zooming for better visualization. After this initial period, the BIC decreases slower and the fitting error reaches a plateau where the exploitation phase becomes dominant. Finally, convergence to the optimal values (marked by the vertical magenta line) is achieved

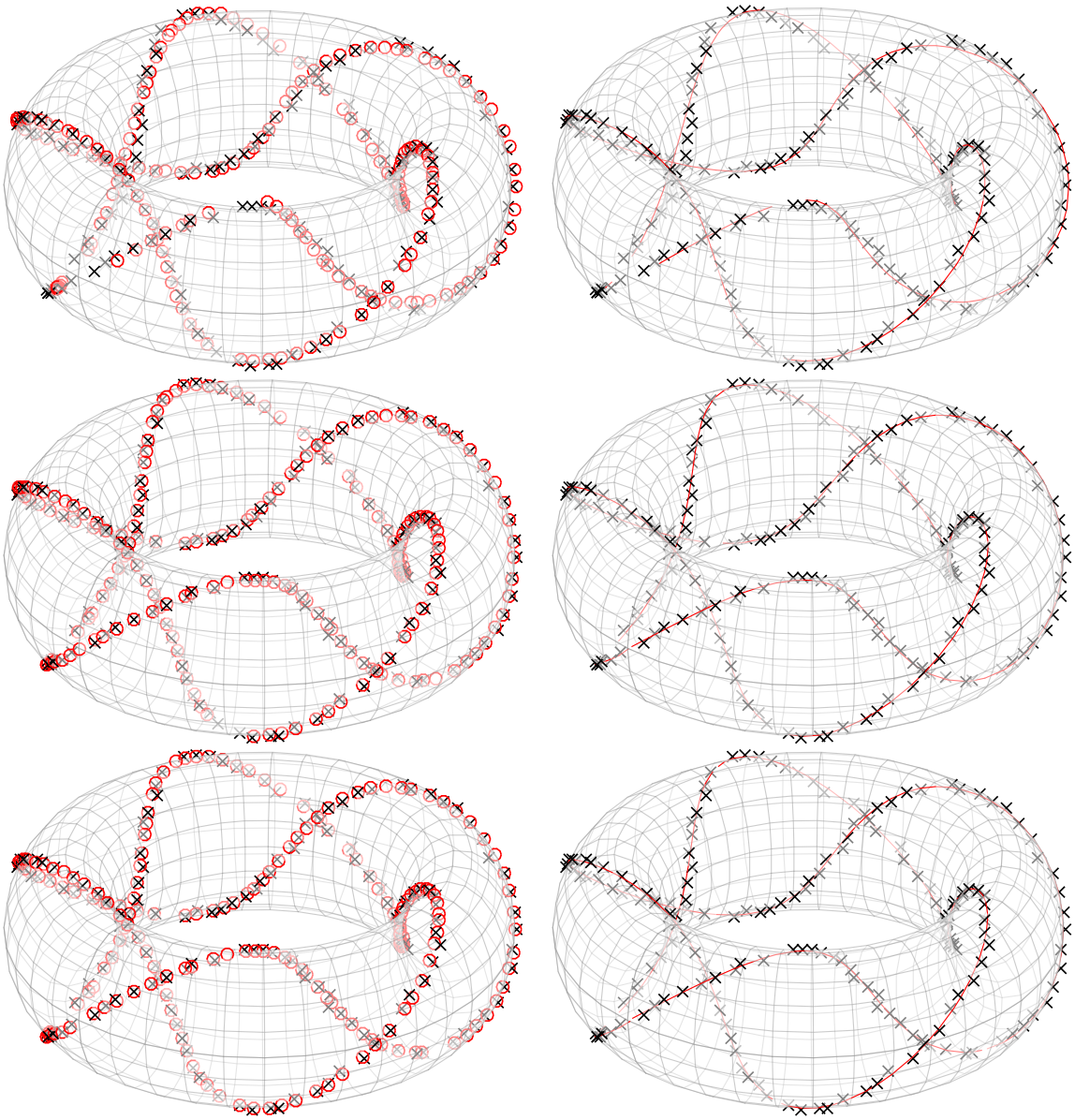


Figure 6.13: Experimental results for Example 3 (with SNR = 50): left: reconstructed points (red circles); right: best fitting curve (red solid line); top: non-rational case; middle: AIO rational case; bottom: sequential rational case.

and the fitting error does no longer improve. These pictures clearly show that the method is well suited for multi-modal problems, being able to escape from local minima, thus preventing premature convergence to happen.

The main limitations of our approach concern its performance in situations of high-intensity noise. Figure 6.15 shows our results for the third example with noise of SNR = 10. In general, our method is able to capture the tendency of the data even under these strongly adverse conditions, but some problems may arise in the neighborhood of the initial and last poles of closed curves. In particular, the continuity of such curves at that point cannot be assured. This situation is not critical at all; it can

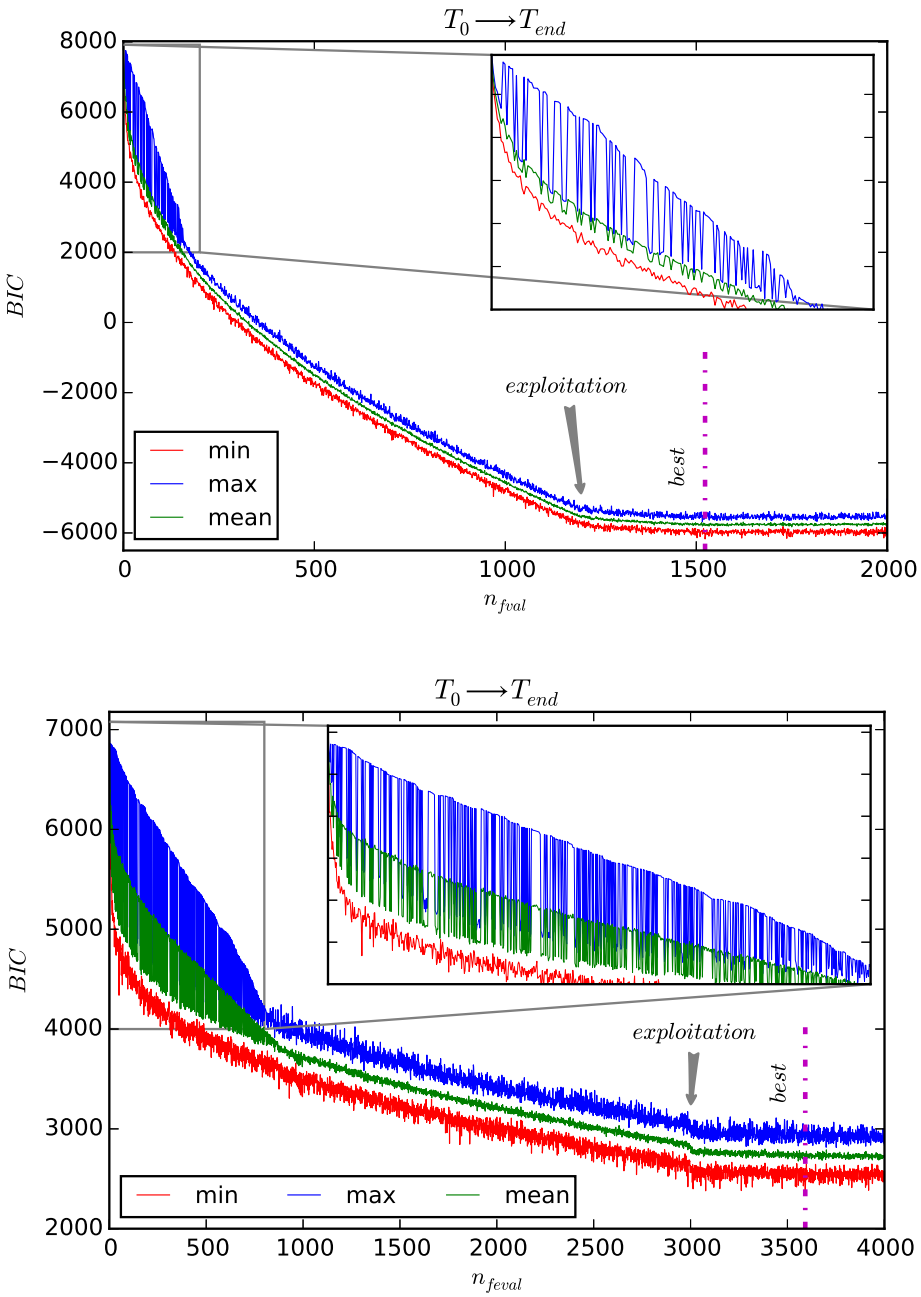


Figure 6.14: Evolution of the maximum, mean, and minimum value of BIC (in blue, green, and red color, resp.) versus the number of function evaluations for the curve on a torus example: top: noiseless case; bottom: noisy case (with SNR $= 50$). The inner boxed pictures also show a zoom of the initial exploration stages in both cases for better visualization.

readily be avoided by introducing additional constraints in our problem. However, as expected, the performance of the method is affected by the noise intensity, meaning

that some kind of preprocessing (such as filtering) might be advisable in highly noisy environments for real-world applications.

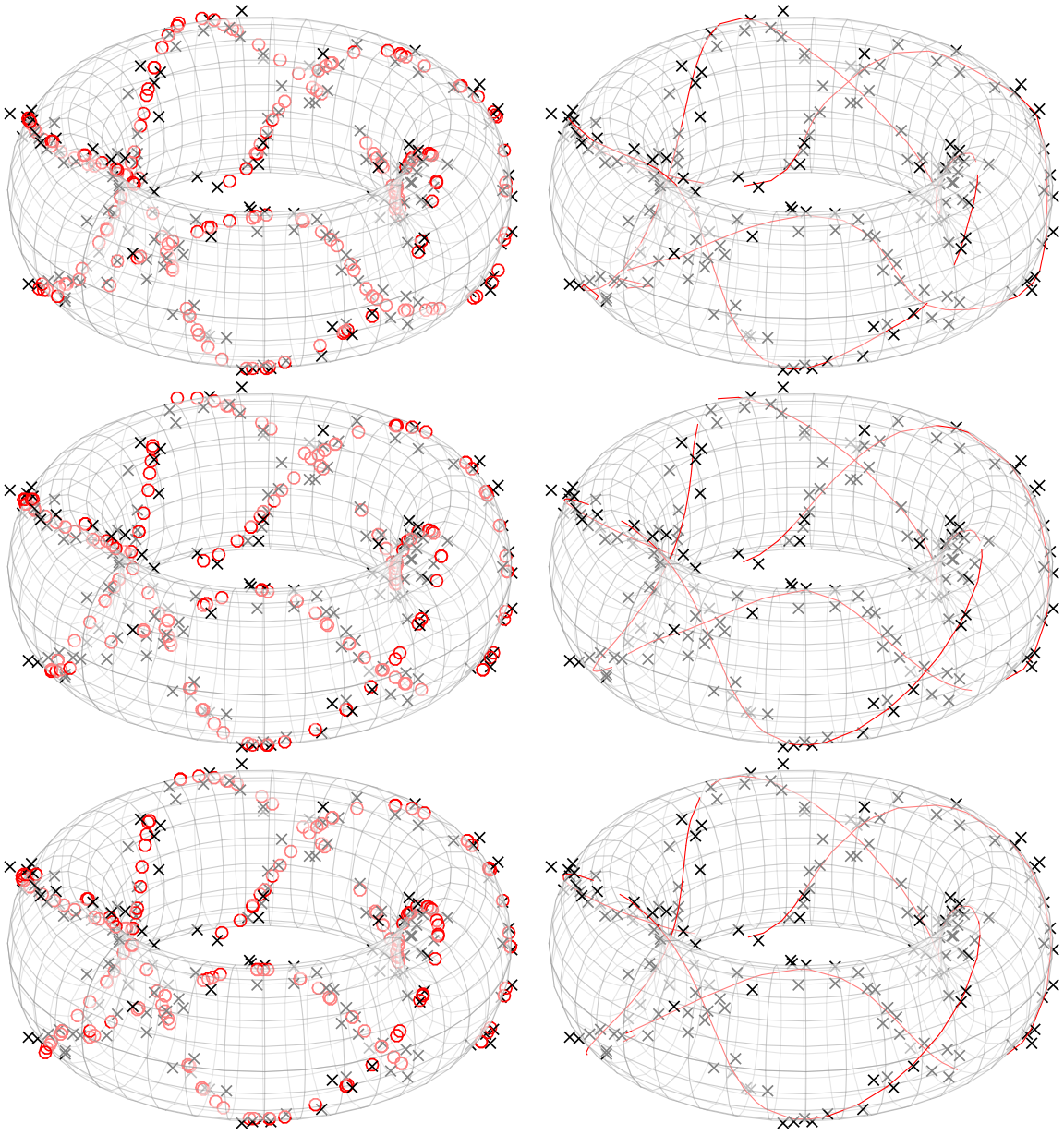


Figure 6.15: Experimental results for Example 3 (with SNR = 10): left: reconstructed points (red circles); right: best fitting curve (red solid line); top: non-rational case; middle: AIO rational case; bottom: sequential rational case.

6.4 Rational Bézier Surfaces

In this section we reproduce the results presented in the paper *Simulated Annealing and Natural Neighbor for Rational Bézier Surface Reconstruction from Scattered*

Data Points during the *International Conference on Harmony Search Algorithm* (ICHSA 2017) [Loucera et al., 2017b].

6.4.1 Introduction

Surface reconstruction is a very important problem in fields such as geometric modeling and processing, and CAD/CAM. Most of the methods proposed to solve this problem rely on parametric polynomial schemes. However, there are shapes that cannot be described by using a strictly polynomial approach. In this paper we introduce a new method to address the surface reconstruction problem from scattered data points through rational Bézier surfaces. Our approach is based on the combination of Simulated Annealing, the natural neighbor interpolation method, and least-squares minimization to perform data parameterization, data fitting, and weight computation. Some computer experiments carried out for both organized and unorganized data sets show the good performance of our approach.

6.4.2 The problem

Let $\left\{\mathbf{Q}_{p,q}\right\}_{p=0,q=0}^{N,M}$ be a given set of data points in $\mathbb{R}^3$. We seek to find the rational Bézier surface that approximates the given data better in the least-squares sense. This means that, for a given degree (m,n), the problem consists of finding the rational Bézier surface $\mathbf{S}$ given by equation (2.10) that minimizes the following least-squares functional E:

$$E = \sum_{p=0}^{N}\sum_{q=0}^{M}\left\|\mathbf{Q}_{p,q} - \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}B_{i,m}(u_p)B_{j,n}(v_q)\mathbf{P}_{i,j}}{\sum_{k=0}^{m}\sum_{l=0}^{n} w_{k,l}B_{k,m}(u_p)B_{l,n}(v_q)}\right\|_2^2 \tag{6.15}$$

which closely follows the general problem presented in chapter 3.

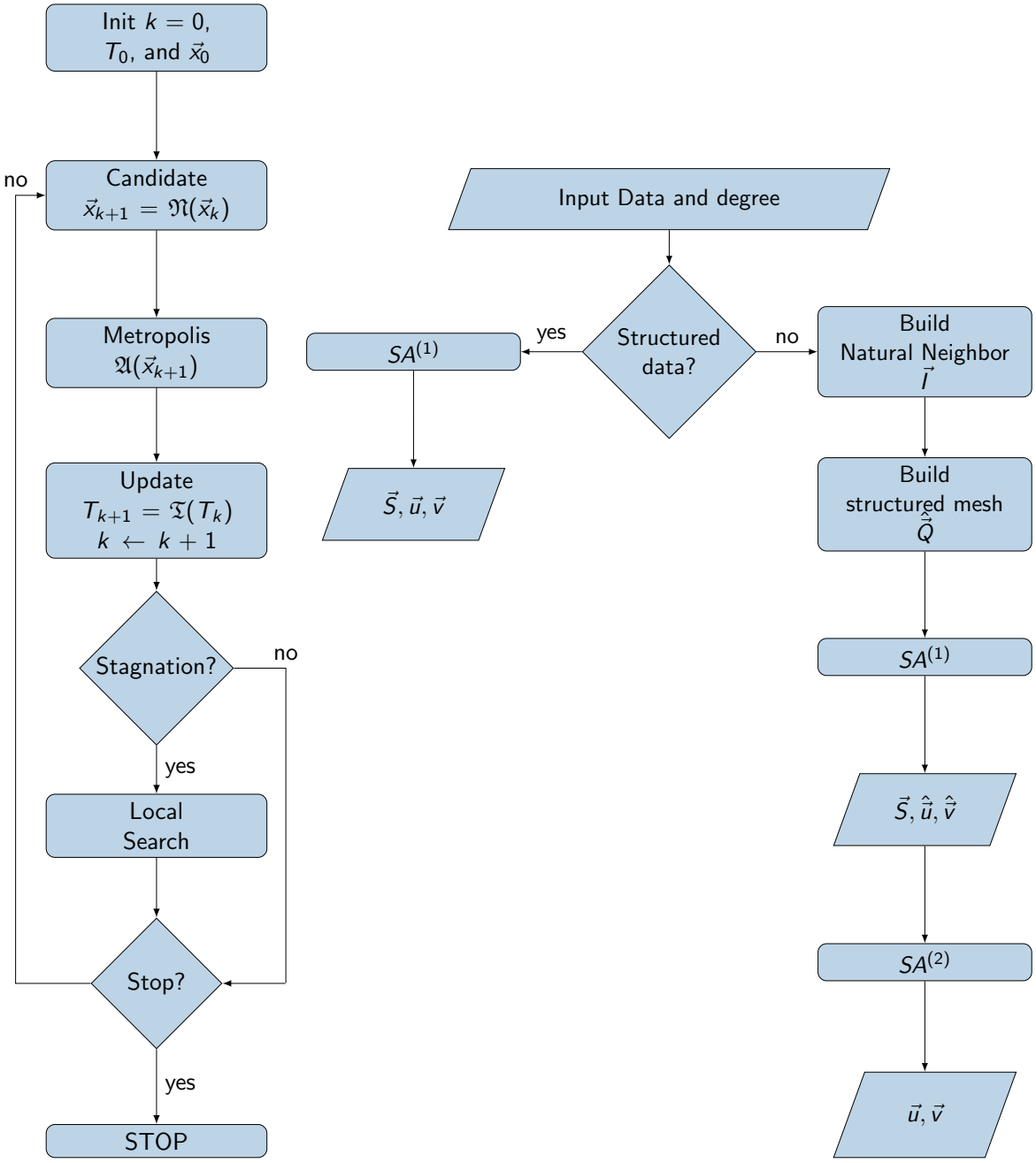
In the case of scattered data $\{\mathbf{Q}_\mu\}_{\mu=0}^{M}$, i.e. the topology of the point cloud is unknown, we handle the case a similar way. The equation to be minimized is given by:

$$E = \sum_{\mu=0}^{M}\left\|\mathbf{Q}_\mu - \mathbf{S}\left(u_\mu, v_\mu\right)\right\|_2^2 \tag{6.16}$$

6.4.3 Method implementation

The implementation of our algorithm for selecting the optimal rational Bézier surface is illustrated in Figure 6.16b. For the rest of this section, η represents the number of free variables of the system. In our method, two cases are considered:

I *Organized data.* If the data is structured the Simulated Annealing performs all required computations, namely: the surface parameters and weights via the SA neighborhood function and the control points by solving equation (6.15).



(a) SA Flowchart. **(b)** Method Flowchart.

Figure 6.16: Methodology flowchart for rational Bézier surface reconstruction.

Thus SA process is referred to as $SA^{(1)}$. The solution encoding for the $SA^{(1)}$ algorithm is given by column-wise stacking the following vectors $\mathbf{u}, \mathbf{v}$ and $\mathbf{w}$ where $\mathbf{w} = \text{vec}\left(\left(\{w_{i,j}\}\right)^T\right)$.

II *Scattered data.* When dealing with unorganized data $\{\mathbf{Q}_\mu\}_{\mu=0}^{M}$, we first compute the natural neighbor interpolant $\mathbf{I}$ of the data, see chapter 5. Then, evaluate it at an evenly spaced mesh which generates an structured set of points $\left\{\mathbf{I}\left(\tilde{u}_{\tilde{p}}, \tilde{v}_{\tilde{q}}\right) = \mathbf{Q}_{\tilde{p},\tilde{q}}\right\}$. Now we can approximate $\left\{\mathbf{Q}_{\tilde{p},\tilde{q}}\right\}$ using the previously outlines methodology, i.e. solving equation (6.15) to obtain a rational Bézier surface $\mathbf{S}\left(\tilde{u}_{\tilde{p}}, \tilde{v}_{\tilde{q}}\right)$. Finally, we find the associated parameterization (u_μ, v_μ) to the original data $\mathbf{Q}_\mu$ by minimizing equation (6.16) by means of the simulated annealing process referred as $SA^{(2)}$. note, however, that the only unknowns in (6.16) are the parameters (u_μ, v_μ). The solution encoding for the $SA^{(2)}$ process is given by the column-wise stacking of vectors $\{u_\mu\}_\mu$ and $\{v_\mu\}_\mu$.

In all cases the model selection step is done via the Bayesian Information Criterion BIC.

6.4.4 Simulated Annealing implementation

Let $f : \mathbb{R}^d \to \mathbb{R}^+$ be a real-valued function. The algorithm starts with a randomly chosen state $\mathbf{x}_0 \in \mathbb{R}^d, f_0 \equiv f(x0)$, and an initial temperature $T_0 \in \mathbb{R}^+$ Then, at each iteration k, a new candidate is generated via a neighborhood function $\mathfrak{N} : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^+$ by taking into account the system temperature, the previous candidate, and the solution space intrinsic characteristics. Each new proposed state is accepted or rejected in accordance with a modified Metropolis criterion [Metropolis et al., 1953]: a better proposal is always accepted; a worse one is only accepted with a probability that depends on the system temperature, and the energy transition between the previous and current state. Then, the temperature is reduced by a monotonically decreasing function $\mathfrak{T} : \mathbb{R}+ \to \mathbb{R}^+$ and a check for stagnation is made, i.e. to see if the average change in the objective function after N_s iterations is below a threshold ε_s. In case of stagnation, the algorithm triggers a local search with the current candidate as an initial guess. In either case, if the stopping criterion is not met, a new candidate is generated, effectively restarting the annealing cycle. A flowchart for our Simulated Annealing implementations for real-valued optimization problems is shown in Figure 6.16a. The main components of the algorithm are:

I *Temperature reduction.* $\mathfrak{T} : T_{k+1} \leftarrow \frac{T_0}{k}$.

II *Neighbor function.* $\mathfrak{N} : \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \Delta\mathbf{x}_k$, where $\Delta\mathbf{x}_k$ is a random variable sampled from the Cauchy temperature-driven distribution given by equation (4.6), thus:

$$\Delta\mathbf{x}_k \ni \frac{T_k}{\left(\|\Delta\mathbf{x}_k\|^2 + T_k^2\right)^{\frac{d+1}{2}}}$$

where d is the dimension of the search space.

III *Acceptance function.* $\mathfrak{A} = \min\left\{1, \left(1 + \exp\left(\frac{\Delta f}{T_k^{ac}}\right)\right)^{-1}\right\}$

Parameter tuning. Regarding the choice of values for the parameters of our method, they have been chosen as follows:

I *Initial temperature.* The initial temperature $T0$ has been selected as: $T_0 = \max(1, 0.8f^*)$, where f^* is the maximum distance between the evaluation of 100 random points.

II *Stagnation.* We set $N_s = 10\eta$ and $\varepsilon = 10^{-4}$.

III *Stopping Criterion.* An iteration budget of $N_{end} = 10^4$ is considered.

IV *Local search.* We refine each candidate by means of the Nelder-Mead simplex optimization algorithm [Nelder and Mead, 1965] as implemented in the `NLopt` library [Johnson, 2010].

6.4.5 Experimental results

Our methodology for surface fitting has been tested against three different datasets providing a broad range of challenging features for surface reconstruction. Our experiments were run on a AMD-FX^{tm}-4100 Quad-Core Processor at 3600 *Mhz* with 8GB DDR3 RAM running *Linux 3.14 LTS kernel* and *MATLAB 2012a*. Each experiment has been executed 16 times; then, we removed the three best and three worst executions in order to provide statistical evidence for the results presented and assert the experiment reproducibility. Each example is reconstructed for surfaces of degrees (m, n) with $m, n \in \{3, \ldots, 20\}$. In this work we show only the best results according to their BIC value.

Franke's test function. First example is constructed by evaluating the Franke's bivariate test function, given by:

$$\frac{3\mathrm{e}^{-\frac{9y}{10}-\frac{(9x+1)^2}{49}-\frac{1}{10}}}{4} - \frac{\mathrm{e}^{-(9x-4)^2-(9y-7)^2}}{5} + \frac{3\mathrm{e}^{-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}}}{4} + \frac{\mathrm{e}^{-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}}}{2} \tag{6.17}$$

at an evenly spaced grid of size 51×51, i.e. 2601 points. Then, a Gaussian noise of intensity 0.03 is applied to every point and 100 points are randomly removed. Finally we permute 100 randomly chosen points. As a consequence, the resulting point cloud is noisy, dense, and unorganized. But even under these very adverse conditions, our method is able to reconstruct the underlying shape with high fidelity as shown in Figure 6.17. The degree of the resulting surface is $(6, 5)$ with an RMSE (root-mean-square error) of 0.0162.

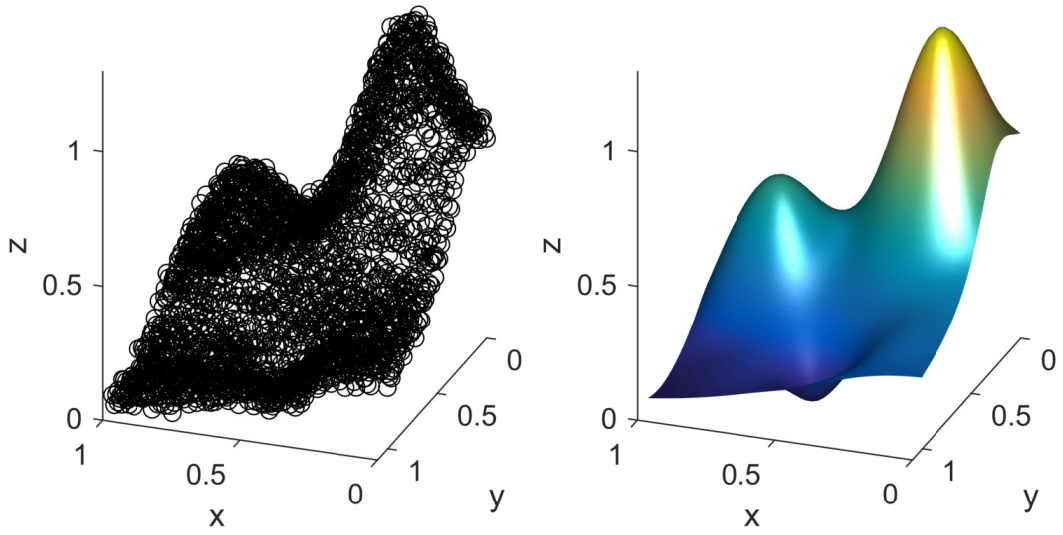


Figure 6.17: A high density and noisy point cloud (Franke function) reconstructed with a rational Bézier surface.

Pipe elbow. This example uses a data point set coming from a NURBS surface, so it is also a good test for our method. It consists of 10,000 points generated from a NURBS surface of degree $(4, 4)$ with three and four free knots, evaluated at an evenly spaced mesh of 100×100 points. Since the data is structured, we apply the $SA^{(1)}$ process to compute the data parametrization, the weights, and the control points. The best reconstructed rational Bézier surface is shown in Figure 6.18. The surface degree is $(7, 9)$: as expected from the original NURBS surface, the degree is higher on the v direction. The fitting RMSE value of the reconstructed surface is 0.0040.

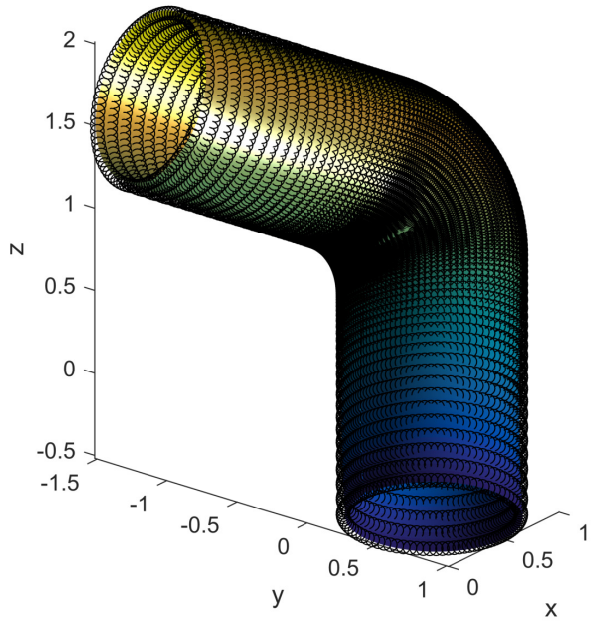


Figure 6.18: A NURBS surface reconstructed with a rational Bézier surface.

Big Sur Data. The final example corresponds to the Big Sur dataset [Foley, 1987], a point cloud generated by taking water temperature measurements from a boat. This low density (64 points), rapid varying and unorganized dataset poses a challenge for most interpolation and approximation methods. Our method automatically reconstructs the underlying shape without the need of any user input or subjective parameter. Because we are dealing with a very low density dataset where we typically require the interpolation/approximation to be extremely accurate at the vertical component, we have computed the normalized mean square error, and we got $NMSE_x = 0.85, NMSE_y = 0.83, NMSE_z = 0.96$. Note that this metric vary between $-\infty$ (bad fit) and 1 (perfect fit). The best reconstructed rational Bézier surface is displayed in Figure 6.19.

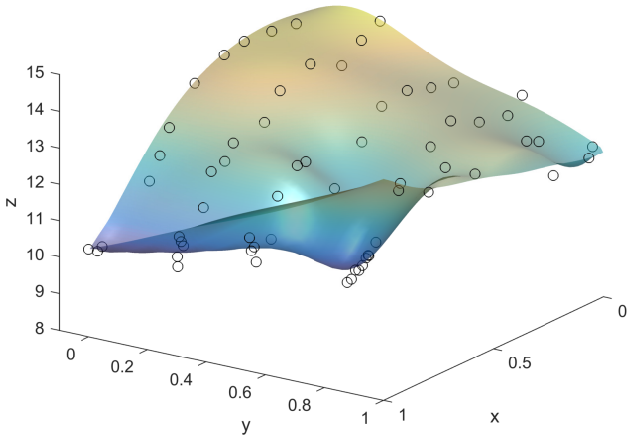


Figure 6.19: The Big Sur dataset reconstructed with a rational Bézier surface.

Chapter 7
Curve and surface fitting with B-spline models

> Nobody Understand Rational B-spline Surfaces, but see the possibilities.
>
> Some engineer at Boeing

In this chapter we present the results for data fitting by means of B-spline models. We start with the explicit case, continue with parametric curves and conclude with rational B-spline surfaces.

7.1 Free-knot splines

This section is devoted to the presentation of the results of a yet to be published paper on data fitting with free-knot splines. The *free-knot spline* terminology comes from the regression analysis [Marsh and Cormier, 2001] and approximation theory [Rice and Saloin, 1969, Cox, 1990] fields of expertise.

7.1.1 Introduction

Data fitting through free-knot splines has been extensively studied for its numerous applications in CAD/CAM, medical imaging, ship building, virtual reality and many more fields. When the number and locations of knots are treated as free parameters, the resulting problem is highly non-linear and multi-modal, thus very difficult to optimize. In this work we have developed an optimization framework to automatically compute the knots and its number by coupling a global optimization meta-heuristic, a Simulated Annealing schema paired with local optimizers, and the Bayesian Information Criterion for model selection. The resulting method can identify the general shape of the underlying function of the data, even in the presence of noise, and is able to reconstruct its critical points by computing truly identical knots. The results of numerical experiments are presented for a variety of shapes.

7.1.2 The problem

Let $\{\mathbf{Q}\}_{i=1}^{N}$ be a set of data points in $\mathbb{R}^d$. The problem consists of obtaining a free-knot spline, $\mathbf{s}(t)$, that fits the data and provides a good trade-off between fidelity and complexity.

We assume that $d = 2$ and that the data to be fitted can be written in the following functional form:

$$y_i = F(t_i) + \epsilon_i \quad (i = 1, \ldots, N) \tag{7.1}$$

where $F(t)$ is the underlying (unknown) function of the data and ϵ_i is the measurement error.

We fit the data given by equation (7.1) with a free-knot spline of degree k defined as per equation 2.12. Let E be the sum of the squares of the residuals:

$$E = \sum_{i=1}^{N} \mu_i \left\| y_i - \sum_{j=0}^{n} N_{j,k}(t_i) P_j \right\|_2^2 \tag{7.2}$$

where μ_i are weights associated to the data used in situations when the measurements are uncorrelated but have different uncertainties.

As it has been explained in chapter 3, equation (7.2) describes a multivariate continuous highly non-linear problem, known to be non-convex and multi-modal [Jupp, 1978, Laurent-Gengoux and Mekhilef, 1993].

7.1.3 Method implementation

By means of equation (7.2) we have transformed the geometrical problem of fitting a spline to the data into an optimization problem. This data fitting technique assumes a given number of knots and poles (the free variables). As this quantity increases so does the resulting B-spline complexity; even if the fitting error decreases, it is desirable to minimize this complexity, as the resulting curve would be easier to manipulate (and given the non-linearity, it would also be more numerically stable). To compute the resulting model complexity we make use of the Bayesian information criterion (BIC) as presented in equation (5.2).

To summarize, for each p in a given range, find the optimal fitting B-spline of degree k and p free (inner) knots, by making use of the provided optimizer (introduced below). Then, compute the resulting model transformed fitness by means of the BIC. Finally, we obtain the optimum number of knots, and the control points as a side effect, by selecting the model with the smallest BIC.

7.1.4 Heuristic Pattern Search with Cauchy Annealing

The *Fast Simulated Annealing* (FSA) [Szu and Hartley, 1987] and the *Simulated Annealing Heuristic Pattern Search* (SAHPS) [Hedar and Fukushima, 2004] algorithms provide the foundation of our simulated annealing implementation.

For readability, let D denote the dimension of the solution space, $f : (0,1)^D \to \mathbb{R}^+$ the function to minimize, $\mathbf{x}$ one of such solutions and $k, k+1$ the previous and current iteration, respectively. The algorithm presentation is organized as follows: first, we introduce de main ideas behind the algorithm, then each core idea is described with more detail, and finally, we put all the concepts together.

The execution begins with a randomly chosen solution. In order to freely explore the search space at the initial stages, a set of very high temperatures are paired with an acceptance function that initially rewards hill climbing transitions. At each iteration, a new solution is generated from the previous one, in accordance to a Cauchy distribution that samples a deviation from the latter. This sampling technique depends on the general temperature and allows a good compromise between local and global exploration, when paired with and acceptance temperature that decreases at a slower rate than the general temperature. After a certain number of iterations, the algorithm checks if the system has stagnated, i.e. no significant improvement has been seen in the optimum. A local exploration phase begins when the system falls into stagnation, and an instance of the local optimizer is launched with the current solution as initial guess. A tabu list is maintained in order to minimize the time wasted when visiting previously visited solutions, a solution is appended to the tabu list when it has been locally exploited at least once. After each iteration the stopping criterion is checked in order to finalize the algorithm.

The neighborhood generation function. The next candidate point $\mathbf{x}_{k+1}$ is generated by slightly perturbing the previous one, $\mathbf{x}_k$, as stated in (7.3)

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k \tag{7.3}$$

where $\Delta\mathbf{x}$ is a random variable sampled from the following Cauchy distribution (at generation temperature T_k^{gen}):

$$g_k(\Delta\mathbf{x}) = \frac{T_k}{\left(\|\Delta\mathbf{x}\|^2 + T_k^2\right)^{(D+1)/2}} \tag{7.4}$$

Note that it is just equation 4.6 reprinted here for clarity purposes. By combining together equations (7.3) and (7.4) we obtain the *neighborhood* function, $\mathfrak{N}$, which is clearly non-deterministic. In essence, we are sampling a deviation $\Delta\mathbf{x}_k$ from the previous point $\mathbf{x}_k$ taking by into account the current temperature. Therefore, the proposed visiting distribution provides a good compromise between global and local exploration [Szu and Hartley, 1987, Locatelli, 2000].

Cooling schedule. In order to guarantee an appropriated acceptance ratio, understood as the proportion between the proposed solutions and the accepted ones, we make use of two different (artificial) control parameters, namely the *acceptance* (T^{ac}) and *global* (T^{gen}) temperatures:

$$\begin{aligned} T_{k+1}^{ac} &= \frac{T_0^{ac}}{\log(k+1)} \\ T_{k+1}^{gen} &= \frac{T_0^{gen}}{k+1} \end{aligned} \tag{7.5}$$

where log indicates the natural logarithm.

Acceptance criterion. In order to decide if a solution is accepted we use the classical acceptance function derived from the Metropolis-Hastings [Metropolis et al., 1953] sampling algorithm:

$$\mathfrak{A} = \min\left\{1, \left(1 + \exp\left(\frac{\Delta f}{T_k^{ac}}\right)\right)^{-1}\right\} \tag{7.6}$$

This acceptance function has been carefully explained in chapter 4.

Local optimizers. For the sake of completeness we include here the Heuristic Pattern Search HPS and Approximate Descent Direction ADD algorithms. Together, this heuristic processes build the local search procedure, see [Hedar and Fukushima, 2004] for a deeper discussion.

I *Initialization.* Fix an initial mesh size $\Delta_0 > 0$, the shrinkage coefficient $\sigma \in (0,1)$ and the maximum number of iterations m_h. Set $k = 0$ and the first trial point x_0.

II *ADD.* Set $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \Delta_k \mathbf{v}$ if $f(\mathbf{x}_k + \Delta_k \mathbf{v}) < f(\mathbf{x}_k)$ where $\mathbf{v}$ is the result of running the ADD algorithm with $\mathbf{x}_k$ as the initial guess.

III *Trial points.* Obtain D_k^p via eq. (7.7). Compute f for each trial point in

$$\left\{\mathbf{p}_j = \mathbf{x}_k + \Delta_k \mathbf{d}_j : \mathbf{d}_j \in D_k^p, d_j = 1, \ldots, |D_k^p|\right\}$$

IV *Gather solutions.* Update the solution:

$$\begin{aligned} \Delta_{k+1} \leftarrow \sigma\Delta_k \quad \text{if} \min_{1\le j\le|D_k^p|} f(\mathbf{p}_j) > f(\mathbf{x}_k) \\ \mathbf{x}_k \leftarrow \arg\min_{1\le j\le|D_k^p|} f(\mathbf{p}_j) \quad \text{otherwise} \end{aligned}$$

V *Iterate.* Update $k \leftarrow k+1$, if the stopping condition is not met, go to (II)

The pruned direction set D_p^k is obtained as follows:

$$\begin{aligned} D_k^p &= \left\{\mathbf{d} \in D : \mathbf{d}^T\mathbf{v} \ge \beta|\mathbf{d}||\mathbf{v}|\right\} \quad \text{if } \mathbf{v} \text{ is a descent direction} \\ D_k^p &= \left\{\mathbf{d} \in D : \mathbf{d}^T\mathbf{v} \le -\beta|\mathbf{d}||\mathbf{v}|\right\} \quad \textit{otherwise} \end{aligned} \tag{7.7}$$

The *Approximate Descent Direction* (ADD) method tries to find and approximate descent direction, $\mathbf{v}$, from a point $\mathbf{x} \in \mathbb{R}^D$ by randomly generating $\{\mathbf{y}\}_{i=1}^{m_a}$ points in the neighborhood of $\mathbf{x}$ and calculating $\mathbf{v}$ in the following way:

$$\mathbf{v} = \sum_{i=1}^{m_a} \frac{f(\mathbf{y}_i) - f(\mathbf{x})}{\sum_{j=1}^{m_a} |f(\mathbf{y}_i) - f(\mathbf{x})|} \frac{\mathbf{p} - \mathbf{y}_i}{||\mathbf{x} - \mathbf{y}_i||}$$

where $||\mathbf{y}_i - \mathbf{x}|| \leq r_a$ for each $i \in \{1, \ldots, m_a\}$ and $r_a > 0$.

The algorithm. Our simulated annealing implementation can be summarized as:

I *Initialization.* Start at $k = 0$, with an arbitrary $\mathbf{x}_k$. Choose T_0^{gen} and T_0^{ac}.

II *Cauchy.* Then randomly generate $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$ according to equations (7.3) and (7.4).

III *Metropolis.* Accept or reject $\mathbf{x}_{k+1}$ by following the acceptance criterion given by equation (7.6).

IV *Thermal equilibrium.* In case of stagnation, if the solution is not in the tabu list: run the ADD and HPS local optimizers with $\mathbf{x}_{k+1}$ as the initial guess. Update both, $\mathbf{x}_{k+1}$ with the optimizer output and the tabu list.

V *Iterate.* Calculate the new set of temperatures, T_{k+1} and T_k^{ac} using equation (7.5)) and go back to (II) until the system freezes.

7.1.5 Experimental results

Our methodology for curve fitting has been tested against a well-known set of benchmark functions and datasets. All the experiments were run on a AMD-FX^{tm}-4100 Quad-Core Processor at 3600 *Mhz* with 8GB DDR3 RAM running *Linux 3.14.x LTS kernel* and *MATLAB 2012a*. Each experiment has been ran 30 times, dropping the five best and worst runs, in order to provide statistical evidence for the results presented and assert the experiment reproducibility. To evaluate the performance of our method, the results of our experiments are compared with a set of state of the art methods.

Parameter tuning. The parameter tuning step is crucial to the behavior of most meta-heuristic methods. In our case, there are two set of parameters that are very closely interlinked by the algorithm flow: the local search procedure parameters, and those of the *global* annealing.

- $\mathbf{x}_0$ is randomly selected in $(0, 1)^p$.
- $T_0^{gen} = 1$ and T_0^{ac} is generated according to (7.8) in order to ensure an initial transition probability close enough to 1.

- To decide if a given point $\mathbf{x}_{k+1}$ is accepted, we compute a random number $R_{k+1} \in (0,1)$, if it is lower than $\mathfrak{A}(\mathbf{x}_{k+1}, \mathbf{x}_k, T_k^{ac})$ we accept the new solution.
- *Stagnation* Check if $|f_k - f_j| < f_{\text{Tol}} = 10^{-6}$ for $j \in \{k - n_S, \ldots, k\}$ where $n_S = 100$.
- The neighborhood radius r_a is the minimum between $\left\{T_k^{gen}, 10^{-3}\right\}$ and $n_a = 10$.
- The mesh size Δ_0 is set to $1/10$, the shrinkage coefficient σ is set to 0.7 as per [Hedar and Fukushima, 2004] and the maximum number of iterations for the local search m_l is defined as $p \times 10$.
- The pruning control parameter $\beta = {}^{-1}/_{\sqrt{p}}$ provided the most consistent results .
- *Global stopping criterion*: Stop when a budget of $p \times 10^3$ $n_{f\text{eval}}$ is exhausted (but never stop in the middle of a local optimization).

The initial acceptance temperature is set upon a random search of feasible points in the neighborhood of the starting point.

$$T_0^{ac} = -\frac{1}{\log(0.9)} \cdot \max_{1 \le i \le n_a} |f(\mathbf{x}_0) - f(\mathbf{x}_i)| \quad \text{where} \quad ||x_i - x_0|| < 10^{-1} \tag{7.8}$$

Benchmark functions

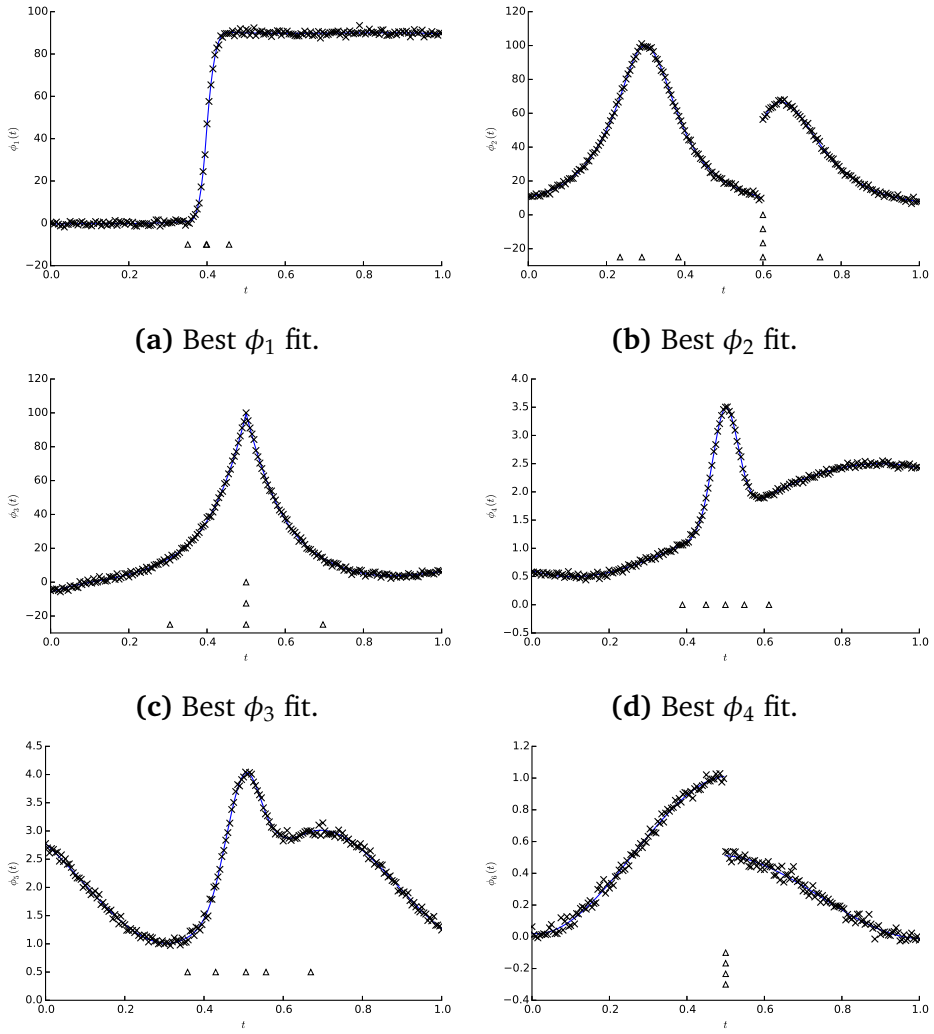
In this section we test our method against a set of benchmark functions extracted from the literature on the topic, we assume that all points must be treated equally, i.e. $\mu_i = 1$ for all i in equation (7.2), since no further information about the problem is available except for the data. Each function has been evaluated at $N = 201$ evenly spaced points with an additive Gaussian noise with variance σ (see each benchmark definition for further details on the chosen noise intensity). The numerical results and the best fitting spline can be consulted in Tables B.2-B.7 and Figures 7.1a-7.1f, respectively.

The first benchmark (7.9) consists in a continuous function with an asymptomatically behavior around $t = 4$, resembling and step function. The noise is computed by taking $\sigma = 1.0$.

$$\phi_1(t) = \frac{90}{1 + e^{-100(t-0.4)}} \quad \text{with} \quad t \in [0,1] \tag{7.9}$$

As can be seen in figure 7.1a, the optimal computed spline has a total of four knots which are grouped around $t = 0.4$ where the underlying data representation behaves like an step function.

The second benchmark (7.10) consists in a smooth function with a discontinuity in $t = 6.0$, and a different concavity behavior around it. The noise is computed by taking $\sigma = 1.0$.



(a) Best ϕ_1 fit.

(b) Best ϕ_2 fit.

(c) Best ϕ_3 fit.

(d) Best ϕ_4 fit.

(e) Best ϕ_5 fit.

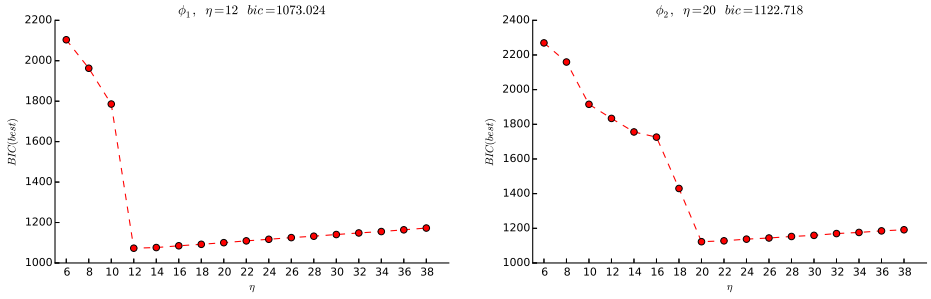
(f) Best ϕ_6 fit.

Figure 7.1: Best fitting splines (in continuous blue) to each example (in crosses).

$$\phi_2(t) = \begin{cases} \dfrac{1}{0.01 + (t - 0.3)^2} & \text{if} \quad t \in [0, 0.6) \\ \\ \dfrac{1}{0.015 + (t - 0.65)^2} & \text{if} \quad t \in [0.6, 1] \end{cases} \tag{7.10}$$

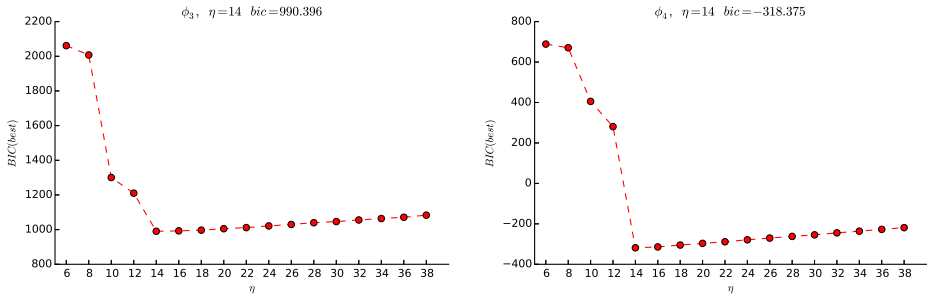
Although the example is particularly challenging since a B-spline of degree k needs a knot of multiplicity $k+1$ at $t = t_0$ in order to have a discontinuity at t_0, our method captures the singularity at $t = 0.6$ by obtaining exactly 4 equal knots at $t = 0.6$, remember that we are dealing with cubic splines ($k = 3$). The best computed spline totals 8 knots, with four used to correctly represent a discontinuity, the remaining

knots are used to accurately reconstruct the smooth part of the underlying curve, as it can be seen in Figure 7.1b.



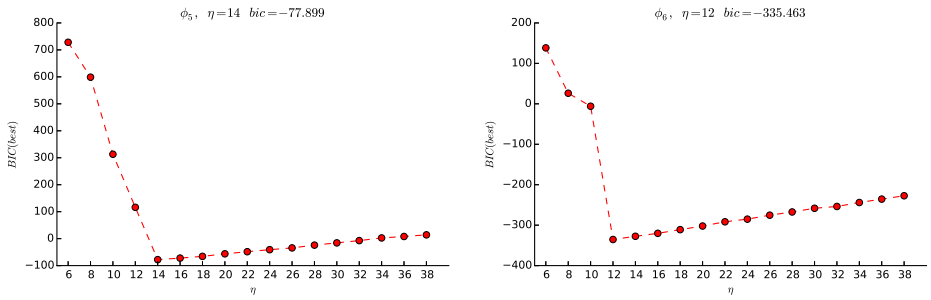
(a) Best BIC chart for ϕ_1.

(b) Best BIC chart for ϕ_2.



(c) Best BIC chart for ϕ_3.

(d) Best BIC chart for ϕ_4.



(e) Best BIC chart for ϕ_5.

(f) Best BIC chart for ϕ_6.

Figure 7.2: BIC evolution charts for each example.

The third benchmark (7.11) is a continuous function that loses the differentially at $t = 0.5$. The noise is computed by taking $\sigma = 1.0$.

$$\phi_3(t) = \frac{100}{e^{\|10t-5\|}} \frac{(10t-5)^5}{500} \quad \text{with} \quad t \in [0,1] \tag{7.11}$$

The best model obtained by our method has a total of 5 knots which are distributed such that 3 are identical at $t = 0.5$ where the cusp takes place and the remaining 2 knots are used to re-construct the general shape of the curve, see figure 7.2c for a visual representation. As has been pointed out in chapter 2 a cubic B-spline needs exactly 3 equal knots to loose de differentiability while still being continuous

The fourth benchmark (7.12) is a continuous function with a *soft* cusp-like point near $t = 0.5$, and a varying smoothness with sharply increasing and decreasing areas where several changes of concavity take place. The noise is computed by taking $\sigma = 0.06$.

$$\phi_4(t) = \sin(t) + 2e^{-30t^2} \quad t \in [-2, 2] \tag{7.12}$$

The 5 knots of the best fitting model are gathered around the cusp neighborhood, where the challenging features of the underlying function take place (Figure 7.2d).

The fifth benchmark provides a good example of a highly oscillating sinusoidal-type function with a soft peak-like point near $t = 0.5$. The noise intensity is set to $\sigma = 0.06$.

$$\phi_5(t) = \sin(2t) + 2e^{-16t^2} + 2 \quad t \in [-2, 2] \tag{7.13}$$

The best fitting model, Figure 7.2e, has 5 knots distributed along the changes of slope, with an special emphasis around $t = 0.5$, providing a very good fit to data.

The last benchmark function, with $\sigma = 0.03$, provides an extremely challenging example: a set of polynomials joined with continuity $\mathcal{C}^1$ at $t = 0.75$ and a strong discontinuity at $t = 0.5$.

$$\phi_6(t) = \begin{cases} 4t^2(3-4t) & \text{if} \quad t \in [0, 0.5) \\ \frac{4}{3}t(4t^2 - 10t + 7) - \frac{3}{2} & \text{if} \quad t \in [0.5, 0.75) \\ \frac{16}{3}t(t-1)^2 & \text{if} \quad t \in [0.75, 1] \end{cases} \tag{7.14}$$

As it has been mentioned before, in order to reconstruct the discontinuity with a B-spline we need $k + 1$ equal knots. The best fitting model, Figure 7.2f, has exactly 4 equal knots at $t = 0.5$ and represents with great accuracy the general shape of the curve.

In Table 7.2 we provide a graph of the BIC evolution for each example: the x-axis represents the free parameters η of the system whereas the y-axis shows the best BIC value for each baseline model. Note that the number of inner knots can be directly obtained from η as per chapter 3. These results are in the same range as those presented in the literature on the subject [Yoshimoto et al., 2003, Ülker and Arslan, 2009, Gálvez and Iglesias, 2011, Gálvez et al., 2015].

7.2 Approximation with local support curves

This chapter is devoted to present the results of *Memetic Simulated Annealing for Data Approximation with Local-Support Curves*, a paper to be presented in the *International Conference on Computational Science 2017* and published in the *Procedia of Computer Science* journal [Loucera et al., 2017a].

7.2.1 Introduction

This work introduces a new memetic optimization algorithm called MeSA (Memetic Simulated Annealing) to address the data fitting problem with local-support free-form curves. The proposed method hybridizes simulated annealing with the constrained optimization by linear approximations (COBYLA) local search optimization method. This approach is further combined with the centripetal parameterization and the Bayesian information criterion to compute all free variables of the curve reconstruction problem with B-splines. The performance of our approach is evaluated by its application to four different shapes with local deformations and different degrees of noise and density of data points. The MeSA method has also been compared to the non-memetic version of SA. Our results show that MeSA is able to reconstruct the underlying shape of data even in the presence of noise and low density point clouds. It also outperforms SA for all the examples in this paper.

7.2.2 The problem

Let $\{\mathbf{Q}_k\}_{k=1}^{M}$ be a set of points in $\mathbb{R}^d$. Our goal consists of finding a B-spline curve $\mathbf{s}(t)$ approximating the given data with high fidelity while trying to keep the model complexity as low as possible. Because the curve is parametric, our method must perform data parametrization, i.e. finding the $\{t_k\}$ associated with the original data. Then, we have to compute the control points $\mathbf{P}_i$ as well as the knots u_j and, finally, deal with the model complexity: how to minimize the number of free parameters of the system. Due to the constraints imposed on the boundary knots, we can assume that $\mathbf{s}(t_1) = \mathbf{Q}_1$ and $\mathbf{s}(t_M) = \mathbf{Q}_M$. As a result, the equation to minimize in a least-squares sense is given by:

$$E = \sum_{k=2}^{M-1} \left\| \mathbf{Q}_k - \sum_{i=0}^{n} N_{i,p}\left(t_k\right) \mathbf{P}_i \right\|_2^2 \tag{7.15}$$

where $N_{i,p}$ are the B-spline blending functions introduce in chapter 2. Note that when the order of the curve, the data parameterization, and the knot vector are known, equation (7.15) becomes a simple linear system. However, see chapter 3 for a full discussion on the topic, in many real-world problems, such values cannot be obtained directly from the data; instead, they have to be fully computed. In such a case, the least-squares minimization problem (7.15) becomes highly nonlinear, continuous, and multivariate. In addition, the computation of the knot vector has been proved to be a non-convex and multi-modal optimization problem [Jupp, 1978, Laurent-Gengoux and Mekhilef, 1993]. To overcome such difficulties we propose an optimization schema that deals with each sub-problem sequentially: firstly, data parameterization; then, knot vector and control points computation; and, finally, model complexity. This new method has two important contributions to the general methodology presented in chapter 5: On the one hand, the parameterization is computed by means of the centripetal method; on the other hand, we introduce

for the first time our memetic variant of the Simulated Annealing (the core of our method).

7.2.3 Memetic simulated Annealing.

Algorithm 7.1: MeSA: Memetic Simulated Annealing

Input: An initial guess $\mathbf{x}_0$, the cost function f, lower $\mathbf{l}$ and upper bounds $\mathbf{u}$
Output: The final solution $\mathbf{x}$
$T_0^{ac} \leftarrow \texttt{LearnParameters}(f, \mathbf{l}, \mathbf{u})$
$\mathbf{x} \leftarrow \mathbf{x}_0$ and $f_{\mathbf{x}} \leftarrow f(\mathbf{x})$
$T^{ac} \leftarrow T_0^{ac}$
$T^{gen} \leftarrow T_0^{gen}$
while *The System is not Frozen* **do**
  while *Thermal Equilibrium is not Reached* **do**
    $\mathbf{x}_{new} \leftarrow \mathfrak{N}(\mathbf{x})$ and $f_{new} \leftarrow f(\mathbf{x}_{new})$
    if $\mathfrak{A}(f_{new}, f_{\mathbf{x}}, T^{ac})$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}_{new}$ and $f_{\mathbf{x}} \leftarrow f_{new}$
    end
  end
  $\mathbf{x} \leftarrow \texttt{LocalLearn}(\mathbf{x}, \mathbf{l}, \mathbf{u})$ and $f_{\mathbf{x}} \leftarrow f(\mathbf{x})$
  $T^{ac} \leftarrow \mathfrak{T}^{ac}(T_0^{ac})$
  $T^{gen} \leftarrow \mathfrak{T}^{gen}(T_0^{gen})$
end
return $\mathbf{x}$

Memetic algorithms were originally introduced as an enhancement for genetic-driven meta-heuristics by introducing the idea of individual learners potentially able to refine some members of a population [Harris and Ifeachor, 1998], thus mimicking more closely the domain-specific processes of the universal Darwinian theory (incorporate knowledge of the problem). Nowadays, the memetic optimization approach has been applied with varying grades of success to a wide range of meta-heuristic optimization techniques beyond the genetic algorithms paradigm. Some illustrative examples of memetic approaches can be found, for instance, in [Petalas et al., 2007] where a memetic enhancement of the original particle swarm optimization algorithm is proposed, whereas in [Gálvez and Iglesias, 2016] the firefly optimization algorithm is enhanced with a local learning phase. In recent years the number of memetic approaches used in the field of data fitting with splines has greatly increased, for instance, the previously mentioned memetic firefly algorithm is used in [Iglesias and Gálvez, 2015] for curve fitting with rational Bézier curves, while a memetic enhanced bat algorithm is used for non-rational Bézier curve fitting in [Iglesias et al., 2016a]. Bézier rational surfaces are fitted by means of the memetic electromagnetism algorithm in [Iglesias and Gálvez, 2016].

In this paper we propose a memetic variant of the simulated annealing algorithm that performs solution refinement by means of the constrained optimization by linear approximations (COBYLA) local search procedure. The proposed optimization framework is described in Algorithm 7.1. Our memetic approach can be divided into three closely intertwined phases: information gathering, cooling, and local learning. About the information gathering phase, a remarkable feature of recent developments for memetic optimization is the inclusion of problem knowledge during the generation of the initial population [Hart et al., 2004]. Our MeSA approach learns the temperature parameter by initially exploring the fitness landscape via the `LearnParameters` process. The method generates a random population within the search space and sets the initial temperature to 0.8 times the worst energy transition.

During the *cooling phase*, the general SA schema is applied to compute the knots. It consists of two nested loops. The main or outer loop, labeled in the literature as the annealing or cooling loop, controls the temperature update process and the stop criterion. In our implementation, the stop criterion consists of running the outer loop for a predefined number of iterations N_{outer}. Let k be the outer iteration index from now on. The method keeps track of two control parameters, the acceptance (T^{ac}) and generation (T^{gen}) temperatures, which are initialized during the information gathering phase and subsequently updated during the cooling phase as: $\mathfrak{T}^{ac} : T^{ac}_{k+1} \leftarrow \frac{T^{ac}_0}{k}$ and $\mathfrak{T}^{gen} : T^{gen}_{k+1} \leftarrow \left(\frac{k_{outer}}{N_{outer}}\right)^{-1}$, respectively. The inner loop mimics the achievement of thermal equilibrium system state at a given temperature. Similarly to the outer loop, our implementation runs the inner loop N_{inner} iterations. During the inner loop, the new candidate solutions, $\mathfrak{N} : \mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$, are generated according to the inverse μ-law function [Sklar, 2001] given by:

$$\Delta\mathbf{x} = g^{-1}_{\mu}(\mathbf{y}) \odot (\mathbf{u} - \mathbf{l}) \quad \text{with} \quad g^{-1}_{\mu}(\mathbf{y}) = \frac{(1+\mu)^{|\mathbf{y}|} - 1}{\mu} \odot \text{sign}(\mathbf{y})$$

where the $\odot$ symbol represents the element wise vector multiplication, $\mu = 10^{100T^{gen}}$ and $\mathbf{y} \in U^d([-1,1])$. See [Yang et al., 2005] for a more detailed discussion on this search procedure. Then, the law governing the probability of accepting a given transition follows the modified Metropolis criterion [Metropolis et al., 1953]:

$$\mathfrak{A} \leftarrow \min\left\{1, \left(1 + \exp\left(\frac{\Delta f}{T^{ac}_k}\right)\right)^{-1}\right\}$$

At the end of each inner loop , i.e. when the thermal equilibrium for a given temperature has been reached, the algorithm performs a local search procedure with the last accepted point as an initial guess. This is called the *local learning phase*. This local search is performed via the COBYLA algorithm with a budget of N_{local} function evaluations. We provide a brief outline of the constrained optimization by linear approximations (COBYLA) method in section A.3, further information can be found in [Powell, 1994].

7.2.4 Method implementation

To summarize, our problem consists of finding the best B-spline fitting curve to a given set of (possibly noisy) data points while keeping the complexity of the model as low as possible. To do so, we need to compute a suitable parameterization of the data, the optimal number of knots along with their location, and the B-spline control net. We solve all those problems by combining four different techniques: the centripetal method for data parameterization, our Memetic Simulated Annealing (MeSA) method based on hybridizing Simulated Annealing with the COBYLA local optimization method along with least-squares minimization for the determination of the knot vector and the control points respectively, and finally, BIC (Bayesian information criterion) for model selection.

Centripetal parameterization. The centripetal parameterization is one of the most popular data parametrization methods [Farin, 2002], as it takes into account both the distribution of data and sharp turns. It is given by:

$$t_1 = 0 \qquad \text{and} \qquad t_k = \frac{\sum_{j=2}^{k} \left|\mathbf{Q}_j - \mathbf{Q}_{j-1}\right|^{\frac{1}{2}}}{\sum_{j=2}^{M} \left|\mathbf{Q}_j - \mathbf{Q}_{j-1}\right|^{\frac{1}{2}}} \qquad \text{for } k = 2, \ldots, M. \tag{7.16}$$

Solution encoding and parameter tuning. In this work, we consider cubic B-spline curves (although our method is actually independent on the order of the fitting curve). Now, assuming a given parameterization and the number of knots, the only unknowns in (7.15) are the control points and the knot vector. To compute the knots, we apply the MeSA algorithm, where each state is given by the following representation scheme: $\mathbf{x} \in (0,1)^{\sigma}$, where σ refers to the number of free (i.e. not-clamped) knots. The elements in $\mathbf{x}$ are then sorted to conform to the ordered structure of the knot vector. Finally, our fitness function is taken as $f \equiv E$. Regarding the parameter tuning, it is as follows: $N_{outer} = 500, N_{inner} = 50, N_{local} = 200, \sigma \in \{1, \ldots, 70\}$. We use the same parameter setup for the non-memetic SA approach, used here for comparative purposes. However, we increase the number of iterations as $N_{outer} = 1000, N_{inner} = 100$ for the non-memetic version to allow this simpler (and arguably slower) version to reach convergence. Once the knots are obtained, the control points can be computed by linear least-squares minimization of the functional E, leading to an overdetermined linear system that can be solved by standard numerical methods.

Model selection. The functional E does not contain any information about the model complexity, so the model might (potentially) be affected by over-fitting. This is a common problem when approximating data with B-splines, as E decreases as the number of knots increases. To overcome this limitation, we compute the modified

BIC (Bayesian information criterion) cost [Schwarz, 1978], given by:

$$BIC = M\log(E) + \zeta\log(M) \tag{7.17}$$

where ζ represents the total number of free parameters of the problem and $\log(.)$ represents the natural logarithm function. As a general rule, the model with the lower BIC is preferred.

Other metrics. To measure the goodness of the fit, we also compute the *normalized mean square error* NMSE (7.18) for each spatial component. The NMSE is given by:

$$NMSE_i = 1 - \left\| \frac{\mathbf{Q}_i - \mathbf{Q}_i^*}{\mathbf{Q}_i - \text{mean}(\mathbf{Q}_i^*)} \right\|^2 \tag{7.18}$$

where $\mathbf{Q}_i^*$ represents the reconstructed point associated with $\mathbf{Q}_i$, and i represents the spatial component (x and y in our examples). Note that the NMSE values vary between $-\infty$ and 1; the closer to one, the better the fit.

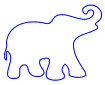
Shape	Parameters	σ	Method	E	BIC	$NMSE_x$	$NMSE_y$
	208	35	MeSA	0.0029422	-64.777	0.99988	0.99974
			SA	0.0081525	147.21	0.99911	0.99972
	204	50	MeSA	0.0012596	9.968	0.99993	0.99992
			SA	0.0027904	172.24	0.99984	0.99983
	208	60	MeSA	0.0008605	79.845	0.99992	0.99991
			SA	0.0059459	481.88	0.99946	0.99935
	752	40	MeSA	0.058327	1187.7	0.99991	0.99993
			SA	0.0700	1325.25	0.99981	0.99991

Table 7.1: Numerical fitting errors for MeSA and SA on the four examples in our benchmark.

7.2.5 Experimental results

To assess our MeSA method, we consider four datasets of synthetic shapes (elephant, camel, beetle, and bell) from [Carlier et al., 2016, Thakoor et al., 2007], depicted as black dots on the left of Figs. 7.3–7.6. The datasets consist of 104, 102, 104, and 376 points, respectively (only 124 are drawn for the bell example for better visualization). The figures also show the best fitting curve (blue solid line) obtained with the MeSA method according to the BIC. To this aim, Figs. 7.3–7.6 (right) show the evolution of the BIC value against σ.

A total of 26 independent runs are executed for each σ value. The three best and worst runs are then removed to provide statistical evidence for the results and assert the experiment reproducibility. The mean BIC value is displayed as a red solid line, while the minimum and maximum values are represented by the lower and upper

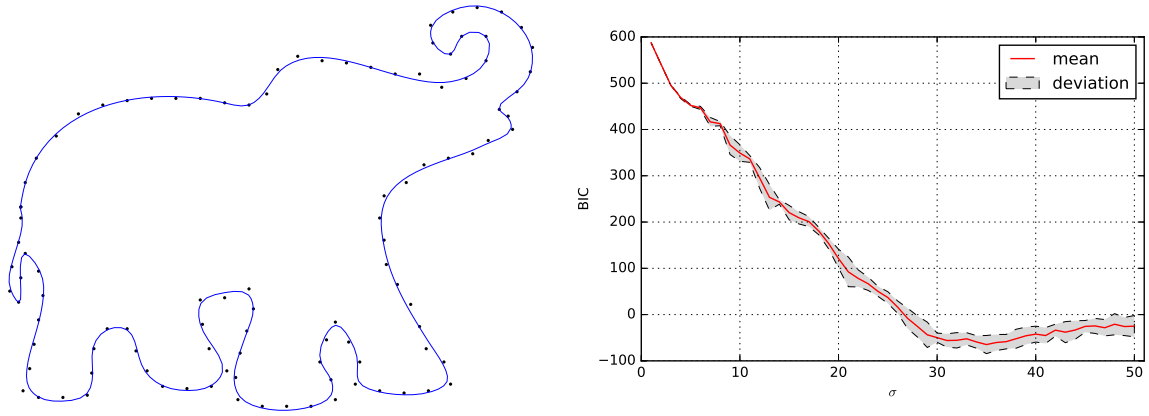


Figure 7.3: Reconstruction of the elephant shape: (l) best fitting curve ($\sigma = 35$); (r) BIC vs. σ.

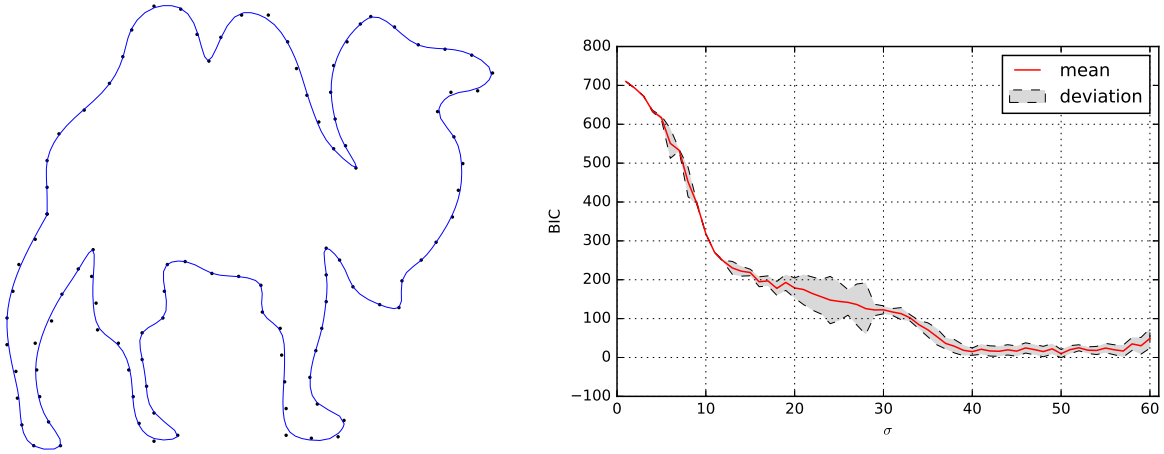


Figure 7.4: Reconstruction of the camel shape: (l) best fitting curve ($\sigma = 50$); (r) BIC vs. σ.

dashed lines. The deviation area is filled up with a gray tone for better visualization. From the figures and the numerical results, the best value for σ is determined and used for the fitting curves in Figs. 7.3–7.6 (left). Note that all shapes present difficult geometric features, such as strong changes of slope and curvature. Still, our method is able to reconstruct the general shape of the data with good accuracy, as confirmed visually in those figures. Table 7.1 summarizes our numerical results. The following data are reported (in columns): the shape, number of free parameters, best value for σ, the method used, and the E, BIC, and NMSE (for x and y) fitting errors. For further assessment, the results for the MeSA method are compared with those of a standard implementation of the (non-memetic) SA. As shown in Table 7.1, our method outperforms SA for all shapes in our benchmark.

Machine implementation issues. Regarding the implementation issues, all computations have been carried out on an `Intel i7-7600` quad-core processor with 16GB of

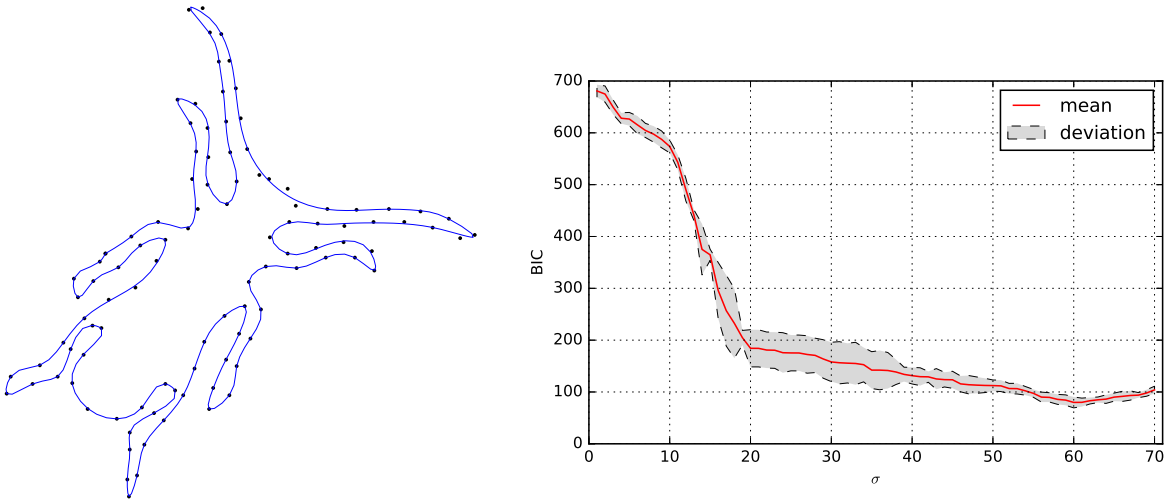


Figure 7.5: Reconstruction of the beetle shape: (l) best fitting curve ($\sigma = 60$); (r) BIC vs. σ.

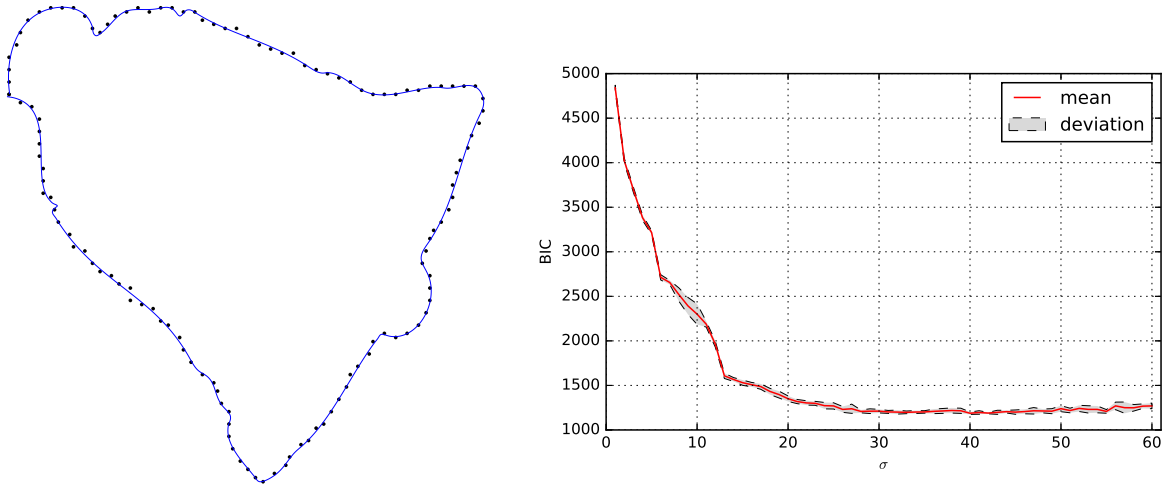


Figure 7.6: Reconstruction of the bell shape: (l) best fitting curve ($\sigma = 40$); (r) BIC vs. σ.

RAM. The source code has been implemented by the authors in the native programming language of MATLAB, v.2014b. We also used the COBYLA implementation in the `NLopt` library [Johnson, 2010]. All the simulations took less than 30 seconds. Future work includes the comparison with other optimizations schemes reported in the literature, as well as the extension of the proposed methodology to surface fitting.

7.3 Rational B-spline surfaces

We conclude our reverse engineering applications chapter with a very experimental case for data fitting by means of a NURBS surface. These results demonstrate that the findings of this Thesis are a solid foundation for future work.

7.3.1 Introduction

In this section we address the problem of obtaining an optimal, in a least squares sense, rational B-spline surface (NURBS) that fits a given point cloud. Our approach extends the methodology previously used on the case of rational Bézier surfaces by incorporating the Memetic Simulated Annealing, used as the vessel for B-spline fitting, and a set of data-filtering techniques used as a pre-process step in the case of massive and noisy datasets. The method has been applied to two challenging problems, we start with an academic free-form shape (a pipe elbow) used as a unit test. Then, we fit a massive and noisy point cloud which consists of (real world) 3D scanned data of the forehead of a Spanish regional Virgin statue. Our experimental results show that the proposed fitting method performs very well: on the one hand, the parameters of the underlying NURBS for the pipe elbow case are retrieved with extremely high precision. On the other hand, the underlying shape of the forehead is fitted with a NURBS that is capable of capturing the sharp changes of curvature while reconstructing the general shape of the data.

7.3.2 The problem

The problem that we address in this section follows the exact formulation that we provided in chapter 3, as we want to fit a NURBS surface $\mathbf{S}$, as per equation (2.17), to a point cloud in $\mathbb{R}^3$.

Given an structured point cloud $\left\{\mathbf{Q}_{p,q}\right\}_{p,q}$, we need to minimize the residual sum of squares given by equation (3.1), which we repeat here for easy reference:

$$E = \sum_{p=0}^{N}\sum_{q=0}^{M} \left\|\mathbf{Q}_{p,q} - \mathbf{S}\left(u_p, v_q\right)\right\|_2^2 = \sum_{p=0}^{N}\sum_{q=0}^{M} \left\|\mathbf{Q}_{p,q} - \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j} N_{i,k}\left(u\right) N_{j,l}\left(v\right)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j} N_{i,k}\left(u\right) N_{j,l}\left(v\right)}\right\|_2^2$$

As has been mentioned in numerous occasions along this dissertation, the problem is multivariate, multi-modal, non-convex and highly non-linear. Remember that the basic functions depends on two knot vectors $\boldsymbol{\tau} = (\tau_0, \ldots, \tau_{m_u})$ and $\boldsymbol{\zeta} = (\zeta_0, \ldots, \zeta_{m_v})$ which introduce a *hidden* set of unknowns of great importance for the shape of the fitting surface [Dierckx, 1995].

We handle the case of an unorganized point cloud $\{\mathbf{Q}_\mu\}_{\mu=0}^{M}$ in a similar way:

$$E = \sum_{\mu=0}^{M} \left\|\mathbf{Q}_\mu - \mathbf{S}_\mu\right\|_2^2$$

7.3.3 Method implementation

The implementation of our schema for selecting the optimal Rational B-spline surface distinguishes between structured point clouds, i.e. the sample connectivity is known, and unorganized data. When dealing with an organized dataset, we compute all the free parameters of the fitting NURBS by following the *all-in-one* schema discussed

in chapter 5. Otherwise, we follow a procedure to get a point cloud, with a known topology, that approximates the original unstructured data. Then, we follow the previous method for organized data with this baseline point cloud as the input.

Although our method only deals with be-cubic NURBS, a de-facto industry standard, the algorithms do not depend on the degree of the baseline model. In fact, if models of different degrees are supplied to the first phase in our methodology, the method automatically takes into account these differences as the number of free parameters is going to change and so will happen with the transformed fitness.

Structured data. As has been mentioned before, if the original data is structured we follow the *all-in-one* schema for data fitting: all the NURBS parameters are computed by a new take on the Memetic Simulated Annealing. The full methodology can be summarized as follows.

I *Selection of candidate models.* The first step in our methodology consists in selecting the underlying model that we are going to fit to the data. In this case, we make use of a selection of bi-cubic NURBS surfaces differentiated by the number of inner knots at each parametric direction. Note that, each choice also impacts the number of control points, as the degree is fixed.

II *Model fitting.* For each candidate model we perform the data fitting process by transforming the problem into an optimization scenario: the minimization of equation (3.1). Thus, we obtain a fitness score, the resulting RSS, along with the NURBS parameters (parameterization, knots, weights and poles) for each baseline model.

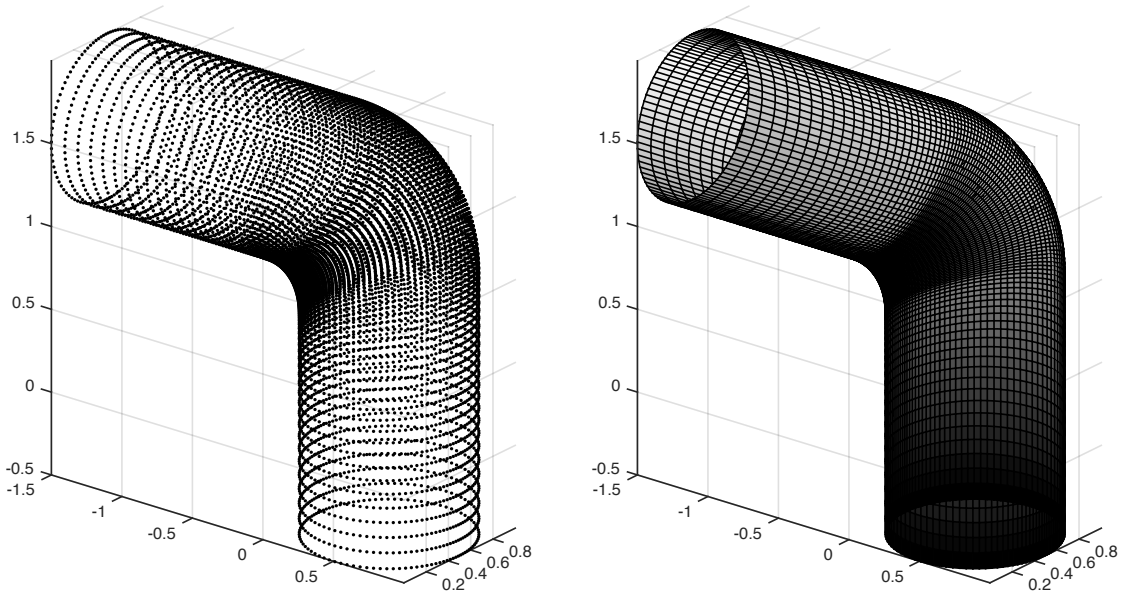
III *Model selection.* By means of the Bayesian information criterion (BIC) we compute the transformed fitness score for each fitted model. Finally, we compare all model scores in order to choose the one with the best BIC score (i.e. the model with the least BIC).

Unorganized point clouds. When the point cloud topology is unknown we have designed a fitting schema that improves on the work presented in [Loucera et al., 2017b] for Rational Bézier Surface fitting. This new methodology is specially tailored to real-world 3D point clouds which are characterized by their lack of a connectivity topology, massive size and noise due to measurement errors. In section 7.3.5 we explain the changes required to our method in order to deal with such difficulties.

7.3.4 Pipe Elbow

In this section we reconstruct the pipe elbow dataset previously used as an example for Rational Bézier Surface Fitting in [Loucera et al., 2017b]. As has been mentioned before, the point cloud was generated by evaluating a NURBS surface, so this is the perfect candidate to *unit* testing our framework for Rational B-spline Surface fitting.

As expected, our method is able to reconstruct with high level of detail the underlying surface, as evidenced by a NMSE of almost 1 for each component. Furthermore, our method reconstructs the full NURBS with the exact number of knots on each parametric direction with a MSE of $1e^{-16}$, and the full weights and control points with a combined MSE of $1e^{-14}$. See Figure 7.7 for the visual result of our fitting technique.



(a) Pipe elbow dataset. **(b)** Best fitting NURBS.

Figure 7.7: Reconstruction of the pipe elbow point cloud.

7.3.5 A real world scenario: a Spanish Virgin statue

In this section we will explain each part of the method for NURBS fitting by following its application to a real world scenario: a 3D scanned forefront of a Spanish Virgin wooden statue, of great importance from a historical heritage point of view (from now on referred as the *Virgin* dataset). This wooden statue is one of the few pieces of art that survived the catastrophic fire that hit the city of Santander in 1949. Thus, its preservation, inducing geometric models, are of great importance. The statue can be admired at the cathedral of Santander.

The point cloud has been kindly provided by 3DINTELLIGENCE, a company devoted to the research and development of new visualization techniques, specially in the historical heritage field. The point cloud has been acquired by means of the process known as *convergent photogrammetry*. In Figure 7.8 we can see a portion of the point cloud, the big rectangle refers to the supplied point cloud by 3DINTELLIGENCE whereas the small region refers to actual area reconstructed.

As discussed above, our problem consists of reconstructing the underlying shape of a point cloud by using a Rational B-spline Surface. In addition to the challenges already faced when fitting surfaces (a highly non-linear, multi-modal non-convex

Figure 7.8: Picture of the real-world Virgin wooden statue (left). Supplied dataset by 3DINTELLIGENCE (right).

function), real-world 3D scanned objects have a series of characteristics that makes them extremely complicated from a surface fitting point of view. Here we will present the most relevant challenges of such datasets along with our proposed solution.

Massive and noisy. The first challenge comes from its massive size (from thousands to millions points) [Pauly et al., 2002]: 3D scanned objects are represented by point clouds so dense, that they present a problem even for on-screen rendering. Due to the physical and mechanical processes used to obtain the data, the measurements are often corrupted by noise and other distortions. For instance, the *Virgin* dataset is comprised of nearly 400k points which have been obtained from a modern 3D scanning device in an attempt to retrieve a digital model of a Spanish Virgin statue. To circumvent this problem, we have added a pre-processing step to our methodology which consists in the concatenation of two procedures, namely:

I *Point cloud denoising.* We make use of a simple filter based on the nearest neighbor distance: a given point $\mathbf{Q}_\mu$ is marked as an outlier if the average distance of its ι nearest neighbors is greater than $\lambda \cdot \varepsilon$, where ε represents the median of average distance to neighbors of all points. For the *Virgin* dataset, $\iota = 4$ and $\lambda = 1$ provided the most reliable results. See Algorithm A.1 for the companion pseudo-code on the subject.

II *Data reduction.* We use a voxel grid filter to reduce the size of the point cloud, a method commonly referred as *downsampling*. A 3D voxel grid can be seen as a grid of small cubes, the voxels, which cover the entire point cloud. Then, we approximate all the points inside each voxel by its centroid. Thus, the *downsampling*.

Orientation. Our methodology for surface fitting, as any other method based on parametrized surfaces, needs an orientation: a parametric plane from which to

project the surface, i.e. a mapping $(u, v) \mapsto \mathbf{S}(u, v)$. To find a suitable reference system, we perform an orthogonal planar regression to the point cloud by using Principal Components Analysis (PCA), a well-known method outlined in Algorithm A.2. The parametric directions for the *Virgin* dataset are obtained from the PCA as seen in Figure 7.9. The defining vectors of the PCA plane along with the normal, form our new reference system.

A note on the implementation. The entire point cloud of the *Virgin* dataset is a real-world example that requires to be segmented in order to fit a surface to each segment. The complete problem of how to segment a point cloud in order to apply our method is part of a larger project beyond the limits of the Thesis. However, we have provided all the necessary tools to fit a NURBS to one of those segments. To showcase the visual performance of our method, we have manually segmented the point cloud by making use of the Meshlab software [Cignoni et al., 2008]. We do not have permission to show the entire point cloud, only our segmented piece which corresponds to the brow and part of both eyebrows. The downsampling has been carried out with the Point Cloud Library (PCL): a large scale, open project for point cloud processing [Rusu and Cousins, 2011]. A render of the best fitting NURBS, in a BIC sense, for the *Virgin* dataset could be seen in Figure 7.10. The results are quite promising for a first attempt of reconstructing a real world 3D scanned point cloud. Note that the segmented point cloud, after the pre-processing steps, has exactly 1713 points (i.e. 5139 parameters to be fitted).

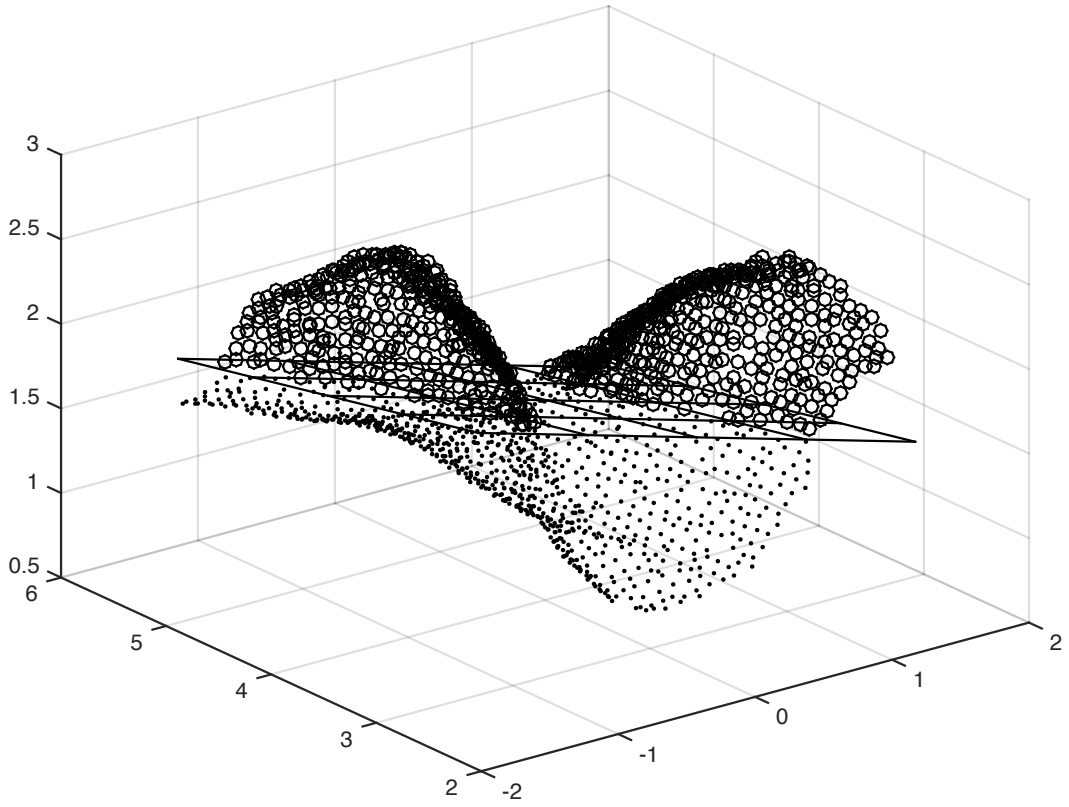


Figure 7.9: Get parametric plane $(\mathbf{u}, \mathbf{v})$ through PCA for the *Virgin* dataset.

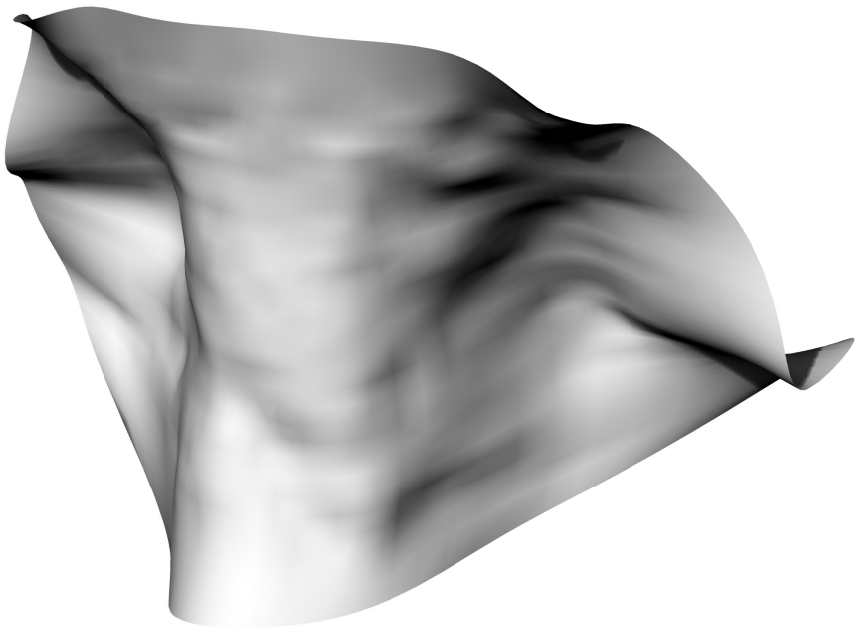


Figure 7.10: A rendered view of the best fitting NURBS surface for the *Virgin* dataset.

Part IV

Conclusions

Chapter 8
Conclusions and future work

> Every real story is a never ending story.
>
> Michael Ende, The Neverending Story

During the last two decades bio-inspired computing has attracted great interest in almost every area of science, engineering, and industry. In particular, the field of stochastic-driven mathematical optimization has become an attractive framework to resolve a wide range of non-linear problems in almost every science branch. The rich metaphors behind these methods usually lead to a better understanding of the problem being solved, whereas the ever-increasing push towards a better mathematical core, has developed into the first stages of a unified theory.

In this Thesis we have proposed the application of several Simulated Annealing implementations, a well-known thermodynamics-driven meta-heuristic optimization method, to different problems in non-linear data fitting such as the point cloud approximation by Bézier and B-spline geometric models. The core of our methodology consists of transforming the data fitting problem into an optimization procedure which is solved my merging least-squares regression with the Simulated Annealing technique. To maintain a good trade-off between model complexity and data-fidelity, we glue all the parts together by making use of some information sciences criteria, the Akaike and Bayesian Information Criteria to be more precise. In the case of point clouds with unknown topologies, we reconstruct the samples connectivity by approximating the original data with a baseline surface by means of the natural neighbor interpolant.

The main contributions to the state of the art presented in this Thesis are:

- A general methodology for point cloud approximation with Bézier and B-spline models.
- Several Simulated Annealing implementations tailored to each specific problem.
- A unified framework to deal with scattered data.

- A new Simulated Annealing variant which merges a signal-processing based visiting distribution with a memetic approach.

The rest of this chapter is devoted to summarize the conclusions of each part of the Thesis. We conclude with a brief outline of future lines of research to be built upon our findings.

8.1 Analysis of the general methodology

We have proposed a curve and surface reconstruction method consisting in three major phases. The first phase consists in reconstructing the data topology through a baseline model, if unknown, by means of the natural neighbor interpolation method. In phase two, we select a range of spline models of increasing complexity and transform the data fitting reconstruction into an optimization problem solved with a problem-specific Simulated Annealing procedure which retrieves the parameters of each model. In addition to the model-defining parameters, the second phase also assigns a fitness score to each proposed model. Finally, the last phase builds a transformed fitness score for each model, by means of the Akaike and Bayesian Information Criteria, and the model with the best fitness (i.e. the model with the lowest transformed fitness) is selected. The proposed methodology looks for the best model by taking into account the numerical quality of the approximation and the complexity of the model. Furthermore, this is done in a fully automated way, without the need for subjective decisions or parameters. To assert the goodness of the reconstruction we have applied it to several datasets that exhibit challenging features for data fitting, such as self-intersections, noisy and scattered point clouds. The results are robust from a numerical point of view, and ascetically pleasant.

8.1.1 Bézier models

The Bézier models used in chapter 6 for curve and surface fitting provide a series of low-complexity methods which can produce very pleasant results from a computer graphics point of view, making the resulting curve or surface indistinguishable from the ones produced by higher complexity models.

The papers in which these methods were first presented provide a good overview of the evolution of our Simulated Annealing implementations. We started in [Loucera et al., 2014] with an implementation of our methodology for data fitting by means of non-rational Bézier curves. The study provided an empirical evaluation of the performance of distinct cooling strategies and candidate generators for the optimization phase, the findings were clear: the SA by itself provided a good enough solution that could be further improved by performing a local search at the end. Of all the cooling and visiting distributions used, the *fast-fast* schedule, an initial approximation to the Fast Simulated Annealing [Szu and Hartley, 1987], performed the best. From this point on, it became clear that both, the Bayesian and Akaike Information Criteria provided similar results from an aesthetically point of view, but

the models chosen by means of the BIC were better suited to our needs due to the higher penalization.

The findings of the preceding article were later confirmed by the natural extension of the methodology to data fitting by means of Bézier surfaces [Iglesias et al., 2015b]. This article further refined the SA implementation by incorporating the local search procedure into the candidate generator function. In order to solve some issues with the premature convergence of the proposed method, we included a restarting strategy that performed fast sampled runs of the entire algorithm with different starting temperatures.

In [Iglesias et al., 2016b] we proposed two different schemes for Rational Bézier curve fitting. The problem being solved has a high number of unknowns as neither the parameterization nor the control points and its associated weights are given. Both schemes use the same implementation of the Simulated Annealing built on top of the previous ones: On the one hand, the time consuming restarting strategy used for Bézier surfaces was substituted by the procedure presented in Algorithm 4.2. On the other hand, we incorporated an adaptation of the acceptance and generation temperatures of Adaptive Simulated Annealing [Ingber, 1993a] in order to increase the exploitation of good directions in the fitness landscape while maintaining the chance to explore the solution space. The algorithm is further improved with the addition of adaptive rules for checking the thermal equilibrium and stop criteria.

The proposed schemes can be summarized as follows: the *all-in-one* schema search for all the unknowns at once whereas the *sequential* schema computes each set of unknowns in sequence, using the previous set of computed variables as the new input.

Finally, the Bézier cycle of papers ends with the fitting of point clouds by means of rational Bézier surfaces [Loucera et al., 2017b]. The main contributions of the paper are, an updated *all-in-one* schema that now fully embraces the Cauchy visiting distribution, and the addition of the natural neighbor interpolation method to fully reconstruct the point cloud topology in the case of scattered data. When the samples has not known connectivity structure, we build a baseline surface by constructing the natural neighbor interpolant of the data, thus transforming the problem into one already solved by the *all-in-one* schema.

8.1.2 B-spline models

The Bézier models discussed above provide the baseline for our B-spline data fitting methodology. Now, even in the most simple case, the optimization procedure must deal with a highly non-linear problem due to the inclusion of the knot vector, which we always treat as a new set of free variables of the system.

As we have seen in chapter 2, the B-splines curves/surfaces are a super set of the Bézier curves/surfaces, so in addition to the data fitting difficulties of the latter we must deal with one knot vector per each parametric direction. This fact has a direct impact on the optimization phase, as the problem being solved by the Simulated Annealing implementation is in the general case a continuous multivariate non-convex

multi-modal function. Another type of problem-driven difficulties arises from the point cloud underlying function of the data which can present non-smooth features, even to the point of losing the continuity.

To overcome such difficulties we propose a series of enhancements to the Simulated Annealing. In section 7.1 we present a SA implementation which further improves on the method presented for Bézier data fitting: instead of launching separate instances of a local search procedure, we merge the Cauchy visiting distribution with local optimization methods based on pattern search heuristics presented in [Hedar and Fukushima, 2004] which are closely interlinked with the flow of the Simulated Annealing algorithm. Although the results are on par with other state of the art techniques, the parameter fine-tuning exceeds by far those of other nature-inspired methods which have been successfully applied to free-knot spline reconstruction, such as genetic algorithms [Yoshimoto et al., 2003], particle swarm optimization [Gálvez and Iglesias, 2011] or clonal selection [Gálvez et al., 2015].

In section 7.2 we present the Memetic Simulated Annealing (MeSA) and its application to parametric B-spline curve fitting, that appeared for the first time in [Loucera et al., 2017a]. This memetic variant of the SA tries to simplify all the parameter tuning of the free-knot variant: on one side, the algorithm tries to locally learn the fitness landscape in order to provide a good starting temperature, with a simple statistical procedure, and exploit promising basins by means of a fast run of the *constrained optimization by linear approximations* [Powell, 1994] local optimizer (each time the thermal equilibrium is reached). Besides that, a new candidate generation function is introduced which follows the communications-based μ^{-1}-law. On the data fitting side, we propose the use of the centripetal parametrization, thus reducing the parametric-fitting to the already discussed free-knot spline fitting problem.

Finally, we present a further improved MeSA for NURBS fitting in section 7.3. The algorithm is tested against two very different problems: a point cloud generated by evaluating a rational B-spline surface, and real-world scanned data in the form of an unordered massive and noisy point cloud. In the original MeSA the thermal and stop criteria were fixed at the start of the algorithm whereas in the current implementation the variance of the fitness is checked in order to prematurely end a thermal cycle.

When dealing with the NURBS-induced dataset, our methodology does not require further improvements as we can recreate the original surface parameters with tremendous precision. However, when dealing with a massive scattered dataset, we first apply a de-noising filter and data-reduction techniques, then our methodology for unordered point clouds is applied with a NURBS as baseline model.

8.2 Lines of research

In the following we list some interesting lines of research. We distinguish between two main areas: the meta-heuristic approach, where we present some enhancements to our Simulated Annealing implementation, and the data fitting problem.

8.2.1 Meta-heuristics

Our future work on the Simulated Annealing proposal for data fitting is going to be directed towards getting rid of the local optimizers as most derivative-free procedures suffer from the *curse of dimensionality* and slow down the system. Our plan to overcome the refinement provided by the local search algorithm goes towards the inclusion of various visiting distributions that adapt to both the state of the system and the fitness landscape, thus fully embracing the memetic approach. Our research in this area has started with the inclusion of a pool of Lévy distributions in conjunction with the temperature-binded Cauchy distribution and the μ^{-1}-law. As the system advances, the visiting distribution has the ability to perform Lévy flights in order to explore the space. How far this *random* walk goes depends on the state of the system energy distribution during the thermal cycle. Thus, we guarantee the ability to escape basins of attraction and energy plateaus. To exploit promising regions, we run short temperature-unchained μ^{-1}-law searches.

This new MeSA schema is still under heavy development, we need to further refine the adaptive choice of a distribution at any given temperature and study the mathematical behavior of the complete model. Once these steps are fulfilled, we plan to test the algorithm against the classical optimization benchmark problems and compare the results with other state of the art nature-inspired methods.

8.2.2 Data fitting

Future work will start with the adaptation of the full MeSA algorithm for NURBS data fitting and its comparison with other optimization schemes reported in the literature.

Once this task is finished we plan to expand our research horizon into two directions. On the one hand, we want to solve the *complete* problem, i.e. given a massive scattered point cloud, segment it, fit each part and then glue all the patches together. On the other hand, we want to build a one-step free-knot method, one that does not rely on the optimization of each possible model.

The complete model

The *complete* problem refers to the case when the original data is a massive and noisy point cloud. In this case, we plan to divide the problem into three clearly differentiated stages:

Point cloud segmentation. We need to segment the point cloud into regions that can be fitted with the methodology proposed in this Thesis, i.e. regions that ca be fitted in the form $(u, v) \longmapsto (\tilde{x}, \tilde{y}, \tilde{z})$ in some coordinated axis.

Hole detection. The method needs to detect the presence of holes which can be of two types: either the hole is intrinsic to the underlying function of the data, e.g. a torus surface, or is the result of errors during the data acquisition. The problem is as important as it is difficult to solve: if an original hole is filled, then the resulting surface does not resemble the underlying shape. In case of

missing data, if the problem is not properly addressed it could drive to serious numerical issues.

Data fitting. This is the final result of the present work.

Region merging. Once all the previous steps has been taking care of, we need to glue all the fitted patches together by attending a full set of constraints on the boundaries, typically $\mathcal{G}^k$ continuity.

One step free-knot splines

The methodology presented in this Thesis relies on the optimization of a given range of models is what is often referred as a two-step process: select a set of models differentiated by their complexities, then for each model optimize its parameters to obtain a score. For each score, compute its transformed fitness by means of some criteria that penalizes complex models. Our intention here is to use meta-heuristics that allow for the mix of continuous and binary variables to search for the correct number of knots and their optimal locations at the same time, by minimizing a transformed problem where a binary variable is introduced for each internal knot. The fitness score to minimize is going to be BIC, as the *extra* binary variables would decide which knot plays a role in the model.

In the literature there are not many methods that address this problem in a one-step way. To the knowledge of the authors, beside the classical or heuristic-driven methods, the main ones are [Yoshimoto et al., 2003], where no true multiple knots are obtained and [Valenzuela et al., 2013] which produces redundant knots. Our first research on the subject, based on the enhanced scatter search proposed in [Egea et al., 2009, Egea et al., 2010] are very promising: in the case of continuous data, even in the presence of cusps and other singularities, our methodology computes the correct number of knots and its optimal positions, even producing true multiple knots.

Appendix A
Algorithms

In this appendix we present some algorithms that have been kept out of the dissertation body due to information-flow reasons.

A.1 The Omega glyph

In this section we provide the necessary code to reproduce the Bézier glyph seen in Figure 2.2

```
// glyph.asy :
// http://tex.stackexchange.com/questions/178892/
// can-i-visualize-the-bezier-control-points-of-a-letter-in-tex
size(7cm);
\usepackage{fontspec}
\setmainfont{Times New Roman}
import fontsize;

defaultpen(TimesRoman()),fontsize(9pt));

real wd=0.6bp;
pen dotPen=deepblue+wd;
pen dotFill=dotPen;

pen dotPenB=blue+wd;
pen dotPenC=red+wd;

pen linePen=deepblue+wd;
pen fillPen=lightgrey+opacity(0.5);

pen thinLinePen=black+wd/2;


guide[] g;
```

```
g=texpath("$\Omega$");

filldraw(g,fillPen,linePen);

pair a,b,c,d;
pair labdir;
int pointNo=0;

for(int i=0;i<g.length;++i){
for(int j=0;j<size(g[i])-1;++j){
a=point(g[i],j);
d=point(g[i],j+1);
if(straight(g[i],j)){
draw(a--d,thinLinePen);
}else{
b=postcontrol(g[i],j);
c=precontrol(g[i],j+1);
draw(a--b--c--d,thinLinePen);
dot(b,dotPenB,UnFill);
dot(c,dotPenC,UnFill);
}
dot(a,dotPen,Fill(dotFill));
labdir=rotate(-90)*dir(g[i],j);
label("$\scriptsize "+string(pointNo)+"$",a,labdir);
++pointNo;
}
dot(d,dotPen,Fill(dotFill));
labdir=rotate(-90)*dir(g[i],size(g[i])-1);
label("$\scriptsize "+string(pointNo)+"$",d,labdir);
}
```

A.2 An introduction to simplex-driven methods

In this section we present a brief introduction to the simplex methods [Nelder and Mead, 1965] used for the minimization of real valued function in $\mathbb{R}^d$. Note that, the simplex idea also plays an important role in the COBYLA algorithm.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a real valued function. For the purposes of this introduction we call $d+1$-simplex a set of $d+1$ points $\{\mathbf{x}_k\}_{k=0}^{d}$ in $\mathbb{R}^d$, which have been already ordered to satisfy $f_0 \leq f_1 \leq \ldots \leq f_d$ where $f_k \equiv f(\mathbf{x}_k)$, such that the polytope formed by their convex hull has a non-zero finite volume.

A Nelder-Mead solves the minimization of f by maintaining a $(d+1)$-simplex which is transformed at each iteration, through the use of a pre-configured set of rules. These heuristics try to substitute a simplex vertex in such a way that the re-

placement has a lower function value that the preceding one. The rules are backed by a set of parameters λ_r, λe, λ_c, and λ_c which are respectively the reflection, expansion, contraction and shrink coefficients.

To be more concise, let $\{\mathbf{x}_k\}_{k=0}^{d}$ be the simplex at any given iteration and $\tilde{\mathbf{x}}$ its centroid (with the last point removed for its computation). Then:

I *Reflection.* Compute the reflection $\tilde{\mathbf{x}}_r = \tilde{\mathbf{x}} + \lambda_r (\tilde{\mathbf{x}} - \mathbf{x}_d)$. If the reflection condition $f_0 \leq f(\tilde{\mathbf{x}}) < f_d$ holds, replace the worst point in the simplex with $\tilde{\mathbf{x}}_r$, reorder and start over.

II *Expansion.* If $\tilde{\mathbf{x}}_r < f_0$, then compute the centroid extension as per equation $\tilde{\mathbf{x}}_e = \tilde{\mathbf{x}} + \lambda_e (\tilde{\mathbf{x}}_r - \tilde{\mathbf{x}})$. If $\tilde{\mathbf{x}}_e < \tilde{\mathbf{x}}_r$ holds, replace the worst point in the simplex with $\tilde{\mathbf{x}}_e$, reorder and start over. In other case, the replacement is done with $\tilde{\mathbf{x}}_r$.

III *Contraction.* When $f(\tilde{\mathbf{x}}_r) < f_{d-1}$, we compute the contracted point by following $\tilde{\mathbf{x}}_c = \tilde{\mathbf{x}} + \lambda_c (\tilde{\mathbf{x}}_d - \tilde{\mathbf{x}})$. If $\tilde{\mathbf{x}}_c < f_d$ holds, replace the worst point in the simplex with $\tilde{\mathbf{x}}_c$, reorder and start over.

IV *Shrink.* Replace all simplex vertex by following $\mathbf{x}_j \leftarrow \mathbf{x}_0 + \lambda_s (\mathbf{x}_j - \mathbf{x}_0)$ for all $j > 0$ and start over.

A.3 Optimization by linear approximations

In this section we present the Constrained Optimization By Linear Approximations (COBYLA) algorithm [Powell, 1994]: a nonlinear derivative-free constrained optimization method that uses a linear approximation approach for minimizing a real valued function $f : \mathbb{R}^d \to \mathbb{R}$. The algorithm can be summarized as a sequential trust-region method that employs linear approximations to the objective and constraint functions. Each approximation is constructed as a linear interpolation at certain points, a $(d+1)$-simplex, in the solution space while maintaining a regular-shaped simplex over iterations.

What follows is a brief outline of the COBYLA method, we do not provide the pseudo-code. However, the full source code was released under the GNU Lesser General Public License (GNU LGPL). In all of our works we make use of the implementation found in NLopt [Johnson, 2010]: a free/open-source library for nonlinear optimization.

A.3.1 The COBYLA algorithm.

The COBYLA algorithm iteratively minimizes a problem with the following canonical form:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq b_j,\ j = 1, \ldots, m. \end{aligned}$$

At each iteration i a trust region is constructed with an adaptive radius $R > 0$. The method approximates the objective and constraint functions by performing linear interpolations over all the vertexes of a $(d+1)$-simplex. The simplex size R is adaptively changed at each iteration between $[R_{start}, R_{end}]$, the initial and final values provided at the beginning of the algorithm. The simplex plays the trust region role of a trust-region driven algorithm, a zone around a search point where f is expected to have a local minimum. For a given radius R, the method the solutions are perturbed in such a way that they stay with the region. The simplex size is reduced until R_{end} is reached, and the reductions are based on how well the approximation model agrees with the cost function evaluations.

Let $\tilde{f}$ be the linear interpolation which approximates f, and $\{\mathbf{x}_k\}_{k=0}^{d}$ the simplex of size R at any given iteration. The next point $\tilde{\mathbf{x}}$ is obtained by minimizing $\tilde{f}$ subject to $\|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2$, whose solution can be directly computed as:

$$\tilde{\mathbf{x}} = \mathbf{x}_0 - \left(\frac{R}{\|\nabla \tilde{f}\|_2} \right) \nabla \tilde{f}$$

note that, $\tilde{f}$ is constructed as and interpolant of the simplex vertexes, so (by construction) satisfies $\tilde{f}(\mathbf{x}_k) = f(\mathbf{x}_k)$. Thus, $\tilde{f}(\tilde{\mathbf{x}}) < \tilde{f}_0 = f_0 \leq f_k$ holds for all k. Now, a vertex from the simplex is substituted by $\tilde{\mathbf{x}}$. The replacement is done by following a simple heuristic that avoids the construction of degenerated polytopes (those with no volume). A careful choice of R_{end} must be carried out to ensure that $\nabla \tilde{f}$ is not misleading the search.

A.4 Point cloud denoising

In this section we present our simple, yet robust, technique for point cloud denoising based on the one presented in [Rusu et al., 2008]. Algorithm A.1 provides a high level interface to the method, the implementation uses a kd-tree for indexing, and all the computations are done by the returned indexes. Thus, no list appending is needed.

Algorithm A.1: Point cloud denoising

Input: A point cloud $\mathbf{Q} = \{\mathbf{Q}_\mu\}_{\mu=1}^{M}$, number of neighbors $k \in \mathbb{N}$, $\lambda \in \mathbb{N}$
Output: $\tilde{\mathbf{Q}} = \{\tilde{\mathbf{Q}}_{\tilde{\mu}}\}_{\tilde{\mu}=1}^{\tilde{M}} \subset \{\mathbf{Q}_\mu\}_{\mu=1}^{M}$
Initialize $\tilde{\mathbf{Q}}$ as an empty list.
foreach *point* $\mathbf{x}_\mu \in \mathbf{Q}$ **do**
  Compute the distances $\mathbf{d}_\mu$ of the k nearest neighbors of $\mathbf{x}_\mu$, excluding itself
  $m_\mu \leftarrow \text{mean}(\mathbf{d}_\mu)$
end
$\varepsilon_{mean} \leftarrow \text{mean}(\{m_\mu\})$
$\varepsilon_{std} \leftarrow \text{std}(\{m_\mu\})$
Compute the threshold $t \leftarrow \varepsilon_{mean} + \lambda\varepsilon_{std}$
foreach *point* $\mathbf{d}_\mu$ **do**
  if $\mathbf{d}_\mu \leq t$ **then**
    Append $\mathbf{x}_\mu$ to $\tilde{\mathbf{Q}}$.
  end
end
return $\tilde{\mathbf{Q}}$

A.5 Orthogonal Planar Regression by PCA

In this section we present the pseudo code for Orthogonal Planar Regression by using Principal Components Analysis (PCA), a method that tries to fit a plane to a set of given data. Not that the algorithm could be easily generalized to other dimensions .

Algorithm A.2: PCA regression plane

Input: A point cloud $\mathbf{Q}_\mu = (x_\mu, y_\mu, z_\mu)$
Output: Orthogonal regression plane defining vectors $\mathbf{e}_u, \mathbf{e}_v$ and its normal $\mathbf{n}$
Let $\mathbf{C}, \mathbf{s}$ be the coefficients and scores associated to $\text{pca}(\{\mathbf{Q}_\mu\})$
The regression plane normal gets determined as $\mathbf{n} \leftarrow \frac{\mathbf{C}_z}{\|\mathbf{C}_z\|_2}$
The u, v parametric directions get defined as:

$$\begin{cases} \mathbf{e}_u \leftarrow \frac{\mathbf{C}_x}{\|\mathbf{C}_x\|_2} \\ \mathbf{e}_v \leftarrow \frac{\mathbf{C}_y}{\|\mathbf{C}_y\|_2} \end{cases}$$

return $\mathbf{e}_u, \mathbf{e}_v, \mathbf{n}$

Appendix B
Tables

Model			Statistical results						
dataset	type	schema	n_{fval}	BIC	BIC_{avg}	n_{pol}	x_{mean}	y_{mean}	z_{mean}
logo	NR	-	10010	-691.913	-691.83	13	0.0005	0.0001	NA
logo	R	ALL	4208	-661.908	-658.804	13	0.0002	0.0003	NA
logo	R	SEQ	13031	-770.212	-701.338	13	0.0001	0.0001	NA
shoe	NR	-	16624	2915.2	3020.1	24	0.097	0.030	0.011
shoe	R	ALL	6648	2098.859	2470.634	24	0.008	0.012	0.008
shoe	R	SEQ	20202	1996.226	2289.712	24	0.011	0.012	0.008
torus	NR	-	18546	-2204.821	-2301.567	20	1.063e-04	1.231e-04	1.064e-04
torus	R	ALL	1529	-6067.465	-6001.324	21	6.42e-05	5.3904e-05	2.1686e-05
torus	R	SEQ	5286	-7798.643	-7796.846	20	1.4814e-05	1.4814e-05	6.75e-06
torus10	NR	-	13276	4307.409	4390.492	17	0.311	0.342	0.225
torus10	R	ALL	3768	4354.618	4414.523	19	0.300	0.304	0.200
torus10	R	SEQ	23119	4341.604	4381.367	17	0.312	0.323	0.201
torus20	NR	-	12027	3887.839	4991.823	20	0.222	0.201	0.183
torus20	R	ALL	3778	3474.463	3479.345	20	0.130	0.137	0.091
torus20	R	SEQ	45775	3446.804	3465.673	20	0.128	0.139	0.092
torus30	NR	-	16174	3033.538	3561.901	20	0.109	0.094	0.086
torus30	R	ALL	5546	3060.624	3066.298	20	0.097	0.089	0.075
torus30	R	SEQ	48511	3021.939	3035.634	20	0.098	0.089	0.071
torus40	NR	-	11015	6662.487	6700.738	40	0.183	0.144	0.173
torus40	R	ALL	5111	2740.208	2788.422	20	0.074	0.073	0.047
torus40	R	SEQ	53640	2708.680	2729.980	20	0.074	0.072	0.048
torus50	NR	-	14101	2386.160	2501.900	25	0.057	0.059	0.038
torus50	R	ALL	4205	2357.285	2408.811	21	0.054	0.053	0.038
torus50	R	SEQ	45206	2344.050	2378.280	20	0.055	0.052	0.038
torus60	NR	-	12052	3574.142	4713.190	20	0.179	0.121	0.170
torus60	R	ALL	4012	2116.800	2145.321	20	0.041	0.045	0.031
torus60	R	SEQ	45191	2051.304	2080.235	20	0.031	0.033	0.019
torus70	NR	-	8860	3574.142	3866.889	21	0.058	0.057	0.058
torus70	R	ALL	4205	2210.012	2284.590	21	0.039	0.0406	0.027
torus70	R	SEQ	45797	1963.317	1995.759	21	0.038	0.0393	0.028
torus80	NR	-	14100	3574.142	3934.296	25	0.038	0.044	0.026
torus80	R	ALL	4002	1814.950	2008.400	21	0.032	0.036	0.022
torus80	R	SEQ	13366	1803.356	1931.721	20	0.032	0.0364	0.022
torus90	NR	-	14100	1828.028	1981.534	25	0.031	0.041	0.024
torus90	R	ALL	4204	1671.145	1799.627	21	0.0307	0.0309	0.021
torus90	R	SEQ	13350	1654.944	1786.873	20	0.0307	0.0312	0.022

Table B.1: Rational Bézier curve fitting: Experimental results for each dataset.

p	stat	E	RMSE	BIC	Runtime
1	max	29970.64076	12.21096	2103.72251	11.49650
	mean	29970.64004	12.21096	2103.72251	11.24640
	min	29970.63980	12.21096	2103.72250	11.20110
2	max	14108.04217	8.37790	1962.88100	13.35840
	mean	14108.03803	8.37790	1962.88094	12.42730
	min	14108.03630	8.37790	1962.88091	12.34860
3	max	5546.76717	5.25318	1785.84813	13.84280
	mean	5546.76404	5.25317	1785.84802	13.63670
	min	5546.76380	5.25317	1785.84801	13.56930
4	max	151.69946	0.86875	1073.04183	15.00680
	mean	151.69369	0.86873	1073.03418	14.8365
	min	151.68610	0.86871	1073.02412	14.77720
5	max	151.47680	0.86811	1083.35320	16.46310
	mean	148.56240	0.85972	1079.44829	16.10750
	min	146.22320	0.85292	1076.25825	15.99370
6	max	149.60590	0.86273	1091.46178	18.19350
	mean	145.64000	0.85122	1086.06158	17.45940
	min	144.83150	0.84885	1084.94265	17.31000
7	max	143.99910	0.84641	1094.39071	19.74890
	mean	143.31650	0.84440	1093.43564	18.91000
	min	142.48050	0.84194	1092.25972	18.71130
8	max	144.05970	0.84659	1105.08189	21.09730
	mean	142.67600	0.84251	1103.14194	20.31340
	min	140.71400	0.83670	1100.35872	19.99530
9	max	141.70660	0.83965	1112.37821	24.41490
	mean	140.58690	0.83632	1110.78369	23.83150
	min	139.53160	0.83318	1109.26922	23.37060
10	max	140.00710	0.83460	1120.55964	26.76100
	mean	138.65570	0.83056	1118.61009	25.84430
	min	137.61070	0.82742	1117.08948	24.63840
11	max	138.27430	0.82942	1128.66305	28.94090
	mean	137.20160	0.82619	1127.09765	27.88740
	min	136.04780	0.82271	1125.40019	26.53520
12	max	137.58170	0.82734	1138.26034	31.06730
	mean	135.81060	0.82199	1135.65605	30.17210
	min	133.56320	0.81516	1132.30207	29.24250
13	max	136.12580	0.82295	1146.72862	32.64620
	mean	133.75280	0.81574	1143.19380	31.76430
	min	132.05320	0.81054	1140.62333	30.74540
14	max	134.11910	0.81686	1154.35013	35.68190
	mean	132.27940	0.81124	1151.57394	34.47980
	min	130.21940	0.80490	1148.41912	33.02300
15	max	133.23850	0.81417	1163.63266	38.07740
	mean	130.28290	0.80509	1159.12372	36.59570
	min	127.86990	0.79760	1155.36605	35.26970
16	max	131.45260	0.80870	1171.52689	41.00360
	mean	128.92810	0.80090	1167.62921	39.62690
	min	126.56910	0.79353	1163.91745	37.89200
17	max	130.32930	0.80524	1180.40852	43.21570
	mean	127.90270	0.79770	1176.63082	41.95970
	min	125.41460	0.78991	1172.68222	40.40720

Table B.2: ϕ_1

p	stat	E	RMSE	BIC	Runtime
1	max	68402.37117	18.44750	2269.58555	11.56210
	mean	68402.36217	18.44750	2269.58552	11.50500
	min	68402.36200	18.44750	2269.58552	11.45560
2	max	37479.62025	13.65525	2159.26851	13.05990
	mean	37479.61246	13.65525	2159.26847	12.90750
	min	37479.61220	13.65525	2159.26847	12.82990
3	max	10563.94681	7.24962	1915.33870	14.42060
	mean	10563.94658	7.24962	1915.33869	14.38930
	min	10563.94300	7.24962	1915.33863	14.33370
4	max	8126.02660	6.35830	1873.20796	15.88050
	mean	6871.94770	5.84712	1839.51543	15.84820
	min	6683.83580	5.76653	1833.93657	15.80070
5	max	4299.07455	4.62476	1755.84344	17.90920
	mean	4299.07022	4.62476	1755.84323	17.72160
	min	4299.06430	4.62476	1755.84296	17.43600
6	max	3514.86960	4.18174	1725.96918	19.64220
	mean	3514.86122	4.18173	1725.96870	19.45490
	min	3514.85440	4.18173	1725.96831	19.10640
7	max	768.12770	1.95487	1430.89464	21.65880
	mean	765.65040	1.95172	1430.24535	21.42900
	min	763.74590	1.94929	1429.74475	21.10210
8	max	157.27600	0.88457	1122.72455	23.32230
	mean	157.27370	0.88457	1122.72161	23.1160
	min	157.27120	0.88456	1122.71841	22.58670
9	max	157.26160	0.88453	1133.31275	24.04900
	mean	155.84430	0.88054	1131.49305	23.52540
	min	152.94640	0.87231	1127.72030	22.73710
10	max	156.99730	0.88379	1143.58127	26.34130
	mean	155.00060	0.87815	1141.00854	25.63890
	min	152.27110	0.87038	1137.43748	24.96020
11	max	156.14220	0.88138	1153.09012	28.10870
	mean	152.25300	0.87033	1148.02019	27.57220
	min	149.28590	0.86181	1144.06444	26.97710
12	max	156.10720	0.88128	1163.65167	30.99950
	mean	150.69380	0.86586	1156.55778	29.40220
	min	147.87580	0.85773	1152.76346	28.66590
13	max	154.57820	0.87695	1172.27987	32.64220
	mean	149.49050	0.86240	1165.55295	31.45280
	min	144.87190	0.84897	1159.24498	30.01950
14	max	154.71750	0.87735	1183.06753	34.75550
	mean	148.36500	0.85915	1174.64052	33.69370
	min	144.70550	0.84849	1169.62059	33.01380
15	max	151.51630	0.86822	1189.47170	37.75210
	mean	145.08280	0.84959	1180.75059	36.31950
	min	141.77210	0.83984	1176.11076	35.00960
16	max	147.50510	0.85665	1194.68539	39.78080
	mean	143.05570	0.84363	1188.52902	38.69920
	min	140.66370	0.83655	1185.13974	37.37080
17	max	147.91320	0.85784	1205.84734	42.18510
	mean	142.78900	0.84285	1198.76056	41.23680
	min	137.73270	0.82779	1191.51387	40.08130

Table B.3: ϕ_2

p	stat	E	RMSE	BIC	Runtime
1	max	24292.68959	10.99360	2061.50391	11.27830
	mean	24292.68209	10.99359	2061.50385	11.23330
	min	24292.67290	10.99359	2061.50377	11.21980
2	max	18196.09310	9.51461	2014.02784	12.55090
	mean	17670.14130	9.37609	2008.13238	12.39690
	min	17611.70230	9.36058	2007.46653	12.35690
3	max	495.84560	1.57063	1300.49223	13.91820
	mean	495.84350	1.57063	1300.49138	13.71700
	min	495.84250	1.57063	1300.49097	13.65070
4	max	315.87100	1.25359	1220.46177	15.46300
	mean	301.81750	1.22539	1211.31399	14.86660
	min	300.25600	1.22222	1210.27138	14.76490
5	max	95.39400	0.68891	990.40743	16.57930
	mean	95.39081	0.68890	990.40071	16.0784
	min	95.38850	0.68889	990.39583	15.99880
6	max	94.38110	0.68524	998.86839	17.75770
	mean	92.98190	0.68014	995.86625	17.37390
	min	91.61640	0.67513	992.89255	17.29040
7	max	92.58990	0.67871	1005.62368	19.57680
	mean	90.50440	0.67102	1001.04458	18.83530
	min	88.65040	0.66411	996.88430	18.61320
8	max	89.33390	0.66667	1009.03469	21.22680
	mean	88.58860	0.66388	1007.35074	20.33100
	min	87.64300	0.66033	1005.19372	20.00260
9	max	88.62520	0.66402	1018.04037	24.25370
	mean	87.17910	0.65858	1014.73360	23.80230
	min	85.81150	0.65339	1011.55547	23.36920
10	max	87.25230	0.65886	1025.50891	26.28920
	mean	86.03220	0.65423	1022.67837	25.61280
	min	85.40790	0.65185	1021.21448	25.21040
11	max	86.65010	0.65658	1034.72344	28.17860
	mean	85.48680	0.65216	1032.00669	27.57610
	min	84.59930	0.64876	1029.90905	27.17860
12	max	85.30540	0.65146	1042.18633	30.80330
	mean	84.81820	0.64960	1041.03508	29.63500
	min	84.28270	0.64755	1039.76204	29.12190
13	max	84.63180	0.64889	1051.19947	32.28520
	mean	84.07110	0.64673	1049.86338	31.50240
	min	82.63940	0.64120	1046.41095	30.74290
14	max	84.19150	0.64720	1060.75764	34.77100
	mean	82.86650	0.64208	1057.56916	33.92020
	min	82.04660	0.63890	1055.57052	33.07610
15	max	83.25720	0.64360	1069.12122	38.30550
	mean	82.38870	0.64023	1067.01347	36.66810
	min	81.19960	0.63559	1064.09134	35.32530
16	max	82.51000	0.64070	1077.91580	40.14510
	mean	81.45780	0.63660	1075.33608	39.01170
	min	79.82430	0.63019	1071.26440	37.55890
17	max	81.69280	0.63752	1086.52173	42.71040
	mean	81.03540	0.63495	1084.89769	41.70070
	min	80.42310	0.63255	1083.37318	40.29230

Table B.4: ϕ_3

p	stat	E	RMSE	BIC	Runtime
1	max	26.29369	0.36168	688.95492	11.50170
	mean	26.28733	0.36164	688.90632	11.48740
	min	26.28170	0.36160	688.86328	11.47150
2	max	22.80089	0.33680	670.91317	12.93180
	mean	22.79446	0.33676	670.85650	12.88210
	min	22.78670	0.33670	670.78802	12.82510
3	max	5.78345	0.16963	405.78818	14.56780
	mean	5.77796	0.16955	405.59737	14.48730
	min	5.77750	0.16954	405.58123	14.38230
4	max	3.06010	0.12339	288.44763	15.90660
	mean	2.96860	0.12153	282.34584	15.87040
	min	2.94570	0.12106	280.78930	15.82230
5	max	0.15328	0.02762	-302.72457	17.94660
	mean	0.14559	0.02691	-313.07710	17.7515
	min	0.14180	0.02656	-318.37460	17.45370
6	max	0.14150	0.02653	-308.19369	19.78080
	mean	0.13890	0.02629	-311.92133	19.48530
	min	0.13730	0.02614	-314.25010	19.04810
7	max	0.13940	0.02634	-300.59248	21.79590
	mean	0.13730	0.02614	-303.64349	21.47530
	min	0.13640	0.02605	-304.96538	21.10900
8	max	0.13720	0.02613	-293.18333	23.44180
	mean	0.13600	0.02601	-294.94908	23.06980
	min	0.13490	0.02591	-296.58143	22.26800
9	max	0.13710	0.02612	-282.72328	24.10460
	mean	0.13470	0.02589	-286.27304	23.57800
	min	0.13310	0.02573	-288.67486	22.95880
10	max	0.13540	0.02595	-274.62459	26.36140
	mean	0.13380	0.02580	-277.01392	25.53110
	min	0.13250	0.02567	-278.97638	24.92950
11	max	0.13340	0.02576	-267.00910	28.13490
	mean	0.13220	0.02565	-268.82538	27.47240
	min	0.13110	0.02554	-270.50484	26.66080
12	max	0.13250	0.02567	-257.76316	30.29060
	mean	0.13080	0.02551	-260.35872	29.60250
	min	0.12970	0.02540	-262.05623	28.33720
13	max	0.13170	0.02560	-248.37382	31.95970
	mean	0.12960	0.02539	-251.60465	31.16920
	min	0.12780	0.02522	-254.41589	30.55670
14	max	0.13080	0.02551	-239.14550	34.35050
	mean	0.12890	0.02532	-242.08664	33.50730
	min	0.12730	0.02517	-244.59721	32.05900
15	max	0.12900	0.02533	-231.32415	36.89300
	mean	0.12760	0.02520	-233.51747	36.04780
	min	0.12590	0.02503	-236.21337	35.26330
16	max	0.12860	0.02529	-221.34177	39.69190
	mean	0.12600	0.02504	-225.44717	38.44630
	min	0.12480	0.02492	-227.37063	37.59300
17	max	0.12660	0.02510	-213.88569	41.86790
	mean	0.12490	0.02493	-216.60303	41.12980
	min	0.12350	0.02479	-218.86875	40.12130

Table B.5: ϕ_4

p	stat	E	RMSE	BIC	Runtime
1	max	31.98679	0.39892	728.34975	11.51720
	mean	31.98245	0.39889	728.32245	11.49330
	min	31.97470	0.39885	728.27377	11.45930
2	max	15.93905	0.28160	598.94961	12.93790
	mean	15.93251	0.28154	598.86711	12.91410
	min	15.92800	0.28150	598.81023	12.87920
3	max	3.65039	0.13476	313.29473	14.46080
	mean	3.64966	0.13475	313.25451	14.41520
	min	3.64430	0.13465	312.95907	14.36300
4	max	1.30994	0.08073	117.90538	15.94110
	mean	1.30402	0.08055	116.99505	15.88900
	min	1.29860	0.08038	116.15830	15.82350
5	max	0.47814	0.04877	-74.06203	18.00490
	mean	0.46921	0.04832	-77.85184	17.8678
	min	0.46910	0.04831	-77.89853	17.65870
6	max	0.46550	0.04812	-68.84040	19.81860
	mean	0.46150	0.04792	-70.57504	19.63540
	min	0.45780	0.04772	-72.19302	19.16140
7	max	0.45820	0.04775	-61.41086	21.66300
	mean	0.45360	0.04750	-63.43896	21.46480
	min	0.44840	0.04723	-65.75650	21.05880
8	max	0.45090	0.04736	-54.03235	23.30190
	mean	0.44760	0.04719	-55.50882	23.12010
	min	0.44580	0.04709	-56.31876	22.82600
9	max	0.44560	0.04708	-45.80234	24.22650
	mean	0.44260	0.04693	-47.16015	23.64200
	min	0.43970	0.04677	-48.48148	23.16680
10	max	0.44270	0.04693	-36.50813	26.46510
	mean	0.43850	0.04671	-38.42417	25.75550
	min	0.43310	0.04642	-40.91479	25.26590
11	max	0.43680	0.04662	-28.59832	28.59210
	mean	0.43170	0.04634	-30.95897	27.75580
	min	0.42440	0.04595	-34.38693	26.81860
12	max	0.43370	0.04645	-19.42331	30.12000
	mean	0.42850	0.04617	-21.84784	29.46920
	min	0.42380	0.04592	-24.06468	29.02520
13	max	0.43050	0.04628	-10.30525	32.55870
	mean	0.42400	0.04593	-13.36324	31.55680
	min	0.41900	0.04566	-15.74761	30.89890
14	max	0.42290	0.04587	-3.27877	34.60420
	mean	0.41890	0.04565	-5.18898	34.07800
	min	0.41450	0.04541	-7.31139	33.28900
15	max	0.42150	0.04579	6.66133	37.60380
	mean	0.41690	0.04554	4.45568	36.47740
	min	0.41310	0.04533	2.61518	35.52860
16	max	0.41460	0.04542	13.95032	39.59980
	mean	0.40840	0.04508	10.92183	38.77020
	min	0.40250	0.04475	7.99688	38.21250
17	max	0.41000	0.04516	22.31436	43.02680
	mean	0.40200	0.04472	18.35365	42.00560
	min	0.39350	0.04425	14.05807	40.78980

Table B.6: ϕ_5

p	stat	E	RMSE	BIC	Runtime
1	max	1.70374	0.09207	138.91726	12.62820
	mean	1.70137	0.09200	138.63768	11.56010
	min	1.69850	0.09193	138.29868	11.45960
2	max	0.93014	0.06803	27.87070	12.96080
	mean	0.92261	0.06775	26.23577	12.91900
	min	0.92250	0.06775	26.21219	12.88460
3	max	0.75050	0.06111	-4.65709	15.10110
	mean	0.74680	0.06095	-5.65048	14.47650
	min	0.74560	0.06091	-5.97372	14.33380
4	max	0.13810	0.02621	-334.29556	16.05420
	mean	0.13740	0.02615	-335.31698	15.9292
	min	0.13730	0.02614	-335.46332	15.82970
5	max	0.13730	0.02614	-324.85671	17.21730
	mean	0.13670	0.02608	-325.73700	17.03670
	min	0.13550	0.02596	-327.50924	16.93070
6	max	0.13590	0.02600	-316.31015	18.58900
	mean	0.13480	0.02590	-317.94370	18.41980
	min	0.13330	0.02575	-320.19288	18.27440
7	max	0.13550	0.02596	-306.29602	20.11880
	mean	0.13440	0.02586	-307.93442	19.91420
	min	0.13230	0.02566	-311.09984	19.74580
8	max	0.13460	0.02588	-297.02892	21.70590
	mean	0.13270	0.02569	-299.88643	21.46350
	min	0.13120	0.02555	-302.17141	21.28370
9	max	0.13300	0.02572	-288.82593	24.96620
	mean	0.13210	0.02564	-290.19070	24.31690
	min	0.13110	0.02554	-291.71806	23.90380
10	max	0.13220	0.02565	-279.43199	26.69210
	mean	0.13050	0.02548	-282.03347	26.10060
	min	0.12850	0.02528	-285.13778	25.57410
11	max	0.13070	0.02550	-271.11905	28.85580
	mean	0.12950	0.02538	-272.97303	27.92780
	min	0.12800	0.02524	-275.31480	27.43970
12	max	0.13000	0.02543	-261.59185	30.91150
	mean	0.12840	0.02527	-264.08105	29.96890
	min	0.12630	0.02507	-267.39561	29.39250
13	max	0.12940	0.02537	-251.91508	32.61710
	mean	0.12750	0.02519	-254.88827	31.67540
	min	0.12540	0.02498	-258.22643	30.90780
14	max	0.12680	0.02512	-245.38823	34.74650
	mean	0.12550	0.02499	-247.45960	33.95020
	min	0.12160	0.02460	-253.80492	33.20850
15	max	0.12750	0.02519	-233.67505	37.46510
	mean	0.12450	0.02489	-238.46100	36.42890
	min	0.12110	0.02455	-244.02650	35.59890
16	max	0.12480	0.02492	-227.37063	43.27920
	mean	0.12300	0.02474	-230.29078	39.58670
	min	0.11980	0.02441	-235.58927	37.95510
17	max	0.12330	0.02477	-219.19452	47.98770
	mean	0.12130	0.02457	-222.48159	42.56110
	min	0.11840	0.02427	-227.34541	39.64010

Table B.7: ϕ_6

List of Figures

List of Algorithms

Bibliography

[Aarts et al., 1985] E. E. Aarts et al. "Statistical cooling: A general approach to combinatorial optimization problems". *Philips Journal of Research*, volume 40, no. 4, page 193, 1985.

[Akaike, 1974] H. Akaike. "A new look at the statistical model identification". *IEEE transactions on automatic control*, volume 19, no. 6, pages 716–723, 1974.

[Akaike, 1998] H. Akaike. "Information theory and an extension of the maximum likelihood principle". In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer, 1998.

[Akima, 1970] H. Akima. "A new method of interpolation and smooth curve fitting based on local procedures". *Journal of the ACM (JACM)*, volume 17, no. 4, pages 589–602, 1970.

[Bajaj et al., 1995] C. L. Bajaj, F. Bernardini, and G. Xu. "Automatic reconstruction of surfaces and scalar fields from 3D scans". In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 109–118. ACM, 1995.

[Barhak and Fischer, 2001] J. Barhak and A. Fischer. "Parameterization and reconstruction from 3D scattered points based on neural network and pde techniques". *IEEE Transactions on Visualization and Computer Graphics*, volume 7, no. 1, pages 1–16, 2001.

[Ben-Ameur, 2004] W. Ben-Ameur. "Computing the initial temperature of simulated annealing". *Computational Optimization and Applications*, volume 29, no. 3, pages 369–385, 2004.

[Bertsimas and Tsitsiklis, 1993] D. Bertsimas and J. Tsitsiklis. "Simulated annealing". *Statist. Sci.*, volume 8, no. 1, pages 10–15, 1993.

[Bézier, 1974] P. Bézier. "Mathematical and practical possibilities of UNISURF". In *Computer Aided Geometric Design*, pages 127 – 152. Academic Press, 1974.

[Blinn and Newell, 1976] J. F. Blinn and M. E. Newell. "Texture and reflection in computer generated images". *Communications of the ACM*, volume 19, no. 10, pages 542–547, 1976.

[Bohachevsky et al., 1986] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein. "Generalized simulated annealing for function optimization". *Technometrics*, volume 28, no. 3, pages 209–217, 1986.

[Brujic et al., 2011] D. Brujic, I. Ainsworth, and M. Ristic. "Fast and accurate NURBS fitting for reverse engineering". *The International Journal of Advanced Manufacturing Technology*, volume 54, no. 5-8, pages 691–700, 2011.

[Carlier et al., 2016] A. Carlier, K. Leonard, S. Hahmann, G. Morin, and M. Collins. "The 2D shape structure dataset: A user annotated open access database". *Computers & Graphics*, volume 58, pages 23–30, 2016.

[Cignoni et al., 2008] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. "Meshlab: an open-source mesh processing tool.". In *Eurographics Italian Chapter Conference*, volume 2008, pages 129–136. 2008.

[Corana et al., 1987] A. Corana, M. Marchesi, C. Martini, and S. Ridella. "Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm". *ACM Transactions on Mathematical Software*, volume 13, no. 3, pages 262–280, 1987.

[Cosido et al., 2014] O. Cosido, R. Catuogno, A. Gálvez, A. Iglesias, C. Loucera, V. Cappellini, M. Campi, and E. Sainz. "Documentación tridimensional del patrimonio histórico mediante hibridación de técnicas de visión artificial e ingeniería inversa: el palacio de la Magdalena Santander". In *Congreso Latinoamericano sobre Patología de la Construcción, Tecnología de la Rehabilitación y Gestión del Patrimonio: REHABEND 2014*, pages 2207–2214. 2014.

[Cosido et al., 2013] O. Cosido, C. Loucera, and A. Iglesias. "Automatic calculation of bicycle routes by combining meta-heuristics and GIS techniques within the framework of smart cities". In *New Concepts in Smart Cities: Fostering Public and Private Alliances (SmartMILE), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[Cox et al., 1990] M. Cox, P. Harris, and H. M. Jones. "A knot placement strategy for least squares spline fitting based on the use of local polynomial approximations". *Algorithms for Approximation, II*, pages 37–45, 1990.

[Cox, 1990] M. G. Cox. *Algorithms for spline curves and surfaces*. National Physical Laboratory, 1990.

[Cressie, 1990] N. Cressie. "The origins of kriging". *Mathematical geology*, volume 22, no. 3, pages 239–252, 1990.

[Crow, 1987] F. Crow. "The origins of the teapot". *IEEE Computer Graphics and Applications*, volume 7, no. 1, pages 8–19, 1987.

[de Boor, 1978] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag GmbH, 1978.

[de Boor and Rice, 1968] C. de Boor and J. R. Rice. “Least squares cubic spline approximation, ii-variable knots”. 1968.

[De Castro and Von Zuben, 2000] L. N. De Castro and F. J. Von Zuben. “The clonal selection algorithm with engineering applications”. In *Proceedings of GECCO*, volume 2000, pages 36–39. 2000.

[Dekkers and Aarts, 1991] A. Dekkers and E. Aarts. “Global optimization and simulated annealing”. *Mathematical programming*, volume 50, no. 1, pages 367–393, 1991.

[Dierckx, 1981] P. Dierckx. “An improved algorithm for curve fitting with spline functions”. 1981.

[Dierckx, 1995] P. Dierckx. *Curve and surface fitting with splines*. Oxford University Press, 1995.

[Eck and Hoppe, 1996] M. Eck and H. Hoppe. “Automatic reconstruction of B-spline surfaces of arbitrary topological type”. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’96, pages 325–334. ACM, 1996.

[Egea et al., 2009] J. A. Egea, E. Balsa-Canto, M.-S. G. Garcia, and J. R. Banga. “Dynamic optimization of nonlinear processes with an enhanced scatter search method”. *Industrial & Engineering Chemistry Research*, volume 48, no. 9, pages 4388–4401, 2009.

[Egea et al., 2010] J. A. Egea, R. Martí, and J. R. Banga. “An evolutionary method for complex-process optimization”. *Computers & Operations Research*, volume 37, no. 2, pages 315–324, 2010.

[Epstein, 1976] M. Epstein. “On the influence of parametrization in parametric interpolation”. *SIAM Journal on Numerical Analysis*, volume 13, no. 2, pages 261–268, 1976.

[Erol and Eksin, 2006] O. K. Erol and I. Eksin. “A new optimization method: Big BangâĂŞBig Crunch”. *Advances in Engineering Software*, volume 37, no. 2, pages 106 – 111, 2006.

[Farin, 2002] G. Farin. *Curves and Surfaces for CAGD*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2002.

[Foley, 1987] T. A. Foley. “Interpolation and approximation of 3-D and 4-D scattered data”. *Computers & Mathematics with Applications*, volume 13, no. 8, pages 711–740, 1987.

[Forrest, 1972] A. R. Forrest. “Interactive interpolation and approximation by bézier polynomials”. *The Computer Journal*, volume 15, no. 1, pages 71–79, 1972.

[Gálvez and Iglesias, 2011] A. Gálvez and A. Iglesias. "Efficient particle swarm optimization approach for data fitting with free knot B-splines". *Computer-Aided Design*, volume 43, no. 12, pages 1683–1692, 2011.

[Gálvez and Iglesias, 2012] A. Gálvez and A. Iglesias. "Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points". *Information Sciences*, volume 192, pages 174–192, 2012.

[Gálvez and Iglesias, 2013] A. Gálvez and A. Iglesias. "An electromagnetism-based global optimization approach for polynomial Bézier curve parameterization of noisy data points". In *Cyberworlds (CW), 2013 International Conference on*, pages 259–266. IEEE, 2013.

[Gálvez and Iglesias, 2013] A. Gálvez and A. Iglesias. "Firefly Algorithm for explicit B-spline curve fitting to data points". *Mathematical Problems in Engineering*, volume 2013, pages 1–12, 2013.

[Gálvez and Iglesias, 2016] A. Gálvez and A. Iglesias. "New memetic self-adaptive firefly algorithm for continuous optimisation". *International Journal of Bio-Inspired Computation*, volume 8, no. 5, page 300, 2016.

[Gálvez et al., 2015] A. Gálvez, A. Iglesias, A. Avila, C. Otero, R. Arias, and C. Manchado. "Elitist clonal selection algorithm for optimal choice of free knots in B-spline data fitting". *Applied Soft Computing*, volume 26, pages 90–106, 2015.

[Gálvez et al., 2007] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, and J. Espinola. "Bézier curve and surface fitting of 3d point clouds through genetic algorithms, functional networks and least-squares approximation". In O. Gervasi and M. L. Gavrilova, editors, *Computational Science and Its Applications – ICCSA 2007: International Conference, Kuala Lumpur, Malaysia, August 26-29, 2007. Proceedings, Part II*, pages 680–693. Springer Berlin Heidelberg, 2007.

[Geman and Geman, 1984] S. Geman and D. Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume PAMI-6, no. 6, pages 721–741, 1984.

[Gu and Yan, 1995] P. Gu and X. Yan. "Neural network approach to the reconstruction of freeform surfaces for reverse engineering". *Computer-Aided Design*, volume 27, no. 1, pages 59–64, 1995.

[Guo, 1997] B. Guo. "Surface reconstruction: from points to splines". *Computer-Aided Design*, volume 29, no. 4, pages 269–277, 1997.

[Hajek, 1988] B. Hajek. "Cooling schedules for optimal annealing". *Mathematics of operations research*, volume 13, no. 2, pages 311–329, 1988.

[Harris and Ifeachor, 1998] S. Harris and E. Ifeachor. "Automatic design of frequency sampling filters by hybrid genetic algorithm techniques". *IEEE Transactions on Signal Processing*, volume 46, no. 12, pages 3304–3314, 1998.

[Hart et al., 2004] W. E. Hart, N. Krasnogor, and J. E. Smith. *Recent advances in memetic algorithms*, volume 166. Springer Science & Business Media, 2004.

[Hasegawa et al., 2013] A. Y. Hasegawa, R. S. Rosso, and M. S. G. Tsuzuki. "Bezier curve fitting with a parallel differential evolution algorithm". *IFAC Proceedings Volumes*, volume 46, no. 7, pages 233–238, 2013.

[Hayes and Halliday, 1974] J. G. Hayes and J. Halliday. "The least-squares fitting of cubic spline surfaces to general data sets". *IMA Journal of Applied Mathematics*, volume 14, no. 1, pages 89–103, 1974.

[Hedar and Fukushima, 2004] A.-R. Hedar and M. Fukushima. "Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization". *Optimization Methods and Software*, volume 19, no. 3-4, pages 291–308, 2004.

[Hoff III et al., 1999] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. "Fast computation of generalized Voronoi diagrams using graphics hardware". In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press/Addison-Wesley Publishing Co., 1999.

[Hoffmann, 2005] M. Hoffmann. "Numerical control of Kohonen neural network for scattered data approximation". *Numerical Algorithms*, volume 39, no. 1, pages 175–186, 2005.

[Hoffmann and Varady, 1998] M. Hoffmann and L. Varady. "Free-form surfaces for scattered data by neural networks". *Journal for Geometry and Graphics*, volume 2, no. 1, pages 1–6, 1998.

[Hoppe et al., 1992] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface reconstruction from unorganized points". *SIGGRAPH Comput. Graph.*, volume 26, no. 2, pages 71–78, 1992.

[Hutchinson and Gessler, 1994] M. Hutchinson and P. Gessler. "Splines- more than just a smooth interpolator". *Geoderma*, volume 62, no. 1-3, pages 45–67, 1994.

[Hutchinson, 1995] M. F. Hutchinson. "Interpolating mean rainfall using thin plate smoothing splines". *International journal of geographical information systems*, volume 9, no. 4, pages 385–403, 1995.

[Iglesias et al., 2004] A. Iglesias, G. Echevarría, and A. Gálvez. "Functional networks for B-spline surface reconstruction". *Future Generation Computer Systems*, volume 20, no. 8, pages 1337–1353, 2004.

[Iglesias and Gálvez, 2008] A. Iglesias and A. Gálvez. "Curve fitting with RBS functional networks". In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, volume 1, pages 299–306. IEEE, 2008.

[Iglesias and Gálvez, 2015] A. Iglesias and A. Gálvez. "Memetic firefly algorithm for data fitting with rational curves". In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 507–514. IEEE, 2015.

[Iglesias and Gálvez, 2016] A. Iglesias and A. Gálvez. "Memetic electromagnetism algorithm for surface reconstruction with rational bivariate Bernstein basis functions". *Natural Computing*, pages 1–15, 2016.

[Iglesias and Gálvez, 2016] A. Iglesias and A. Gálvez. "Nature-inspired swarm intelligence for data fitting in reverse engineering: Recent advances and future trends". In *Nature-Inspired Computation in Engineering*, pages 151–175. Springer Science Business Media, 2016.

[Iglesias et al., 2013] A. Iglesias, A. Gálvez, and A. Avila. "Discrete Bézier curve fitting with artificial immune systems". In *Intelligent Computer Graphics 2012*, pages 59–75. Springer, 2013.

[Iglesias et al., 2015a] A. Iglesias, A. Gálvez, and M. Collantes. "A bat algorithm for polynomial Bézier surface parameterization from clouds of irregularly sampled data points". In *Natural Computation (ICNC), 2015 11th International Conference on*, pages 1034–1039. IEEE, 2015.

[Iglesias et al., 2016a] A. Iglesias, A. Gálvez, and M. Collantes. "Four adaptive memetic bat algorithm schemes for bézier curve parameterization". In *Transactions on Computational Science XXVIII*, pages 127–145. Springer, 2016.

[Iglesias et al., 2015b] A. Iglesias, A. Gálvez, and C. Loucera. "A simulated annealing approach for data fitting with Bézier surfaces". In *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*, page 4. ACM, 2015.

[Iglesias et al., 2016b] A. Iglesias, A. Gálvez, and C. Loucera. "Two simulated annealing optimization schemas for rational bézier curve fitting in the presence of noise". *Mathematical Problems in Engineering*, volume 2016, pages 1–17, 2016.

[Ingber, 1993a] L. Ingber. "Adaptive simulated annealing (ASA)". Technical report, Pasadena, CA, 1993.

[Ingber, 1993b] L. Ingber. "Simulated annealing: Practice versus theory". *Mathematical and Computer Modelling*, volume 18, no. 11, pages 29–57, 1993.

[Ingber, 1996] L. Ingber. "Adaptive simulated annealing (ASA): Lessons learned". In *Control and Cybernetics*. 1996.

[Johnson et al., 1989] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. "Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning". *Operations research*, volume 37, no. 6, pages 865–892, 1989.

[Johnson et al., 1991] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. "Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning". *Operations research*, volume 39, no. 3, pages 378–406, 1991.

[Johnson, 2010] S. G. Johnson. "The NLopt nonlinear-optimization package". 2010.

[Jones et al., 2001–] E. Jones, T. Oliphant, P. Peterson, et al. "SciPy: Open source scientific tools for Python". 2001–.

[Jupp, 1978] D. L. B. Jupp. "Approximation to data by splines with free knots". *SIAM Journal on Numerical Analysis*, volume 15, no. 2, pages 328–343, 1978.

[Kazhdan and Hoppe, 2013] M. Kazhdan and H. Hoppe. "Screened Poisson surface reconstruction". *ACM Transactions on Graphics (TOG)*, volume 32, no. 3, page 29, 2013.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simmulated annealing". *Science*, volume 220, no. 4598, pages 671–680, 1983.

[Kreylos and Hamann, 2001] O. Kreylos and B. Hamann. "On simulated annealing and the construction of linear spline approximations for scattered data". *IEEE Transactions on Visualization and Computer Graphics*, volume 7, no. 1, pages 17–31, 2001.

[Kumar et al., 2004] G. S. Kumar, P. K. Kalra, and S. G. Dhande. "Curve and surface reconstruction from points: an approach based on self-organizing maps". *Applied Soft Computing*, volume 5, no. 1, pages 55–66, 2004.

[Lane and Riesenfeld, 1983] J. M. Lane and R. F. Riesenfeld. "A geometric proof for the variation diminishing property of B-spline approximation". *Journal of Approximation Theory*, volume 37, no. 1, pages 1–4, 1983.

[Laurent-Gengoux and Mekhilef, 1993] P. Laurent-Gengoux and M. Mekhilef. "Optimization of a NURBS representation". *Comput. Des.*, volume 25, no. 11, pages 699–710, 1993.

[Lee, 1989] E. T. Lee. "Choosing nodes in parametric curve interpolation". *Computer-Aided Design*, volume 21, no. 6, pages 363–370, 1989.

[Locatelli, 2000] M. Locatelli. "Simulated annealing algorithms for continuous global optimization: convergence conditions". *Journal of Optimization Theory and applications*, volume 104, no. 1, pages 121–133, 2000.

[Lorentz, 2012] G. G. Lorentz. *Bernstein polynomials*. American Mathematical Soc., 2012.

[Loucera et al., 2014] C. Loucera, A. Gálvez, and A. Iglesias. "Simulated annealing algorithm for Bézier curve approximation". In *Cyberworlds (CW), 2014 International Conference on*, pages 182–189. IEEE, 2014.

[Loucera et al., 2017a] C. Loucera, A. Iglesias, and A. Gálvez. "Memetic simulated annealing for data approximation with local-support curves". *Procedia Computer Science*, 2017.

[Loucera et al., 2017b] C. Loucera, A. Iglesias, and A. Gálvez. "Simulated annealing and natural neighbor for rational Bézier surface reconstruction from scattered data points". In *International Conference on Harmony Search Algorithm*, pages 354–364. Springer, 2017.

[Ma and Kruth, 1995] W. Ma and J.-P. Kruth. "Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces". *Computer-Aided Design*, volume 27, no. 9, pages 663–675, 1995.

[Malchenko, 2005] S. Malchenko. "Model human heart shape variation during cardiac cycle". 2005.

[Marsh and Cormier, 2001] L. C. Marsh and D. R. Cormier. *Spline regression models*, volume 137. Sage, 2001.

[Mathworks, 2001–] Mathworks. "Curve fitting toolbox™". 2001–.

[Metropolis et al., 1953] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. "Equation of state calculations by fast computing machines". *The Journal of Chemical Physics*, volume 21, no. 6, page 1087, 1953.

[Nelder and Mead, 1965] J. A. Nelder and R. Mead. "A simplex method for function minimization". *The computer journal*, volume 7, no. 4, pages 308–313, 1965.

[Nielson, 1993] G. M. Nielson. "Scattered data modeling". *IEEE Computer Graphics and Applications*, volume 13, no. 1, pages 60–70, 1993.

[Osman and Kelly, 2012] I. H. Osman and J. P. Kelly. *Meta-heuristics: theory and applications*. Springer Science & Business Media, 2012.

[Park, 2004] H. Park. "An error-bounded approximate method for representing planar curves in B-splines". *Computer Aided Geometric Design*, volume 21, no. 5, pages 479–497, 2004.

[Park and Kim, 1996] H. Park and K. Kim. "Smooth surface approximation to serial cross-sections". *Computer-Aided Design*, volume 28, no. 12, pages 995–1005, 1996.

[Park and Lee, 2007] H. Park and J.-H. Lee. "B-spline curve fitting based on adaptive curve refinement using dominant points". *Computer-Aided Design*, volume 39, no. 6, pages 439–451, 2007.

[Park et al., 2006] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. Hamann. "Discrete Sibson interpolation". *IEEE Transactions on Visualization and Computer Graphics*, volume 12, no. 2, pages 243–253, 2006.

[Pauly et al., 2002] M. Pauly, M. Gross, and L. P. Kobbelt. "Efficient simplification of point-sampled surfaces". In *Proceedings of the conference on Visualization'02*, pages 163–170. IEEE Computer Society, 2002.

[Petalas et al., 2007] Y. G. Petalas, K. E. Parsopoulos, and M. N. Vrahatis. "Memetic particle swarm optimization". *Annals of Operations Research*, volume 156, no. 1, pages 99–127, 2007.

[Piegl, 1986] L. Piegl. "The sphere as a rational Bézier surface". *Computer Aided Geometric Design*, volume 3, no. 1, pages 45 – 52, 1986.

[Piegl and Tiller, 1987] L. Piegl and W. Tiller. "Curve and surface constructions using rational B-splines". *Computer-Aided Design*, volume 19, no. 9, pages 485–498, 1987.

[Piegl and Tiller, 1995] L. Piegl and W. Tiller. *The NURBS Book*. Springer Berlin Heidelberg, 1995.

[Piegl and Tiller, 2012] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[Piegl and Tiller, 2001] L. A. Piegl and W. Tiller. "Parametrization for surface fitting in reverse engineering". *Computer-Aided Design*, volume 33, no. 8, pages 593–603, 2001.

[Pottmann et al., 2005] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang. "Industrial geometry: recent advances and applications in CAD". *Computer-Aided Design*, volume 37, no. 7, pages 751–766, 2005.

[Powell, 1970] M. Powell. "Curve fitting by splines in one variable". In J. Hayes, editor, *Numerical Approximation to Functions and Data*. The Athlone Press, 1970.

[Powell, 1994] M. J. D. Powell. "A direct search optimization method that models the objective and constraint functions by linear interpolation". In S. Gomez and J.-P. Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer Netherlands, 1994.

[Prautzsch and Gallagher, 1992] H. Prautzsch and T. Gallagher. "Is there a geometric variation diminishing property for B-spline or Bézier surfaces?". *Computer aided geometric design*, volume 9, no. 2, pages 119–124, 1992.

[Rashedi et al., 2009] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi. "GSA: a gravitational search algorithm". *Information sciences*, volume 179, no. 13, pages 2232–2248, 2009.

[Reinsch, 1967] C. H. Reinsch. "Smoothing by spline functions". *Numerische mathematik*, volume 10, no. 3, pages 177–183, 1967.

[Renner, 1982] G. Renner. "Method of shape description for mechanical engineering practice". *Computers in Industry*, volume 3, no. 1-2, pages 137–142, 1982.

[Rice and Saloin, 1969] J. R. Rice and M. Saloin. *The approximation of functions*, volume 2. Addison-Wesley Reading, Mass., 1969.

[Rios and Sahinidis, 2012] L. M. Rios and N. V. Sahinidis. "Derivative-free optimization: a review of algorithms and comparison of software implementations". *Journal of Global Optimization*, volume 56, no. 3, pages 1247–1293, 2012.

[Rogers, 2000] D. F. Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2000.

[Rogers and Fog, 1989] D. F. Rogers and N. Fog. "Constrained B-spline curve and surface fitting". *Computer-Aided Design*, volume 21, no. 10, pages 641–648, 1989.

[Rusu and Cousins, 2011] R. B. Rusu and S. Cousins. "3D is here: Point Cloud Library (PCL)". In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011.

[Rusu et al., 2008] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. "Towards 3D point cloud based object maps for household environments". *Robotics and Autonomous Systems*, volume 56, no. 11, pages 927–941, 2008.

[Salamon et al., 2002] P. Salamon, P. Sibani, and R. Frost. *Facts, Conjectures, and Improvements for Simulated Annealing*. Society for Industrial & Applied Mathematics (SIAM), 2002.

[Sarkar and Menq, 1991a] B. Sarkar and C.-H. Menq. "Parameter optimization in approximating curves and surfaces to measurement data". *Computer Aided Geometric Design*, volume 8, no. 4, pages 267–290, 1991.

[Sarkar and Menq, 1991b] B. Sarkar and C.-H. Menq. "Smooth-surface approximation and reverse engineering". *Computer-Aided Design*, volume 23, no. 9, pages 623–628, 1991.

[Schuur, 1997] P. C. Schuur. "Classification of acceptance criteria for the simulated annealing algorithm". *Mathematics of Operations Research*, volume 22, no. 2, pages 266–275, 1997.

[Schwarz, 1978] G. Schwarz. "Estimating the dimension of a model". *The Annals of Statistics*, volume 6, no. 2, pages 461–464, 1978.

[Sen and Zheng, 1992] S. Sen and S.-Q. Zheng. "Near-optimal triangulation of a point set by simulated annealing". In *Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's*, pages 1000–1008. ACM, 1992.

[Sibson, 1981] R. Sibson. "A brief description of natural neighbor interpolation". In V. Barnet, editor, *Interpreting Multivariate Data*, pages 21–36. Wiley, 1981.

[Sklar, 2001] B. Sklar. *Digital communications*, volume 2. Prentice Hall NJ, 2001.

[Suman and Kumar, 2005] B. Suman and P. Kumar. "A survey of simulated annealing as a tool for single and multiobjective optimization". *J Oper Res Soc*, volume 57, no. 10, pages 1143–1160, 2005.

[Szu and Hartley, 1987] H. Szu and R. Hartley. "Fast simulated annealing". *Physics letters A*, volume 122, no. 3, pages 157–162, 1987.

[Tait et al., 2006] A. Tait, R. Henderson, R. Turner, and X. Zheng. "Thin plate smoothing spline interpolation of daily rainfall for New Zealand using a climatological rainfall surface". *International Journal of Climatology*, volume 26, no. 14, pages 2097–2115, 2006.

[Thakoor et al., 2007] N. Thakoor, J. Gao, and S. Jung. "Hidden Markov model-based weighted likelihood discriminant for 2-D shape classification". *IEEE Transactions on Image Processing*, volume 16, no. 11, pages 2707–2719, 2007.

[Tsallis and Stariolo, 1996] C. Tsallis and D. A. Stariolo. "Generalized simulated annealing". *Physica A: Statistical Mechanics and its Applications*, volume 233, no. 1-2, pages 395–406, 1996.

[Ülker and Arslan, 2009] E. Ülker and A. Arslan. "Automatic knot adjustment using an artificial immune system for B-spline curve approximation". *Information Sciences*, volume 179, no. 10, pages 1483–1494, 2009.

[Valenzuela et al., 2013] O. Valenzuela, B. Delgado-Marquez, and M. Pasadas. "Evolutionary computation for optimal knots allocation in smoothing splines". *Applied Mathematical Modelling*, volume 37, no. 8, pages 5851–5863, 2013.

[Valenzuela and Pasadas, 2010] O. Valenzuela and M. Pasadas. "Using simulated annealing for knot placement for cubic spline approximation". *Mathematical Models for Engineering Science*, volume 22, pages 1012–1019, 2010.

[van Laarhoven and Aarts, 1987] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Springer Science + Business Media, 1987.

[Vanderbilt and Louie, 1984] D. Vanderbilt and S. G. Louie. "A Monte Carlo simulated annealing approach to optimization over continuous variables". *Journal of Computational Physics*, volume 56, no. 2, pages 259–271, 1984.

[Warren and Weimer, 2001] J. Warren and H. Weimer. *Subdivision methods for geometric design: A constructive approach*. Morgan Kaufmann, 2001.

[Wolpert and Macready, 1997] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". *IEEE transactions on evolutionary computation*, volume 1, no. 1, pages 67–82, 1997.

[Yang et al., 2005] W. Y. Yang, W. Cao, T.-S. Chung, and J. Morris. *Applied numerical methods using MATLAB*. John Wiley & Sons, 2005.

[Yang, 2010] X.-S. Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[Yang, 2014] X.-S. Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

[Yannis Haralambous, 2007] P. H. Yannis Haralambous. *Fonts & Encodings*. O'Reilly UK Ltd., 2007.

[Yoshimoto et al., 2003] F. Yoshimoto, T. Harada, and Y. Yoshimoto. "Data fitting with a spline using a real-coded genetic algorithm". *Computer-Aided Design*, volume 35, no. 8, pages 751–760, 2003.

[Yoshimoto et al., 1999] F. Yoshimoto, M. Moriyama, and T. Harada. "Automatic knot placement by a genetic algorithm for data fitting with a spline". In *Shape Modeling and Applications, 1999. Proceedings. Shape Modeling International'99. International Conference on*, pages 162–169. IEEE, 1999.