J Math Chem (2013) 51:1729–1746
DOI 10.1007/s10910-013-0173-4

ORIGINAL PAPER

Roadmap to spline-fitting potentials in high dimensions

M. Patrício · J. L. Santos · F. Patrício · A. J. C. Varandas

Received: 21 January 2013 / Accepted: 22 March 2013 / Published online: 3 April 2013

Abstract The 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. In this regard, the problem of dimensionality is pinpointed, with shape-preserving splines emerging as a viable alternative for fitting surfaces in multidimensional spaces. An algorithm is also presented to allow the use of non-uniform meshes with high accuracy fitting and less interpolation points.

Keywords Splines · Shape-preserving splines · Potential energy surfaces

1 Introduction

A challenging step in reaction dynamics is to model the potential energy surface (PES; this stands generally both for a curve or hypersurface) that governs the motion of the nuclei for any arrangement of the latter [1–8]. Although important progress towards dynamics on-the-fly (the electronic Schrödinger equation is solved simultaneously to the corresponding equations 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 a few atoms). In this approach, the two main streams to the

M. Patrício · J. L. Santos · F. Patrício
CMUC, Departamento de Matemática, Universidade de Coimbra, 3004-535 Coimbra, Portugal

A. J. C. Varandas (✉)
Departamento de Química, Universidade de Coimbra, 3004-535 Coimbra, Portugal
e-mail: varandas@qtvs1.qui.uc.pt

PES consist of modeling it with a suitable functional form (function or generically f), often and hopefully originated from physically motivated arguments [5,7–11], and of using semi-numerical black-box fitting techniques based on splines or related interpolation approaches [5,12–14]. Although these mathematical tools offer unbiased and simple approaches to use, they face the serious problem of dimensionality, so-called [1] 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). In this work, we examine the capabilities of two types of splines when high-dimensionality problems are at stake.

Central to on-the-fly (or direct) dynamics approaches is the necessity of evaluating the forces acting on the nuclei at each time step. Although for particular geometries some quantum chemistry packages can provide accurate electronic energies, gradients, and even higher derivatives at moderate cost, this can be prohibitive for approaches in which an *ab initio* calculation is performed at each of the millions or even billions of geometries that may arise during the calculation. Efficient interpolation schemes then appear as a natural way out. These fitting methods include splines [4,12–15], modified-Shepard interpolation [16,17], gradient-based multiconfiguration Shepard interpolation [18], interpolating moving least-squares [19,20], and reproducing kernel Hilbert space [21,22] approaches, just to mention a few; a vast number of other references to literature on the above and related methods can be found by cross referencing.

The first to employ cubic splines in fitting a PES were McLaughlin and Thompson [12] in their study of the $HeH^+ + H_2 \rightarrow He + H_3^+$ reaction dynamics. For simplicity reasons, these authors have restricted the space by keeping reaction paths with C_{2v} symmetry, and were led to conclude that multidimensional spline fitting of *ab initio points* can serve as a useful, objective interpolator. Although the method removes the necessity of choosing (often somewhat arbitrarily) an analytic interpolation function and can ensure the continuity of the energy and the first derivatives over the entire hypersurface, several unanswered questions persisted, in particular concerning its use for large dimensional configuration spaces. This motivated Sathyamurthy and Raff [13] to develop and apply a 3D cubic spline fitting routine. Despite the unique features of splines, to our knowledge, no development towards higher dimensionalities have yet been reported.

Specifically, we will examine here the problem of obtaining a global fit of a PES for a small molecule. For that purpose, traditional cubic splines (natural cubic splines) will be compared to shape-preserving splines and both tools will be extended for higher dimensions. The theory and properties of both spline-types will first be examined for the $1D$ case. Shape-preserving splines are known to yield visually pleasing plots, as properties inherent to the discrete data such as monotonicity are maintained in the approximation, meaning that these splines are monotonic when the data is monotonic and present local extrema at the data points that are local extrema [23–25]. Conversely, it is shown that under certain conditions the natural splines offer a greater accuracy than the shape-preserving splines at regions of strong curvature when $f \in C^k, k \geq 2$. A comparison between the number of operations involved in fitting a PES in higher dimensions shows that the usage of natural cubic splines is restricted to a small number

of patches and low dimensions. This paper offers an alternative with dramatically smaller computational costs.

As noted above, for medium-size molecules (or even relatively modest ones with 4 or more atoms), the modeling of a global analytic potential function can be a mammoth task, with simplicity calling for strictly numerical techniques to face the problem. The discretisation of the function requires the employment of grids which should be refined until a predefined accuracy is obtained. This implies in turn a great computational effort if high-accuracy is on demand, particularly at regions of large curvature. The use of non-uniform meshes will then be examined too here, aiming at cost-effectiveness.

The paper is organized as follows. Section 2 is devoted to the definition of both shape preserving and natural splines. The accuracy offered by the approximations is examined, and related computational complexities discussed. In Sect. 3, the independency on neighbouring patches offered by shape preserving splines is generalized to the $2D$ case. An algorithm to fit potentials in a more cost-effective manner is then proposed. Test cases and computational examples will also be reported in Sect. 4, namely for the popular Morse curve ($1D$) and a more complicated $2D$ PES taken from the literature. Section 5 gathers the conclusions.

2 Functions of one variable

Consider a set of points $\{x_i, i \in I\}$, 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 versus shape-preserving splines

The classical natural cubic interpolating spline for the function $z = z(x)$ at the points $\{x_i, i \in I\}$ is a function S that satisfies the conditions [14]

- $S_i := S_{|[x_{i-1}, x_i]}$ is a cubic polynomial, for $i \in I \setminus \{0\}$;
- $S(x_i) = z_i := z(x_i)$, 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

$$S_i(x_{i-1}) = z_{i-1}; \quad S_i(x_i) = z_i, \quad \text{for } i \in I \setminus \{0\}; \tag{2}$$

$$S_i'(x_i) = S_{i+1}'(x_i), \quad S_i''(x_i) = S_{i+1}''(x_i) \text{ for } \in I \setminus \{0, n\}; \tag{3}$$

$$S_1''(x_0) = S_n''(x_n) = 0. \tag{4}$$

The determination of these coefficients may be done by solving a $4n \times 4n$ system of linear equations. It is possible to work out the equations further to reduce the order

of the matrix of the related linear system, as it is shown on the "Appendix". However, the size of the matrix will always depend explicitly on the number of patches taken.

An alternative to natural cubic splines, still to interpolate z, are shape-preserving splines P [24,26]. These are defined by

- $P_i := P_{|[x_{i-1},x_i]}$ is a cubic polynomial, for $i \in I \setminus \{0\}$;
- $P(x_i) = z_i := z(x_i)$, for each $i \in I$;
- $P'(x_i) = \Delta_i$, for each $i \in I$.

For each value of $i \in I \setminus \{0, n\}$, the quantity Δ_i is an approximation to the derivative of z at x_i, 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

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

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 easily recognised as 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 employed for the points x_i, with $i \in I \setminus \{0, n\}$. A similar approximation can be employed for the endpoints, cf. [23,24]. The shape preserving spline P may then be interpreted as an Hermite interpolating spline fitting to a function with values z_i and derivatives Δ_i at the points x_i. We denote the spline at the interval $[x_{i-1}, x_i]$ 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 be determined from the linear system related to the equations

$$P_i(x_{i-1}) = z_{i-1}, \quad P_i'(x_{i-1}) = \Delta_{i-1}, \tag{8}$$

$$P_i(x_i) = z_i, \quad P_i'(x_i) = \Delta_i, \tag{9}$$

for $i \in I \setminus \{0\}$. Note that for each value of i the coefficients of the polynomial can be computed independently by solving a 4×4 linear system. Finding all coefficients would require solving n such linear systems. However, it is easy to see that the unknowns $\overline{c}_i$ and $\overline{d}_i$ may be computed from the Eqs. (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 Eq. (9). 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 "Appendix". 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.

2.2 Error analysis and computational complexity

It is desirable to find estimates for the errors arising in the approximation 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 Ref. [14]. The results refer to cubic polynomials that differ from the natural splines only at the end-points. There, the derivatives of the polynomials are required to equal the derivatives of f, instead of the polynomials having the second derivative equal to zero. It can be verified that the bounds set in Ref. [14] are still valid for natural splines. Let S then be the natural interpolating spline for f. The following results hold [14]:

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 also important to find upper bounds for the error of the approximation of f with the latter. 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*

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

Proof Let H be the Hermite cubic spline that satisfies

$$H(x_i) = z_i; \quad H'(x_i) = z'_i, \quad i \in I.$$

Then one has

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

Moreover, it may be shown that

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

and therefore

$$\|H - P\|_\infty \leq 4h \quad \max_i |\Delta_i - z'_i|.$$

The result then follows from the inequality

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

From the results given above, 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 approximation order of h 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 $O(h^{3/2})$. 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.

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 inferred from the number and dimension of matrices one has to invert in order to solve related systems of linear equations, see Table 1.

Inverting a $n \times n$ matrix requires performing an order of n^3 multiplications. For that reason, the computational complexity associated to natural splines has a cubic growth as a function of the number of interpolation points, while shape-preserving splines display a linear growth. The latter are therefore more adequate to deal with a large number of patches. Moreover, a parallel computation approach is straightforward.

3 Functions of two variables

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

Table 1 Attributes of the matrices (number of matrices and order) arising in the interpolation

Spline type	Attribute	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	2	2	2	2	2

3.1 Natural versus shape preserving bicubic splines

We now aim to find splines that interpolate a given function $z = z(x, y)$ at a set of points $\{(x_i, y_j), i \in I, j \in J\}$, where $I = \{0, \ldots, n_x\}$ and $J = \{0, \ldots, n_y\}$. The natural bicubic spline is a function S such that [27]

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

We denote

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

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.

To preserve the independency on neighbouring patches, one may generalise the 1D shape preserving spline definition thus obtaining a spline P that interpolates z, which will still be referred to as shape-preserving spline for the sake of simplicity, which satisfies

- $P_{ij} := S_{|[x_{i-1},x_i]\times[y_{j-1},y_j]}$ is a bicubic polynomial, for each $(i, j) \in I \setminus \{0\} \wedge J \setminus \{0\}$;
- $P(x_i, y_j) = z_{ij} := z(x_i, y_j)$, for each $(i, j) \in I \times J$;
- $\frac{\partial P}{\partial x}(x_i, y_j) = \Delta^x_{ij}$, for each $(i, j) \in I \times J$;
- $\frac{\partial P}{\partial y}(x_i, y_j) = \Delta^y_{ij}$, for each $(i, j) \in I \times J$;
- $\frac{\partial^2 P}{\partial x \partial y}(x_i, y_j) = \Delta^{xy}_{ij}$, for each $(i, j) \in I \times J$;

The quantities Δ^x_{ij}, Δ^y_{ij} and Δ^{xy}_{ij} must be computed a priori, using harmonic means of finite differences approximations of the respective derivatives, as done in 1D case.

Denoting the shape-preserving spline at the ith–jth patch by

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

one has 16 coefficients $\overline{a}_{rs} = \overline{a}_{rs}(i, j)$ for each patch $[x_{i-1}, x_i] \times [y_{j-1}, y_j]$.

Table 2 Number of relevant operations

Dimension (d)	Spline type	Number of patches				
		1^d	2^d	3^d	4^d	5^d
1	*S*	36	232	716	1,616	3,060
	P	36	72	108	144	180
2	*S*	1,616	9.1E4	1.0E6	5.7E6	2.1E7
	P	1,116	6,464	1.5E4	2.6E4	4.0E5
3	*S*	9.1E4	4.5E7	1.7E9	2.3E9	1.7E11
	P	9.1E4	7.3E5	2.5E6	5.9E6	1.1E7
4	*S*	5.7E6	2.3E10	3.0E12	9.4E13	1.4E15
	P	5.7E6	9.1E7	4.6E8	1.4E9	3.5E9
5	*S*	3.6E8	1.2E13	5.1E15	3.9E17	1.1E19
	P	3.6E8	1.1E10	8.7E10	3.7E11	1.1E12
6	*S*	2.3E10	6.0E15	8.8E18	1.6E21	8.7E22
	P	2.3E10	1.5E12	1.7E13	9.4E13	3.6E14

3.2 Computational analysis

To enable a comparison between the shape-preserving and natural splines in higher dimensions, we look at the number of multiplications needed to invert the matrices related to each of these approximations. If the PES is a function of d dimensions and the number of patches is p, without working out the equations the use of natural splines involves an order of $(4^d p)^3$ multiplications, while shape-preserving ones require an order of $p(4^d)^3$ such operations. This is illustrated in Table 2, where Gauss elimination with back substitution is adopted for matrix inversion. This implies performing $n^3/3 + n^2 - n/3$ operations to invert a matrix of order n. Clearly, both methods are computationally quite expensive. However, the dimension of the matrices that one has to invert when utilising shape-preserving splines is much smaller than using natural splines, making it far more feasible for smaller values of d. As we have presented the problem thus far, as the number of dimensions increases, so does the size of the matrices one has to invert, crippling the application of splines to high values of d. However, it is possible to solve the problem for each dimension independently, as has been noted elsewhere [28], for example. We include the details on how to reduce a problem in d dimensions to problems in $d - 1$ dimensions in the "Appendix". In practice, this means that using shape-preserving splines to approximate a function of d dimensions, 8^{d-1} matrices of order 2 have to be inverted at each patch: if the number of patches is p, this translates into $p8^{d-1}$ matrices. This number grows dramatically as d increases, which is reflected on the number of relevant operations presented in Table 3, but the order of the matrices remains equal to 2 instead of growing to accommodate the number of patches and the dimensionality of the problem. This means that approximating a potential in d dimensions is simply a matter of having sufficient computers and computation time.

Table 3 Number of relevant operations when solving separately for each dimension

Dimension (d)	Spline type	Number of patches				
		1^d	2^d	3^d	4^d	5^d
1	P	6	12	18	24	30
2	P	48	192	432	768	1,200
3	P	384	3,072	10,368	24,576	48,000
4	P	3,072	49,152	2.5E5	7.9E5	1.9E6
5	P	24,576	7.9E5	6.0E6	2.5E7	7.7E7
6	P	2.0E5	1.3E7	1.4E8	8.1E8	3.1E9
9	P	1.0E8	5.2E10	2.0E12	2.6E13	2.0E14

As noted above in the paper, the computation of an accurate energy point from electronic structure methods can be a rather expensive process [29]. As a result, the number of points where the value of the potential is required should be minimized, specially when working in higher dimensions. It is also important to reduce the number of patches without compromising the accuracy of the approximation. 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. Bearing this in mind, we have established the algorithm included below. 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 h and $h/2$, respectively, and compute the related splines P_1 and P_2 that interpolate the potential on such points. A refined mesh $X_3 = (x_3(j))_j$ is then generated at each iteration by adding points to X_2 at the locations where $P_1 - P_2$ is greater than or equal to a given tolerance. The spline P_3 related to this refined mesh X_3 is then compared to P_2. The iterative process is continued by refining X_3 further.

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 jth 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 case studies, we will discuss the 1D Morse potential, a 2D modified Morse potential, and a more realistic 3D PES (referred to as DMBE IV [30]) once frozen one degree of freedom (in this case, the included ∠HOO that has been fixed at the equilibrium value). Although of no great relevance in this work, dimensions for the stretching

coordinates are in bohr ($a_0 = 0.5292 \times 10^{-10}$ m), with the energy coming in hartree ($1E_h = 4.3597$ aJ). They will be omitted heretofore for convenience.

The modified Morse potential in 2D, expressed in terms of displacements from equilibrium $x, y \in [-0.5, 5.5]$, assumes the form [31]

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

with V_M being the one-dimensional Morse potential

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

4.1 One-dimensional potential

We start by fitting Eq. (13) with both natural and shape-preserving splines, employing standard functions already implemented in Matlab, so that we can compare both approximations, cf. Ref. [24].

Consider the Morse potential V_M expressed by Eq. (13) and represented in Fig. 1 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. We start by fitting V_M over the domain $I = [-0.5, 5.5]$. A 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 Fig. 1. By observing the plots it is clear that S is a better approximation than P in $I_1 = [-0.5, 0.5]$, whilst both splines offer a good approximation in $I_2 = [0.5, 5.5]$. The errors of approximations of V_M by natural and shape-preserving splines, employing a uniform mesh with mesh width h, are included in Table 4. These results are in accordance with the analysis included in Sect. 2. Indeed, since $V_M \in C^\infty$, we expect S to be more accurate than P at regions where the curvature is large.

We have seen that shape-preserving splines, besides preserving certain geometrical features of the data, are much more straightforward and computationally feasible. However, as h decreases and the number of patches increases, the computational cost will still increase greatly. It is then important to illustrate how Algorithm 1 can be employed to generate a non-uniform mesh for the approximation of the Morse potential V_M. We start by generating two grids, X_1 and X_2, respectively a uniform grid with mesh width $h = 1$ and a refined uniform grid with $h = 0.5$. The corresponding shape preserving splines P_1 and P_2 are computed, as well as the norms of $P_1 - P_2$ and $P_2 - V_M$. The latter grid is then refined at each iteration by successive bisections until $\|P_1 - P_2\|_\infty < Tol$, see Table 5. Note that 3 iterations are sufficient to reach the tolerance $Tol = 0.1$, with only one extra iteration being required to reach the tolerance $Tol = 0.01$ and two more for the tolerance $Tol = 0.001$.

4.2 Two-dimensional potentials

Let us now consider the symmetric $2D$ modified-Morse potential represented in Fig. 2. Using the shape-preserving spline P defined earlier, we fit the potential using both

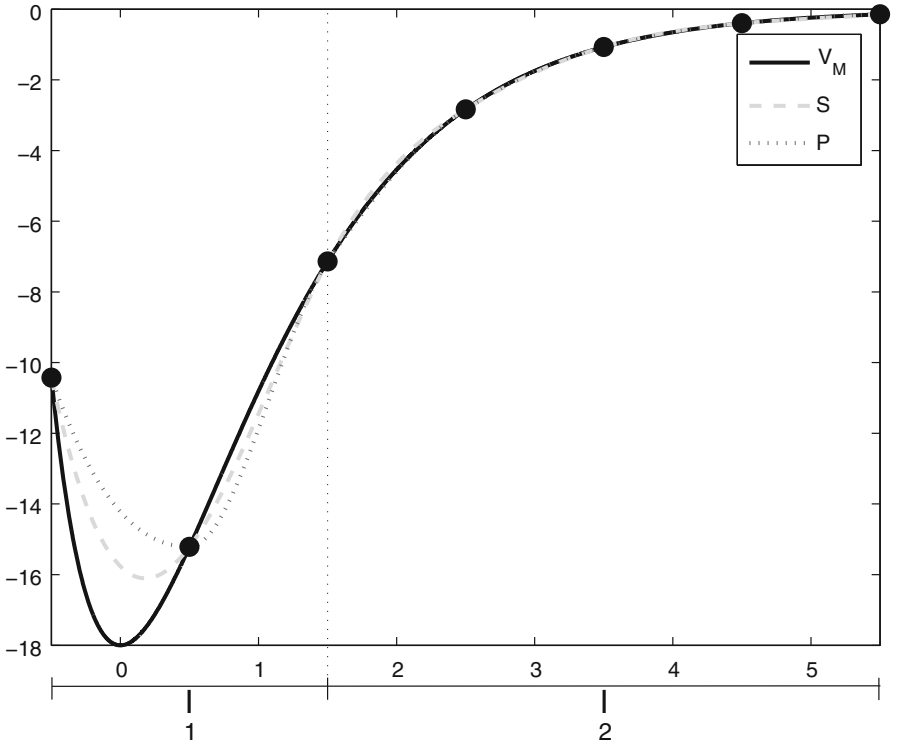

Fig. 1 Fitting for $1D$ Morse potential; uniform mesh with 7 nodes

Table 4 L_∞ norm of the error of the approximations of V_M

Spline type	h				
	1	0.5	0.25	0.125	0.0625
S	2.6393	0.4449	0.0469	0.0038	0.0003
P	4.0995	0.9053	0.1358	0.0267	0.0059

uniform and non-uniform meshes. Several uniform mesh side-sizes h have been employed. Table 6 shows the errors of the approximations, 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 can be seen in Fig. 3.

Non-uniform meshes are also used to obtain improved approximations for each value of h, allowing for a better distribution of the errors in the approximation, see Fig. 4. These meshes have exactly the same number of points as the corresponding uniform meshes and are of the form $X \times X$, where X is a non-uniform mesh generated by Algorithm 1. The symmetry of the problem suggests that it is sufficient to consider meshes of this form. The errors of these approximations are included in Table 6. The same procedure is adapted to approximate the DMBE IV potential. The errors

Table 5 Error in the approximation to the Morse potential with a shape-preserving spline, using Algorithm 1 to select points

Tol	Iteration number					
	1	2	3	4	5	6
0.1						
$\|P_1 - P_2\|_\infty$	9.7E−1	1.5E−1	3.1E−2	–	–	–
$\|P_1 - P_2\|_{L^2}$	1.7E−1	3.4E−3	1.3E−4	–	–	–
$\|P_2 - V_M\|_\infty$	1.4E−1	4.8E−2	4.8E−2	–	–	–
$\|P_2 - V_M\|_{L^2}$	3.9E−3	4.2E−4	4.0E−4	–	–	–
0.01						
$\|P_1 - P_2\|_\infty$	9.7E−1	1.5E−1	3.1E−2	8.9E−3	–	–
$\|P_1 - P_2\|_{L^2}$	1.7E−1	3.8E−3	1.1E−4	5.7E−6	–	–
$\|P_2 - V_M\|_\infty$	1.4E−1	2.7E−2	5.9E−3	9.0E−3	–	–
$\|P_2 - V_M\|_{L^2}$	3.9E−3	1.1E−4	7.8E−6	7.8E−6	–	–
0.001						
$\|P_1 - P_2\|_\infty$	9.7E−1	1.5E−1	3.1E−2	6.9E−3	1.6E−3	4.0E−4
$\|P_1 - P_2\|_{L^2}$	1.7E−1	3.8E−3	1.1E−4	3.2E−6	9.3E−8	2.8E−9
$\|P_2 - V_M\|_\infty$	1.4E−1	2.7E−2	5.9E−3	1.4E−3	8.5E−4	8.5E−4
$\|P_2 - V_M\|_{L^2}$	3.9E−3	1.1E−4	3.2E−6	1.9E−7	1.1E−7	1.1E−7

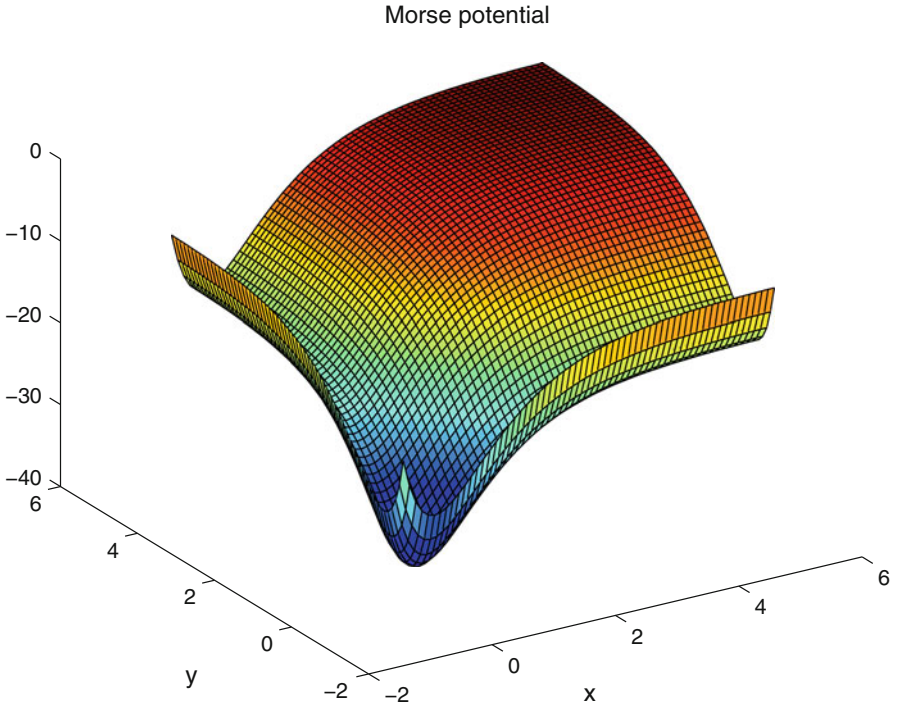

Fig. 2 (Color online) Representation of $2D$ symmetric Morse-type potential

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	1/2	1/4	1/8	1/16	1/32	1/64
Uniform	L^2	1.0	2.2E−1	4.1E−2	8.8E−3	1.7E−3	3.1E−4	5.6E−5
	∞	10.0	3.3	9.0E−1	2.7E−1	7.3E−2	1.9E−2	4.8E−3
Algorithm	L^2	–	–	8.2E−3	1.4E−3	1.7E−4	3.9E−5	7.1E−6
	∞	–	–	1.1E−1	2.7E−2	4.8E−3	1.2E−3	3.5E−4

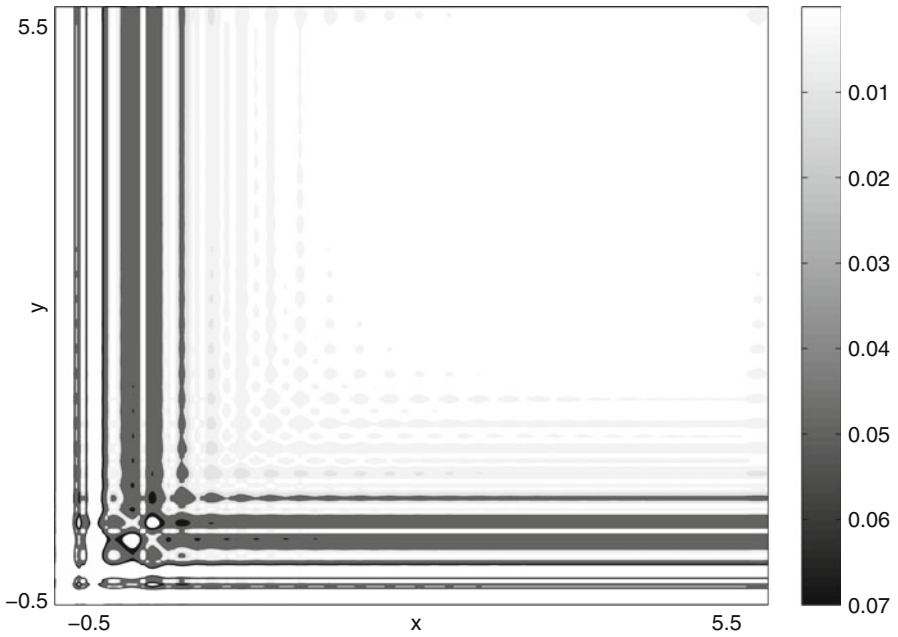

Fig. 3 Errors in approximation of $2D$ symmetrical Morse-type potential using a uniform mesh, $h = 0.25$. *Contours*, in E_h, are at 0.00001, 0.00005 0.0001 0.0005 0.001 0.005 0.01 0.05, and 0.07. *Darker regions* indicate larger errors

of the approximation of the potential are displayed in Table 7. Clearly, the errors decrease greatly and in an approximately linear way with the mesh width, though a greater number of mesh points implies that a greater computational effort must be made. However, even though this does affect the number of operations that must be performed, hence also the computing time, it does not hinder the feasibility of the interpolation with shape-preserving splines. Indeed, the dimensions of the matrices that must be inverted to compute the coefficients of these splines does not depend on the number of patches. Generalising the approach to higher dimensions is straightforward. If a small tolerance is set, a greater number of patches will be required, which implies time-consuming computations. However, parallelisation is straightforward.

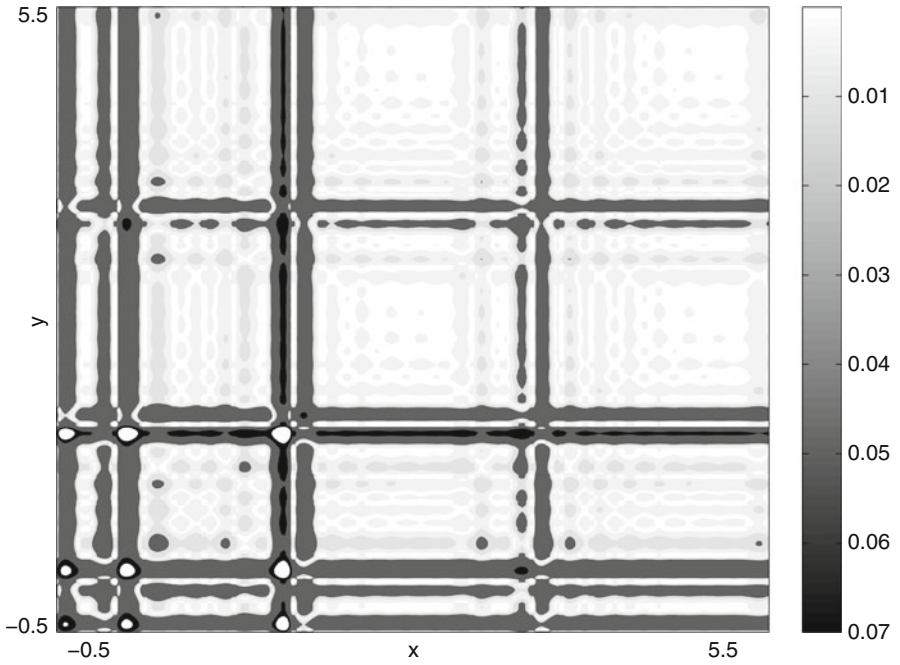

Fig. 4 Errors in approximation of $2D$ symmetrical Morse potential using a non-uniform mesh with 25 points, corresponding to $h = 0.25$. *Contours* as in Fig. 3, with *darker regions* indicating larger errors

Table 7 Error in approximations to the DMBE IV potential

Grid	Norm	h						
		1	1/2	1/4	1/8	1/16	1/32	1/64
Uniform	L^2	1.2E−2	5.5E−3	2.7E−3	1.3E−3	6.6E−4	3.3E−4	1.1E−4
	∞	1.8E−1	1.0E−1	5.0E−2	2.3E−2	1.2E−2	5.8E−3	1.3E−3
Algorithm	L^2	–	–	1.7E−3	7.7E−4	4.0E−4	2.1E−4	1.7E−4
	∞	–	–	1.6E−2	8.0E−3	3.7E−3	2.3E−3	2.9E−3

5 Conclusions

Even though for $1D$ problems natural splines offer, for similar grids of points and in some cases, more accuracy than shape-preserving splines, the former turn out to be inadequate when considering functions of several variables. The dimension of the matrices that must be dealt with when fitting with natural splines vary on both the dimension of the interpolated function and the number of patches, while the order of matrices related to shape-preserving splines is invariant. This aspect renders the usage of natural splines in higher dimensions unfeasible. An algorithm was presented in this paper to allow the computation of non-uniform meshes that make it possible to obtain higher accuracy fitting with less interpolation points. This has no effect on the feasi-

bility of the interpolation, but rather on how much time all the necessary computations will take. 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 financed by FEDER through "Programa Operacional Factores de Competitividade—COMPETE" and national funds under the auspices of Fundação para a Ciência e a Tecnologia, Portugal (projects PTDC/QUI-QUI/099744/2008, and PTDC/ AAC-AMB/099737/2008).

6 Appendix

6.1 Functions of one variable

The conditions that must be satisfied by the natural cubic spline S for the function $z = z(x)$ at the points $\{x_i, i \in I\}$ have been reviewed in Sect. 2. Since it must satisfy, for all $i \in I \setminus \{0\}$, $S_i(x_{i-1}) = z_{i-1}$, n unknowns are trivially determined, $d_i = z_{i-1}$. Also, the continuity requirements for the first and second derivatives at the points x_i, for $i \in I \setminus \{0, n\}$, imply that

$$c_{i+1} = 3a_i h_i^2 + 2b_i h_i + c_i; \quad b_{i+1} = 3a_i h_i + b_i.$$

Together with the conditions at the endpoints x_0 and x_n given by

$$b_1 = 0; \quad b_n = -3a_n h_n,$$

this allows the elimination of all the unknowns except $a_0, a_1, \ldots, a_{n-1}$ and c_1 from the remaining conditions expressed by

$$S_i(x_i) := z_i, \quad \text{for } i \in I \setminus \{0\}.$$

It is then sufficient to invert a $n \times n$ matrix to compute the coefficients of the spline.

A similar analysis may be carried out for the shape-preserving cubic spline P for the function $z = z(x)$, with the difference that the coefficients can be computed for each patch, independently of the other patches. From the conditions

$$P_i(x_{i-1}) = z_{i-1}; \quad P_i'(x_{i-1}) = \Delta_{i-1}, \text{ for } i \in I \setminus \{0\}$$

one concludes that $\overline{d}_i = z_{i-1}$ and $\overline{c}_i = \Delta_{i-1}$. Then, for each patch, only the conditions

$$P_i(x_i) = z_i; \quad P_i'(x_i) = \Delta_i, \text{ for } i \in I \setminus \{0\}$$

need to be resolved in order to find the remaining unknowns $\overline{a}_i$ and $\overline{b}_i$. This translates into a total of n matrices, each of order 2×2, that have to be inverted.

6.2 Dealing with higher dimensions

We start by looking at how to determine the coefficients of a shape-preserving bicubic spline $P = P(x, y)$ for the function $z = z(x, y)$ at the points $\{(x_i, y_j), i \in I, j \in J\}$, where $I = \{0, \ldots, n_x\}$ and $J = \{0, \ldots, n_y\}$. These can be determined independently for each patch. The purpose is to show that the coefficients of this $2D$ spline may be found by solving problems for $1D$ cubic splines, in which the order of the matrices involved (after computing the variables which may be trivially determined) is 2. The generalisation to higher dimensions is straightforward implying that the n-dimensional problem is reducible to a $1D$ problem, as we will see. The spline P is denoted by

$$P_{ij}(x, y) = \sum_{r=0,1,2,3} \sum_{s=0,1,2,3} \overline{a}_{rs}(x - x_{i-1})^r (y - y_{j-1})^s, \tag{14}$$

which may be rewritten as

$$P_{ij}(x, y) = \sum_{r=0,1,2,3} b_r(y)[h(x)]^r, \tag{15}$$

where for $r = 0, 1, 2, 3$ one has

$$b_r(y) = \sum_{s=0,1,2,3} \overline{a}_{rs}[k(y)]^s, \tag{16}$$

and the notation $h(x) = (x - x_{i-1})$ and $k(y) = (y - y_{j-1})$ was employed. The dependence of the coefficients b_r on both i and j was not made explicit on the notation for the sake of simplicity, but indeed they vary from patch to patch. Now, for $y = y_l$, where $l = j - 1, j$, (15) reads

$$P_{ij}(x, y_l) = \sum_{r=0,1,2,3} b_r(y_l)[h(x)]^r. \tag{17}$$

To determine the constants $b_r(y_l)$ it suffices to consider the interpolation conditions for z and its derivative in respect to x at the corners of the patch $[x_{i-1}, x_i] \times [y_{j-1}, y_j]$, giving rise to the equation

$$\mathbf{B}_l = \mathbf{H}^{-1}\mathbf{C}_l, \tag{18}$$

where the matrices above read

$$\mathbf{B}_l = \begin{bmatrix} b_0(y_l) \\ b_1(y_l) \\ b_2(y_l) \\ b_3(y_l) \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & h(x_i) & h^2(x_i) & h^3(x_i) \\ 0 & 1 & 2h(x_i) & 3h^2(x_i) \end{bmatrix} \quad \mathbf{C}_l = \begin{bmatrix} z_{i-1,\,l} \\ \Delta^x_{i-1,l} \\ z_{i,l} \\ \Delta^x_{i,l} \end{bmatrix}.$$

Likewise one has

$$\frac{d}{dx}[P_{ij}(x, y_l)] = \sum_{r=0,1,2,3} b'_r(y_l)\frac{d}{dx}([h(x)]^r) \tag{19}$$

and the coefficients b'_r (the prime symbol is not related to the derivative) can be determined from the interpolation conditions for the derivatives in order to y of both z and its derivative in respect to x at the corners of the patch. The resulting equation is then

$$\mathbf{B}'_l = \mathbf{H}^{-1}\mathbf{C}'_l, \tag{20}$$

where the matrices $\mathbf{B}'_l$ and $\mathbf{C}'_l$ read

$$\mathbf{B}'_l = \begin{bmatrix} b'_0(y_l) \\ b'_1(y_l) \\ b'_2(y_l) \\ b'_3(y_l) \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & h(x_i) & h^2(x_i) & h^3(x_i) \\ 0 & 1 & 2h(x_i) & 3h^2(x_i) \end{bmatrix} \quad \mathbf{C}'_l = \begin{bmatrix} \Delta^{y}_{i-1,l} \\ \Delta^{xy}_{i-1,l} \\ \Delta^{y}_{i,l} \\ \Delta^{xy}_{i,l} \end{bmatrix}.$$

Finally, the sought coefficients of the bicubic polynomial can be computed:

$$\mathbf{A}_r = \mathbf{K}^{-1}\mathbf{D}_r, \tag{21}$$

where

$$\mathbf{A}_r = \begin{bmatrix} a_{r0} \\ a_{r1} \\ a_{r2} \\ a_{r3} \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & k(x_j) & k^2(x_j) & k^3(x_j) \\ 0 & 1 & 2k(x_j) & 3k^2(x_j) \end{bmatrix} \quad \mathbf{D}_r = \begin{bmatrix} b_r(y_{j-1}) \\ b'_r(y_{j-1}) \\ b_r(y_j) \\ b'_r(y_j) \end{bmatrix}.$$

A similar reasoning is valid for the bicubic interpolating spline Q for the function $w = w(X_1, \ldots, X_n)$, where now X_i denotes the ith spacial coordinate. This will allow reducing a problem in d dimensions to a problem in $d-1$ dimensions. The spline Q may be denoted by

$$\begin{aligned} &Q_{i_1\ldots i_n}(X_1, \ldots, X_n) \\ &\quad = \sum_{r_1,\ldots,r_n=0,1,2,3} \overline{a}_{r_1,\ldots,\,r_n}(X_1 - x^1_{i_1-1})^{r_1} \ldots \left(X_n - x^n_{i_n-1}\right)^{r_n}, \end{aligned}$$

for $X_j \in [x^j_{i_j-1}, x^j_{i_j}]$. Now, Q may be rewritten as

$$\begin{aligned} &Q_{i_1\ldots i_n}(X_1, \ldots, X_n) \\ &\quad = \sum_{r_1,\ldots,r_{n-1}=0,1,2,3} b_{r_1,\ldots,\,r_{n-1}} \left(X_1 - x^1_{i_1-1}\right)^{r_1} \ldots \left(X_{n-1} - x^{n-1}_{i_{n-1}-1}\right)^{r_{n-1}}, \end{aligned}$$

where

$$b_{r_1,\ldots,r_{n-1}} = \sum_{r_n=0,1,2,3} \overline{a}_{r_1,\ldots,r_n} \left(X_n - x^n_{i_n-1}\right)^{r_n}.$$

References

1. J.N. Murrell, S. Carter, S.C. Farantos, P. Huxley, A.J.C. Varandas, *Molecular Potential Energy Functions* (Wiley, Chichester, 1984)
2. P.G. Mezey, *Potential Energy Hypersurfaces* (Elsevier, New York, 1987)
3. A.J.C. Varandas, Adv. Chem. Phys. **74**, 255 (1988)
4. G.C. Schatz, Rev. Mod. Phys. **61**, 669 (1989)
5. A.J.C. Varandas, in *Advanced Series in Physical Chemistry*, chap 5 (World Scientific Publishing, Singapore, 2004), p. 205
6. R. Levine, *Molecular Reaction Dynamics* (Cambridge University Press, UK, 2005)
7. T.V. Albu, J. Espinosa-García, D.G. Truhlar, Chem. Rev. **107**, 5101 (2007)
8. B.J. Braams, J.M. Bowman, Int. Rev. Phys. Chem. **28**, 577 (2009)
9. A.J.C. Varandas, J.N. Murrell, Faraday Discuss. Chem. Soc. **62**, 92 (1977)
10. A.J.C. Varandas, Mol. Phys. **53**, 1303 (1984)
11. A.J.C. Varandas, J. Chem. Phys. **138**, 054120 (2013)
12. D.R. McLaughlin, D.L. Thompson, J. Chem. Phys. **59**, 4393 (1970)
13. N. Sathyamurthy, L.M. Raff, J. Chem. Phys. **63**, 464 (1975)
14. P.M. Prenter, *Splines and Variational Methods* (Wiley, New York, 1975)
15. J.M. Bowman, J.S. Bittman, L.B. Harding, J. Chem. Phys. **85**, 911 (1986)
16. M.A. Collins, S. Petie, A.J. Chalk, L. Radom, J. Chem. Phys. **112**, 6625 (2000)
17. M.A. Collins, Theor. Chem. Acc. **108**, 313 (2002)
18. O. Tishchenko, D.G. Truhlar, J. Chem. Phys. **132**, 084109 (2010)
19. T. Ishida, G.C. Schatz, J. Comput. Chem. **24**, 1077 (2003)
20. R. Dawes, A. Passalacqua, A. Wagner, T.D. Sewell, M. Minkhoff, D.L. Thompson, J. Chem. Phys. **130**, 144107 (2009)
21. T. Hollebeek, T.-S. Ho, H. Rabitz, Annu. Rev. Phys. Chem. **46**, 169 (1999)
22. T.-S. Ho, H. Rabitz, J. Chem. Phys. **119**, 6433 (2003)
23. M.Z. Hussain, M. Sarfraz, J. Comput. App. Math. **218**, 446 (2008)
24. C. Moler, *Numerical Computing with MATLAB* (SIAM, Philadelphia, PA, 2004)
25. Tian M. Int, J. App. Math. **5**, 99 (2011)
26. M. Abbas, A.A. Majid, M.N.H. Awang, J.M. Ali, Appl. Math. Sci. **6**, 291 (2012)
27. W.H. Press, S.A. Teukolski, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing* (Cambridge University Press, New York, 1992)
28. G. Endrödi, Comput. Phys. Commun. **182**, 1307 (2011)
29. P.J. Knowles, M. Schütz, H. Werner, in *Theoretical Chemistry: Advances and Perspectives*, ed. by J. Grotendorst (John von Neumann Institute for Computing, Jülich, NIC Series, Jülich, 2000), p. 97
30. M.R. Pastrana, L.A.M. Quintales, J. Brandao, A.J.C. Varandas, J. Phys. Chem. **94**, 8073 (1990)
31. K.M. Christoffel, J.M. Bowman, B.J. Braams, J. Chem. Phys. **115**(24), 11021 (2001)