(To be submitted to . . .)

Roadmap to spline-fitting potentials in high dimensions

M. Patrício†, Author†, F. Patrício† and A.J.C. Varandas[1⋆]

†CMUC, Departamento de Matemática, Universidade de Coimbra
3004-535 Coimbra, Portugal
⋆Departamento de Química, Universidade de Coimbra
3004-535 Coimbra, Portugal

(Draft: January , 2010)

Abstract

Use of the theory of splines to approximate the potential energy surface in molecular dynamics is examined. It is envisaged that such an approximation should be able to accurately capture the potentials' behavior and be computationally cost effective, both for one-dimensional and n-dimensional problems with n arbitrary.

1 Introduction

A key step in reaction dynamics is to obtain the potential energy surface (this stands generally for potential energy curve or potential energy hypersurface, hereinafter denoted by the acronym PES) that governs the motion of the nuclei for any arrangement of the latter. Despite important progress towards dynamics on-the-fly (the electronic Schrödinger equation is then solved simultaneously to the corresponding equation for the nuclei once assumed the Born-Oppenheimer approximation for their separation), the traditional scheme of modeling the PES prior to starting the dynamics studies remains by far the most used approach especially for small systems (up to 4 atoms or so). In this approach, the two main streams to the PES consist of modeling it with a suitable functional form, often and hopefully originated from physically motivated arguments,[?,?,?,?] and of using semi-numerical black-box fitting approaches based on mathematical splines or related interpolation approaches (see later). Although the latter are unbiased and simple to use, they face the serious problem of dimensionality,

[1]Corresponding author: e-mail: varandas@qtvs1.qui.uc.pt.

so-called X^{3N-6}-explosion (this gives the number of geometries at which the electronic Schrödinger equation has to be solved if X is the number of points required to spline-fit a cut of the PES function in one-dimension, $1D$, and N the number of atoms involved). The major motivation of this work is to examine the capabilities of two types of splines (natural vs shape-preserving) when aiming at high-dimensionality problems.

There have been already some works in this area. A piecewise interpolation method has been presented in,[?] where the function and gradient data are reproduced on a multidimensional rectangular grid using a local reduced piecewise Hermite interpolation. More recently, a fitting strategy that is based upon an efficient placement of the ab initio data points is presented in,[?] where a vast number of references to literature on interpolation methods can also be found.

Specifically, we examine here the problem of fitting a PES, by comparing the traditional cubic splines (natural cubic splines) to shape-preserving splines. The theory and properties of both spline-types will be first dealt with for the $1D$ case. This will allow a comparison of the number of operations that are involved, which can give an indication of the required computational labor. It will be shown that shape-preserving splines (or simply shape splines), being a local method, are computationally quite demanding, with the required effort increasing linearly with the number of patches. However, they behave much less critically than the natural cubic splines with respect to the issue of dimensionality. Moreover, shape preserving methods yield visually pleasing plots, as properties found in the discrete data such as monotonicity are maintained in the approximation, meaning that shape preserving splines are monotonic when the data is monotonic and present local extrema at the data points that are a local extreme. Conversely, it is shown that in certain conditions the natural splines offer a greater accuracy than the shape-preserving splines at regions of strong curvature when $f \in C^k$,

$k \geq 2$.

As noted above, for medium-size molecules (or even relatively modest ones with 4 or more atoms), the modeling of an analytic function can be a mammoth task, with simplicity calling for strictly numerical techniques to face the problem. This implies in turn a great computational effort if a high-accuracy is in demand, particularly at regions of large curvature. As a result, refinements of the grid will be required until a predefined accuracy is obtained. The use of non-uniform meshes will then be also examined here, aiming at cost-effective techniques that can be used when high accuracy is at demand. Bearing this in mind, an algorithm to fit potentials is proposed in section 4.

The paper is organized as follows. In section 2, the notion of shape preserving spline is generalized to the $2D$ case. Test cases and computational examples will then be reported in section 3, namely for the popular Morse curve ($1D$) and a more complicated $2D$ numerical potential taken from the literature. Section 4 gathers some conclusions.

2 Functions of one variable

Consider a set of points $\{x_i, \quad i \in I\}$, where $I = \{0, \ \ldots, \ n\}$, and a function $z = z(x)$. Let $h_k = x_{k+1} - x_k$ and $h = \max_k h_k$, for $k = 0, \ \ldots, \ n-1$.

2.1 Natural vs shape-preserving splines

The classical natural cubic interpolating spline for the function $z = z(x)$ at the points $\{x_i, \quad i \in I\}$ is a function S that satisfies the conditions

- $S_i := S_{|_{[x_{i-1}, x_i]}}$ is a cubic polynomial, for $i = 1, \ \ldots, n$;
- $S(x_i) = z_i := z(x_i)$, for $i \in I$;
- $S \in C^2[x_0, x_n]$;
- $S''(x_0) = S''(x_n) = 0$.

Determining such a function S is equivalent to computing the $4n$ coefficients of the polynomials

$$S_i(x) = a_i(x - x_{i-1})^3 + b_i(x - x_{i-1})^2 + c_i(x - x_{i-1}) + d_i \tag{1}$$

that satisfy the following set of equations

$$\begin{aligned} S_i(x_{i-1}) &= z_{i-1}; \quad S_i(x_i) = z_i, \quad \text{for } i = 1, \ldots, n; & (2)\\ S_i'(x_i) &= S_{i+1}'(x_i), \quad S_i''(x_i) = S_{i+1}''(x_i) \quad \text{for } i = 1, \ldots, n-1; & (3)\\ S_1''(x_0) &= S_n''(x_n) = 0. & (4) \end{aligned}$$

In working out this set of equations, it can be shown that only n unknowns remain to be determined, from the resolution of a system of linear equations with a $n \times n$ matrix. This analysis may turn out to be quite useful when the dimensionality increases or there is a great number of patches, as it will reduce the computational cost. For the sake of clarity, we disregard it for now and leave it in the [anexo].

An alternative to natural cubic splines, still to interpolate z, are the shape-preserving splines P. These are defined by

- $P_i := P_{|_{[x_{i-1}, x_i]}}$ is a cubic polynomial, for $i = 1, \ldots, n$;
- $P(x_i) = z_i := z(x_i)$, for each $i \in I$;
- $P'(x_i) = \Delta_i$, for each $i \in I$.

For each value of $k = 1, \ldots, n-1$, the quantity Δ_k is an approximation to the derivative of z at x_k, given by

$$\Delta_i := \begin{cases} \dfrac{w_i^1 + w_i^2}{\left(\frac{w_i^1}{\delta_{i-1}} + \frac{w_i^2}{\delta_i}\right)}, & \text{if } \delta_i \delta_{i-1} > 0, \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where we define

$$\delta_i := \frac{z_{i+1} - z_i}{h_i}, \quad w_i^1 = 2h_i + h_{i-1}, \quad w_i^2 = h_i + 2h_{i-1}. \tag{6}$$

The idea of this approximation is to prevent the function values from overshooting the data by setting the slope of the interpolation spline as the harmonic mean of finite differences approximations to the derivatives of f. For that purpose, whenever the forward and backward approximations for the derivative at x_k, given respectively by δ_k and δ_{k-1}, have opposite signs, or one of the approximations is zero, the slope of the interpolation spline at $x = x_k$ is set to be zero. If instead they have the same signs and in the particular case of $h_k = h_{k-1}$, then the approximation for the derivative expressed by (5) is the harmonic mean of the discrete slopes:

$$\Delta_k := \frac{2}{\frac{1}{\delta_{k-1}} + \frac{1}{\delta_k}}.$$

The approximations above for the slopes of z are valid for the points x_i, with $i = 1, \ldots, n-1$. Similar approximation can be employed for the endpoints, cf.[?] The shape preserving spline P may then be interpreted as an Hermite interpolating spline fitting a function with values z_i and derivatives Δ_i at the points x_i. We denote the spline at the interval $[x_i, x_{i+1}]$ by

$$P_i(x) = \overline{a}_i(x - x_{i-1})^3 + \overline{b}_i(x - x_{i-1})^2 + \overline{c}_i(x - x_{i-1}) + \overline{d}_i. \tag{7}$$

The coefficients of the polynomials may now be determined from the linear system related to the equations

$$S_i(x_{i-1}) = z_{i-1}, \qquad S_i'(x_{i-1}) = z_{i-1}', \tag{8}$$

$$S_i(x_i) = z_i, \qquad S_i'(x_i) = z_i', \tag{9}$$

for $i = 1, \ldots, n$. Again, instead of solving a linear system of order n, it is easy to see that [remover? the unknowns $\overline{c}_i$ and $\overline{d}_i$ may be computed from the equations

in (8) independently from the other unknowns $\overline{a}_i$ and $\overline{b}_i$. The latter may in turn be determined from the coupled equations expressed by (9). endremover?] Finding the shape preserving spline P then translates into solving a system of linear equations of order 2 for each patch where z is to be approximated by a cubic polynomial, see [anexo].

2.2 Error analysis and computational complexity

It is desirable to find estimates for the errors arising in the aproximation of a function by natural and shape-preserving splines. Upper bounds for the infinity and L^2 norms of the errors of the approximation of a given function $f \in C^r[a,b]$, with $r \in \mathbb{N} \setminus \{1\}$ by polynomial cubic splines can be found in in.[?] The results refer to cubic polynomials that only differ from the natural splines we have defined at the end-points, where instead of having the second derivative equal to zero, the derivatives of the polynomials are required to equal the derivatives of f. It can be shown that the bounds are still valid for natural splines. Let S then be the natural interpolating spline for f. The following results hold.

a) if $r \geq 2$, one has

$$\|f - S\|_\infty \leq \sqrt{8}\|f''\|_2 \; h^{3/2} \quad \text{and} \quad \|f' - S'\|_\infty \leq \sqrt{8}\|f''\|_2 \; h^{1/2}$$

as well as

$$\|f - S\|_2 \leq 16\|f''\|_2 \; h^2 \quad \text{and} \quad \|f' - S'\|_2 \leq 4\|f''\|_2 \; h$$

b) if $r \geq 4$, one has

$$\|f - S\|_\infty \leq 16\sqrt{2}\|f^{(4)}\|_2 \; h^{7/2} \quad \text{and} \quad \|f' - S'\|_\infty \leq 16\sqrt{2}\|f^{(4)}\|_2 \; h^{5/2}.$$

To enable a comparison between approximating f with a natural cubic spline S and a shape-preserving spline P, it is important to also find upper bounds for the error of the aproximation of f with shape-preserving splines. The result is established in the

following theorem.

Theorem: Let $f \in C^2[a,b]$ and P be the corresponding shape-preserving interpolating spline. Then

$$\|f - P\|_\infty \leq \frac{1}{8}\|f''\|_\infty h^2 + 4h \ \max_i |\Delta_i - z_i'|.$$

Sketch of proof: Let H be the Hermite cubic spline that satisfies

$$H(x_i) = z_i; \qquad H'(x_i) = z_i', \qquad i = 0, \ \ldots, \ n.$$

Then one has

$$\|f - H\|_\infty \leq \frac{1}{8}\|f''\|_\infty h^2.$$

Moreover, it is easy to show that

$$\begin{aligned} |H(x) - P(x)|_{[x_i, x_{i+1}]} \quad \leq \quad & (x - x_i)(z_i' - \Delta_i) + (x - x_i)^2/h(z_i' - \Delta_i) + \\ & (x - x_i)^2(x - x_{i+1})/h^2(z_i' - \Delta_i + z_{i+1}' - \Delta_{i+1}). \end{aligned}$$

We then conclude that

$$\|H - P\|_\infty \leq 4h \ \max_i |\Delta_i - z_i'|.$$

The result then follows from Theorem 1, together with the inequality

$$\|f - P\|_\infty \leq \|f - H\|_\infty + \|H - P\|_\infty.$$

From Theorems 1 and 2, one concludes that for both natural and shape preserving splines, when the curvature of f is large, the upper bound for the error is also large. Note that as long as $f \in C^r$, with $r \geq 2$, the upper bound of the error for the shape-preserving splines is of order $O(h^2)$, since Δ is an order of h approximation to the derivatives at the node points. For $f \in C^2$, the upper bound for the error for the

natural splines is only of order $O(h^{1.5})$. However, for $f \in C^r$, with $r \geq 4$, the upper bound for the error with these splines is already of order $O(h^7/2)$. This seems to imply that natural splines are better choices, in terms of accuracy, when the function we wish to approximate has its derivatives of order greater or equal to 4 continuous, whilst for $f \in C^2$ the shape-preserving splines offer a better order approximation. In the examples we will include [Morse], a C^∞ potential is considered, for which we should expect the natural spline to offer more accuracy.

It is also important to note that the computational complexity of interpolating functions of one variable, employing either natural or shape-preserving splines, increases with the number of interpolation points. Such a complexity can be measured by the number of multiplications performed to invert the matrices in the related systems of linear equations, see Table 1. The other operations involved in the inversion of the matrices are disregarded in the present analysis.

Table 1: Matrices arising in the interpolation.

Spline		number of patches				
		1	2	3	4	p
Natural	# matrices	1	1	1	1	1
	order	4	8	12	16	$4p$
Shape-preserving	# matrices	1	2	3	4	p
	order	4	4	4	4	4

Recall that solving a linear system of equations via Gauss elimination with back substitution, involves a $n \times n$ matrix, thus implying the need to perform $n^3/3 + n^2 - n/3$ multiplications or divisions.

Figure 1 shows that the computational complexity associated to natural spline has a cubic growth, while shape-preserving spline displays a linear growth. The latter are therefore more adequate to deal with a large number of patches. Moreover, a parallel

Figure 1: Comparison of number of operations.

computation approach is straightforward.

3 Functions of two variables

In this section the concept of natural and shape-preserving splines to $2D$ functions is extended.

3.1 Natural vs shape preserving bicubic splines

We now aim to find splines that interpolate a given function $z = z(x, y)$ at the set of points $\{(x_i, y_j), \quad i \in I, \quad j \in J\}$, where $I = \{0, \ \ldots, \ n_x\}$ and $j \in J = \{0, \ \ldots, \ n_y\}$. The natural bicubic spline is a function S such that

- $S_{ij} := S_{|_{[x_{i-1},x_i]\times[y_{j-1},y_j]}}$ is a bicubic polynomial, for each $(i, j) \in I \times J$;
- $S(x_i, y_j) = z_{ij} := z(x_i, y_j)$, for each $(i, j) \in I \times J$;
- $S \in C^2([x_0, \ x_{n_x}] \times [y_0, \ y_{n_y}])$;
- $\frac{\partial^2}{\partial x^2} S(x_0, y_j) = \frac{\partial^2}{\partial x^2} S(x_{n_x}, y_j) = 0$ for $j \in J \setminus \{0\}$;
- $\frac{\partial^2}{\partial y^2} S(x_i, y_0) = \frac{\partial^2}{\partial y^2} S(x_i, y_{n_y}) = 0$ for $i \in I \setminus \{0\}$;
- $\frac{\partial^2}{\partial x \partial y} S(x_i, y_j) = \frac{\partial^2}{\partial x \partial y} S(x_i, y_j) = 0$ for $i = 1, \ n_x$ and $j = 1, \ n_y$;

We denote

$$S_{ij}(x, y) = \sum_{r,s=0,1,2,3} a_{rs}(x - x_{i-1})^r (y - y_{i-1})^s, \tag{10}$$

where $a_{rs} = a_{rs}(i, j)$ are the local coefficients of the spline. The 16 unknowns can be determined from the resolution of a linear system of equations of order 16.

In turn, the shape preserving spline P that interpolates z is defined by

- $P_{ij} := S_{|_{[x_{i-1},x_i]\times[y_{j-1},y_j]}}$ is a bicubic polynomial, for each $(i,j) \in I \times J$;
- $P(x_i, y_j) = z_{ij} := z(x_i, y_j)$, for each $(i,j) \in I \times J$;
- $\frac{\partial P}{\partial x}(x_i, y_j) = z_{ij}^x$, for each $(i,j) \in I \times J$;
- $\frac{\partial P}{\partial y}(x_i, y_j) = z_{ij}^y$, for each $(i,j) \in I \times J$;
- $\frac{\partial^2 P}{\partial x \partial y}(x_i, y_j) = z_{ij}^{xy}$, for each $(i,j) \in I \times J$;

The quantities z_i^x, z_i^y and z_i^{xy} must be computed a priori, using harmonic means of finite differences approximations of the respective derivatives, as was done in the 1D case.

Table 2: Coefficients.

a_{33}	a_{23}	a_{13}	a_{03}	y^3
a_{32}	a_{22}	a_{12}	a_{02}	y^2
a_{31}	a_{21}	a_{11}	a_{01}	y
a_{30}	a_{20}	a_{10}	a_{00}	1
x^3	x^2	x	1	

Denoting the shape-preserving spline at the i^{th}-j^{th} patch by

$$P_{ij}(x,y) = \sum_{r,s=0,1,2,3} \overline{a}_{rs}(x - x_{i-1})^r (y - y_{i-1})^s \tag{11}$$

one has 16 coefficients $\overline{a}_{rs} = \overline{a}_{rs}(i,j)$ for each patch $[x_i,\ x_{i+1}] \times [y_j,\ y_{j+1}]$. For both the natural and shape preserving splines, the equations may be worked out in order to reduce the number of unknowns, see [anexo].

3.2 Computational analysis

To enable a comparison between the shape-preserving and natural splines in higher dimensions, we look as in the $1D$ case into the number of multiplications needed to invert the matrices related to each of these approximations.

Table 3: Number of operations.

dimension d		number of patches				
		1^d	2^d	3^d	4^d	5^d
d=1	S	36	232	716	1616	3060
	P	36	72	108	144	180
d=2	S	1616	9.1E4	1.0E6	5.7E6	2.1E7
	P	1116	6464	1.5E4	2.6E4	4.0E5
d=3	S	9.1E4	4.5E7	1.7e9	2.3e9	1.7E11
	P	9.1E4	7.3E5	2.5E6	5.9E6	1.1E7
d=4	S	5.7E6	2.3E10	3.0E12	9.4E13	1.4E15
	P	5.7E6	9.1E7	4.6E8	1.4E9	3.5E9
d=5	S	3.6E8	1.2E13	5.1E15	3.9E17	1.1E19
	P	3.6E8	1.1E10	8.7E10	3.7E11	1.1E12
d=6	S	2.3E10	6.0E15	8.8E18	1.6E21	8.7E22
	P	2.3E10	1.5E12	1.7E13	9.4E13	3.6E14

Clearly, the comparison of computational complexity for the S and P splines shows that it is quite computationally expensive to adopt the former in high dimensional cases. If the PES is a function of d dimensions and the number of patches is p, Table 3 shows that natural splines involve $(4^d p)^3/3 + (4^d p)^2 - (4^d p)/3$ multiplications, while shape-preserving splines require $p((4^d)^3/3 + (4^d)^2 - (4^d)/3)$ such operations. Moreover, the dimension of the matrices than one has to invert when utilizing shape-preserving splines is much smaller that one using natural splines, making it a far more feasible method.

As noted above in the paper, the computation of an accurate energy point from electronic structure methods can be a rather expensive process. As a result, the number of points where the value of the potential is required should be optimized, especially

when working in many dimensions. A question that arises naturally is then whether the location of the interpolation points can be chosen a priori so that more accurate approximations are obtained with an equal number of interpolation points.

From the previous analysis, we conclude that it is important to reduce the number of patches without compromising the accuracy of the approximation. Bearing this in mind, we have established the algorithm included bellow. The idea is to start from two coarse uniform meshes $X_1 = (x_1(j))_j$ and $X_2 = (x_2(j))_j$, with mesh widths respectively h and $h/2$. A refined mesh $X_3 = (x_3(j))_j$ is then generated at each iteration. The spline P_3 related to this refined mesh is then compared to the spline P_2 related to X_2. For the patches where such a difference exceeds a certain tolerance, Tol, more points are added.

Algorithm 1

Given: Tol, X_1, X_2.

1. Compute P_1, P_2 and $e := \|P_2 - P_1\|_\infty$.

2. If $e < Tol$, go to step 5. Else, go to step 3.

3. Let $X_3 = X_2$ and $x_2(j)$ the j^{th} node of the mesh X_2. For each value of j from 1 up to $|X_2| - 1$, where $|X_2|$ is the cardinal of X_2, compute $\epsilon = \|P_2 - P_1\|\|_{[x_2(j), x_2(j+1)]}$. If $\epsilon > Tol$, add the point $1/2(x_2(j) + x_2(j+1))$ to the mesh X_3.

4. Let $P_1 := P_2$, $X_1 = X_2$ and $X_2 = X_3$. Compute P_2 to fit the desired function over the mesh X_2, over the relevant domain. Compute $e := \|P_2 - P_1\|_\infty$. Go back to step 2.

5. Let $X = X_2$. Compute P to fit the potential over X_2.

4 Numerical discussion

As cases studies to test the above approximations, we will discuss the two-oscillator system described by a Morse potential and the Varandas' potential.

The Morse potential, expressed in terms of displacements from equilibrium $x,\ y \in [-0.5,\ 5.5]$, assumes the form

$$V(x,y) = V_M(x) + V_M(y) + 0.1(x^2y + xy^2)e^{[-2(x^2+y^2)]}, \tag{12}$$

with V_M being the one-dimensional Morse potential

$$V_M(x) = 18e^{-x}(2 - e^{-x}). \tag{13}$$

We will start by fitting (13) with both natural and shape-preserving splines, employing standard functions already implemented in Matlab, so that we can compare both approximations, cf.[?] These functions are adapted to handle the analysis of a shape-preserving interpolation of (12) and the Varandas' potential.

4.1 One-dimensional potential

Consider the Morse potential V_M expressed by equation (13) and represented in Figure 2 by the solid black line. The problem of interpolating this $1D$ function offers interesting challenges that are important to discuss before proceeding to interpolating functions with more dimensions.

Figure 2: Fitting for Morse potential; uniform mesh with 7 nodes.

We start by fitting V_M over the domain $I = [-0.5,\ 5.5]$. An uniform mesh with 7 nodes is employed to obtain two approximations: a natural spline S and a shape-preserving spline P. Both are represented in Figure 2. By observing the plots it is clear that S offers a better approximation than P in $I_1 = [-0.5,\ 0.5]$, whilst both splines offer

a good approximation in $I_2 = [0.5,\ 5.5]$. These results are in accordance with the results included in the Theorems **??** and 2.2 when we take the curvature of V_M into consideration. A quantitative analysis of such results is presented in Table 4, where the errors of approximations of V_M by natural and shape-preserving splines, employing a uniform mesh with mesh widths h, are included.

Table 4: L_∞? Error of the approximations of V_M by natural and shape-preserving splines, employing an uniform mesh.

Spline	h				
	1	0.5	0.25	0.125	0.0625
S	2.6393	0.4449	0.0469	0.0038	2.73e-004
P	4.0995	0.9053	0.1358	0.0267	0.0059

Table 5: Error in the approximation to the Morse potential with a shape spline, using Algoritm 1 to select points (explain table).

Tol		iteration number			
		1	2	3	4
0.1	Interval	$[-0.5,\ 1.5]$	$[-0.5,\ 1]$	$[-0.5,\ 0]$	–
	$\|P_1 - P_2\|_\infty$	$9.7E-1$	$1.5E-1$	$3.1E-2$	–
	$\|P_2 - V_M\|_\infty$	$1.4E-1$	$2.7E-2$	$1.7E-2$	–
0.01	Interval	$[-0.5,\ 5.5]$	$[-0.5,\ 1.5]$	$[-0.5,\ 0.75]$	$[0,\ 0.25]$
	$\|P_1 - P_2\|_\infty$	$9.7E-1$	$1.5E-1$	$3.1E-2$	$6.9E-2$
	$\|P_2 - V_M\|_\infty$	$1.4E-1$	$2.7E-2$	$5.9E-3$	$3.7E-3$

In general, one should expect the natural spline to offer a better accuracy at regions where the curvature is large. On the other hand, the shape-preserving splines, besides preserving certain geometrical features of the data, are much more straightforward and computationally feasible, in particular when one considers functions of more variables. We will focus on this later type of splines and try to extract the best from their structure.

The algorithm 1 can be employed to generate a non-uniform mesh for the approximation of the Morse potential V_M. We start from a uniform grid with $h = 1$ and a refined grid with $h = 0.5$. The values displayed in Table 5 were obtained by approximating the Morse potential over the points corresponding to the meshes. The intervals where, at each iteration, more points are added, are also displayed. Note that 3 iterations were sufficient to reach the tolerance $Tol = 0.1$, while 4 iterations were needed to reach the tolerance $Tol = 0.01$.

4.2 Two-dimensional potentials

Consider now the $2D$ Morse potential represented in Figure 3.

Figure 3: Representation of the Morse potential V_{xy}.

Using the shape-preserving spline P defined earlier, we fit the potential using both a uniform and a non-uniform meshes.

Table 6: Errors in the approximation to the Morse potential employing shape-preserving splines with both uniform and non-uniform meshes.

grid	norm	h				
		1	0.5	0.25	0.125	0.0625
uniform	L^2	0.9775	0.2094	0.0392	0.0084	0.0016
	∞	10.0023	3.2307	0.8325	0.2540	0.0700
Algorithm	L^2	—	—	0.0120	0.0010	1.9572e-4
	∞	—	—	0.2540	0.0183	0.0047

The numerical solution of the potential is obtained for several uniform mesh side-sizes h. Table 6 shows the error, both in the L^2 and the ∞ norms. As expected, the error decreases with h. Moreover, it is larger in the areas where the curvature is larger,

as seen from Figure 4.

To obtain an improved approximation, we have also use a non-uniform mesh. The algorithm given above has been employed to generate a 1D mesh X. The symmetry of the problem suggests that it is sufficient to fit the potential over the mesh $X \times X$. The errors of this approximation are included in Table 6. For a better visualization, see Figure 5.

Figure 4: Error (uexacto-uaproximacao)of the approximations of the Morse potential employing a uniform mesh, $h = 0.25$.

Figure 5: Error of the approximations of the Morse potential employing a non-uniform mesh with 25 points, corresponding to $h = 0.25$.

The same procedure is adopted to approximate the Varandas' potential. The errors of the approximation of the potential are displayed in Table 7. (more?)

Table 7: Error in approximations to the Morse potential.

grid	norm	h					
		1	0.5	0.25	0.125	0.0625	0.062572
uniform	L^2	0.0130	0.0063	0.0031	0.0015	7.6063e-004	3.8012e-004
	∞	0.1714	0.0904	0.0417	0.0192	0.0095	0.0046
Algorithm	L^2	—	—	0.0022	0.0011	4.0062e-004	1.8581e-004
	∞	—	—	0.0192	0.0092	0.0023	0.0012

5 Conclusion

Even though the natural splines offer, in some cases, more accuracy than the shape-preserving splines, the latter turn out to be more adequate when one considers functions of several variables. The algorithm presented allows the computation of a non-uniform mesh aiming at a higher accuracy with less interpolation points. In a future report, we hope to generalize the algorithm to many dimensions and relate the tolerance of the algorithm to the error of fitting approximation.

Acknowledgments

This work is supported by Fundação para a Ciência e a Tecnologia, Portugal.

Figure 6: A plot ....

Figure 1, Varandas

6 Anexos

6.1 Natural spline

A natural cubic interpolating spline for the function $z = z(x)$ at the points $\{x_0,\ x_1,\ \ldots,\ x_n\}$ is a function S such that

- $S_i := S_{|_{[x_{i-1},x_i]}}$ is a cubic polynomial, $s_i(x) = a_i(x-x_{i-1})^3 + b_i(x-x_{i-1})^2 + c_i(x-x_{i-1}) + d_i$;
- $S(x_i) = z_i := z(x_i)$, for each $i = 0,\ \ldots,\ n$;
- $S \in C^2[x_0, x_n]$;
- $S''(x_0) = S''(x_n) = 0$.

In particular, when $n = 2$, S must satisfy

$S_1(x_0) = z_0 \Rightarrow d_1 = z_0$

$S_1(x_1) = z_1$ (equação full)

$S_2(x_1) = z_1 \Rightarrow d_2 = z_1$

$S_2(x_2) = z_2$ (equação full)

$S_1''(x_0) = 0 \Rightarrow b_1 = 0$

$S_2''(x_2) = 0 \Rightarrow a_2 = b_2/(x_2 - x_1)$

$S_1'(x_1) = S_2'(x_1) \Rightarrow c_2 = \ldots$

$S_1''(x_1) = S_2''(x_1) \Rightarrow b_2 = \ldots$

This means that only two unknows remain to be computed. In a more general case, when $n = p$, S must satisfy the following equations

$S_i(x_{i-1}) = z_{i-1} \Rightarrow d_i = z_{i-1}, \quad$ for $i = 1,\ \ldots, p$

$S_i(x_i) = z_i, \quad$ for $i = 1,\ \ldots, p$ (equação full)

$S_1''(x_0) = 0 \Rightarrow b_1 = 0$

$S_d''(x_2) = 0 \Rightarrow a_p = b_p/(x_p - x_{p-1})$

$S_i'(x_i) = S_{i+1}'(x_i) \Rightarrow c_{i+1} = \ldots, \quad$ for $i = 1, \ \ldots, p-1$

$S_i''(x_i) = S_{i+1}''(x_i) \Rightarrow b_{i+1} = \ldots, \quad$ for $i = 1, \ \ldots, p-1$

This means that only p unknows remain to be computed, and so a $p \times p$ matrix has to be inverted.

6.2 Shape-preserving spline

A shape-preserving cubic interpolating spline for the function $z = z(x)$ at the points $\{x_0, \ x_1, \ \ldots, \ x_n\}$ is a function S such that

- $S_i := S_{|_{[x_{i-1}, x_i]}}$ is a cubic polynomial, $s_i(x) = a_i(x - x_{i-1})^3 + b_i(x - x_{i-1})^2 + c_i(x - x_{i-1}) + d_i$;
- $S(x_i) = z_i := z(x_i)$, for each $i = 0, \ \ldots, \ n$;
- $S'(x_i) = z_i' := z'(x_i)$, for each $i = 0, \ \ldots, \ n$;

Note that for each $i = 0, \ \ldots, \ n$, the quantity $z'(x_i)$ must be computed a priori.

Now, when $n = 2$, S must satisfy

$S_1(x_0) = z_0 \Rightarrow d_1 = z_0$

$S_1(x_1) = z_1$ (equação full in a_1, b_1, c_1, d_1)

$S_2(x_1) = z_1 \Rightarrow d_2 = z_1$

$S_2(x_2) = z_2$ (equação full in a_2, b_2, c_2, d_2)

$S_1'(x_0) = z_0' \Rightarrow c_1 = z_0'$

$S_1'(x_1) = z_1'$ (equação full in a_1, b_1, c_1, d_1)

$S_2'(x_1) = z_1' \Rightarrow c_2 = z_1'$

$S_2'(x_2) = z_2'$ (equação full in a_2, b_2, c_2, d_2)

This means that only four unknows remain to be computed. Now, since it can be seen that a_1, b_1, c_1, d_1 can be computed independently from a_2, b_2, c_2, d_2, that means we have to invert two 2×2 matrices. In a more general case, when $n = p$, S must satisfy the following equations, for $i = 1, \ldots, p$

$S_i(x_{i-1}) = z_{i-1} \Rightarrow d_i = z_{i-1}$

$S_i(x_i) = z_i$ (equação full in a_i, b_i, c_i, d_i)

$S_i'(x_{i-1}) = z_{i-1}' \Rightarrow c_i = z_{i-1}'$

$S_i'(x_i) = z_i'$ (equação full in a_i, b_i, c_i, d_i)

Each of these p systems of four equations each can be solved independently of the others. This means that for each patch a 2×2 matrix has to be inverted. In total, p matrices, each 2×2, have to be inverted.

In other words,

- From the equations in x_{i-1}, one computes c_i and d_i;
- From the equations in x_i, one obtains two coupled equations for a_i and b_i;

6.3 2D shape-preserving spline

A shape-preserving bicubic interpolating spline for the function $z = z(x, y)$ at the points (x_i, y_j), $i = 0, \ldots, n_x$, $j = 0, \ldots, n_y$, is a function S such that

- $S_{ij} := S_{|_{[x_{i-1},x_i]\times[y_{j-1},y_j]}}$ is a bicubic polynomial, $S_{ij}(x, y) = \sum_{r,s=0,1,2,3} a_{rs}(x - x_{i-1})^r(y - y_{i-1})^s$;
- $S(x_i, y_j) = z_{ij} := z(x_i, y_j)$, for $i = 0, \ldots, n_x$, $j = 0, \ldots, n_y$;
- $\frac{\partial S}{\partial x}(x_i, y_j) = z_i^x := \frac{\partial z}{\partial x}(x_i, y_j)$, , for $i = 0, \ldots, n_x$, $j = 0, \ldots, n_y$;

- $\frac{\partial S}{\partial y}(x_i, y_j) = z_i^y := \frac{\partial z}{\partial y}(x_i, y_j)$, , for $i = 0, \ \ldots, \ n_x$, $j = 0, \ \ldots, \ n_y$;
- $\frac{\partial^2 S}{\partial x \partial y}(x_i, y_j) = z_i^{xy} := \frac{\partial^2 z}{\partial x \partial y}(x_i, y_j)$, , for $i = 0, \ \ldots, \ n_x$, $j = 0, \ \ldots, \ n_y$;

Note that for $i = 0, \ \ldots, \ n_x$, $j = 0, \ \ldots, \ n_y$, the quantities z_i^x, z_i^y and z_i^{xy} must be computed a priori.

Now, when $n = 1$, one has the coefficients

Table 8: Coefficients.

a_{33}	a_{23}	a_{13}	a_{03}	y^3
a_{32}	a_{22}	a_{12}	a_{02}	y^2
a_{31}	a_{21}	a_{11}	a_{01}	y
a_{30}	a_{20}	a_{10}	a_{00}	1
x^3	x^2	x	1	

We thus obtain 16 equations.

- From the equations in (x_0, y_0), one computes a_{00}, a_{10}, a_{01}, a_{11};
- From the equations in (x_1, y_0), one obtains two coupled equations for a_{00}, a_{10}, a_{20}, a_{30} and two coupled equations for a_{01}, a_{11}, a_{21}, a_{31};
- From the equations in (x_0, y_1), one obtains two coupled equations for a_{00}, a_{01}, a_{02}, a_{03} and two coupled equations for a_{10}, a_{11}, a_{12}, a_{13};
- From the equations in (x_1, y_1), one obtains four coupled equations for a_{22}, a_{23}, a_{32}, a_{33}.

This means that when $n = d$, for each of the d patches one has to invert a 4×4 matrix and four 2×2 matrices.