A SPARSE INTERPOLATION ALGORITHM FOR DYNAMICAL SIMULATIONS*

J. NANCE† AND C.T. KELLEY†

Abstract. Molecules in nature conform to a geometry that minimizes their potential energy, and some molecules have multiple potential energy minima. One can study how a molecule transitions from one stable geometry to another by studying dynamics on its potential energy surface. The potential energy of a molecule is computed via an expensive optimization process, and thus modeling reaction pathways as a function of all $3N-6$ coordinates can be cumbersome for large molecules. Here we describe a cheaper surrogate model for the potential energy surfaces constructed using a sparse grid interpolation algorithm initially developed by Smolyak [24]. Evaluation of the interpolant is much less expensive than the evaluation of the actual energy function, so our technique offers a more computationally efficient way to study dynamics than traditional methods. Furthermore, molecular vibrations and thermal fluctuations can cause randomness in dynamics, so it is of interest to follow many reaction paths at once, necessitating a fast and efficient implementation of Smolyak's interpolation algorithm. In this paper we describe a new implementation that computes analytical gradients of Smolyak's interpolating polynomial and is designed to evaluate a large number of points simultaneously. We compare performance times of our implementation to MATLAB's Sparse Grid Interpolation Toolbox and present dynamical simulations for the molecule 2-butene.

Key words. Sparse interpolation, sparse grids, Smolyak's algorithm, dynamics, reaction path following

1. Introduction.

In 1963 Smolyak studied tensor product problems and introduced a general approach that uses optimal univariate approximations to construct an almost optimal approximation for the multivariate case [24]. Smolyak's algorithm forms the basis for all sparse grid methods [12], which were introduced in 1991 by Zenger [28]. Zenger proposed a discretization technique that constructs a multidimensional multilevel basis from the tensor product expansion of a one-dimensional multilevel basis. The grid points of the discretization form what is called a "sparse grid," and, compared to full grids, improve the ratio of invested storage and computing time to approximation accuracy [3]. Since their inception, sparse grids have gained a significant amount of traction in the mathematical community, especially since the advent of supercomputers and the need for efficient methods for high dimensional problems. Applications include: fluid flow [9], quantum mechanics [8], stochastic differential equations and optimization [23], economics and finance [14, 4], data mining [7], and uncertainty quantification [27], to name a few. Bungartz and Griebel present a comprehensive review of sparse grids in [3], and the interested reader is referred there for a more thorough introduction.

In this paper we present a new implementation of a sparse grid polynomial interpolation algorithm developed by Barthelmann, Novak, and Ritter in [1] that is based on a reformulation of Smolyak's algorithm from the work of Judd and coworkers in [10]. The reformulation of Smolyak's algorithm eliminates redundant calculations of basis functions by using disjoint set generators for the Smolyak grids and basis functions, instead of the conventional nested set generators [10]. The implementation described

*This work has been partially supported by Army Research Office Grant W911NF-11-1-0367, and NSF Grants CDI-0941253, and 1127914 to the Statistical and Applied Mathematical Sciences Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Army Research Office or the National Science Foundation.

†North Carolina State University, Department of Mathematics, Box 8205, Raleigh, NC 27695-8205, USA (jdnance2@ncsu.edu, Tim_Kelley@ncsu.edu).

in [10] is motivated by the need to evaluate Smolyak's interpolating polynomial at a large number of points for solving dynamic economic models with derivative-free methods. Our implementation is designed to facilitate the integration of dynamics on interpolant, and extends the work of Judd and coworkers in two ways. First, our implementation is capable of evaluating the *analytical* gradient Smolyak's interpolating polynomial, and second, the implementation is designed to quickly evaluate the interpolating polynomial and its gradient at a large number of points *simultaneously*. To test the new algorithm we compare performance times of our implementation to performance times of the robust MATLAB Sparse Grid Interpolation Toolbox [11].

While sparse grids and Smolyak's algorithm are more commonly used in high dimensional problems, the implementation presented here is motivated by a comparatively low dimensional problem where the interpolating polynomial acts as a surrogate model for an expensive function. Specifically, the work presented in this paper is motivated by an application to quantum chemistry, where potential energy surfaces of molecules are approximated by Smolyak's algorithm. The potential energy surface of a molecule describes the energy of an N-atom molecule as a function of its $3N-6$ geometric coordinates. Local minima of these surfaces correspond to stable molecular geometries so there is great interest in studying their structure to follow reaction paths from one stable molecular geometry to another. Current algorithms for reaction path following are computationally burdensome for molecules of moderate size. Molecules of interest can have hundreds of atoms and degrees of freedom so often model or dimension reduction must be applied to efficiently compute the reaction path. We follow an approach from [16] and construct a surrogate model via interpolation on sparse grids and follow reaction paths using a continuous steepest descent method. Molecular vibrations and thermal fluctuations cause variations in reaction paths so it may be necessary to follow dynamics for several hundred or thousand reaction paths at once. The new implementation of Smolyak's algorithm presented in this paper is designed for such a task and allows us to efficiently study reaction path dynamics. As an example we present simulation results for the photoisomerization of 2-butene.

The rest of this paper is outlined as follows: in §2 we review the sparse interpolation algorithm from [1], and in §3 we discuss our new implementation and compare its performance to the MATLAB Sparse Grid Interpolation Toolbox [11]. In §4 we detail how the algorithm is used to construct a surrogate model for reaction path following and present results for a test molecule before concluding in §5.

2. Sparse Interpolation. In this section we outline the sparse interpolation method proposed by Barthelmann, Novak, and Ritter [1]. Unlike Zenger's piecewise linear basis from [28], the method presented here uses global Lagrange interpolating polynomials as basis functions. As we will see in §4, the global smoothness of these basis functions is required by our application.

To understand the multi-dimensional interpolation algorithm, we first consider the one dimensional interpolation problem where we would like to approximate the value of a function $f : [a, b] \to \mathbb{R}$ at some point in the domain. Given a set of $m_i = 2^{i-1} + 1$ nodes $\{x_j\} \in [-1, 1]$ and the corresponding set of function values $\{f(x_j)\}$, we can construct the unique interpolating polynomial of degree $m_i - 1$ denoted by the operator

$$U^i[f](x) = \sum_{i=1}^{m_i} f(x_j^i)\ell_j^i(x) \tag{2.1}$$

where $\ell_j^i(x)$ are the Lagrange basis polynomials

$$\ell_j^i(x) = \prod_{k \neq j} \frac{x - x_k^i}{x_j^i - x_k^i}. \tag{2.2}$$

We will refer to i as the *level* of the interpolation. As suggested in [1], we utilize Chebyshev nodes that are of the form

$$x_j^i = -\cos \frac{\pi(j-1)}{m_i}, \;\; 1 \leq j \leq m_i \tag{2.3}$$

with $x_1^i = 0$ if $m_i = 1$. Note that our choice of m_i leads to sets of nodes that are nested as we increase in level, i.e. if χ^i is the set of nodes for level i we have $\chi^i \subset \chi^{i+1}$.

Expanding this idea to $d > 1$ dimensions, we will use the standard multi-index notation $\mathbf{i} = (i_1, \ldots, i_d)$ and $|\mathbf{i}| = \sum_j i_j$. For $x \in \mathbb{R}^d$ and multi-index $\mathbf{i}$ we define the d-dimensional Lagrange polynomial by a tensor product of one-dimensional Lagrange polynomials:

$$\mathbf{U}^{\mathbf{i}}[f](x) = \bigotimes_{r=1}^{d} U^{i_r}[f](x) \tag{2.4}$$

$$= \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}) \prod_{r=1}^{d} \ell_{j_r}^{i_r}(x_r). \tag{2.5}$$

This tensor product has a poor order of convergence, but serves as the foundation for the more complicated algorithm of Smolyak [19].

In short, Smolyak's interpolation algorithm constructs an interpolating polynomial from a linear combination of Equation 2.5 on different "levels" of sparse grids. Given a degree of exactness k, we define $q = d+k$ and the set of allowable multi-indices

$$Q(q,d) = \{\, \mathbf{i} \in \mathbb{N}^d \,|\, k+1 \leq |\mathbf{i}| \leq q \}. \tag{2.6}$$

Each multi-index $\mathbf{i} \in Q(q,d)$ contains the levels of each dimension's interpolation and can be thought of as representing a different sparse grid on which we must approximate the function. Smolyak's algorithm uses linear combinations of Equation 2.5 on different sparse grids to approximate the multivariate function f, and is given by the operator

$$A(q,d) = \sum_{\mathbf{i} \in Q(q,d)} (-1)^{q-|\mathbf{i}|} \begin{pmatrix} d-1 \\ q-|\mathbf{i}| \end{pmatrix} \mathbf{U}^{\mathbf{i}}. \tag{2.7}$$

To evaluate the Smolyak interpolating polynomial, one only needs to know function values at the sparse grid nodes

$$H(q,d) = \bigcup_{k+1 \leq |\mathbf{i}| \leq q} (\chi^{i_1} \times \ldots \times \chi^{i_d}), \tag{2.8}$$

where $\chi^i = \{x_1^i, \ldots, x_{m_i}^i\}$ is the set of points used by the interpolant U^i. Since the sets of univariate nodes are nested we also have $H(q,d) \subset H(q+1,d)$. Figure 1 shows the nodes for $H(7,2)$ and $H(8,3)$. The total number of unique nodes used by Smolyak's algorithm grows polynomially in d, whereas the number of nodes for conventional

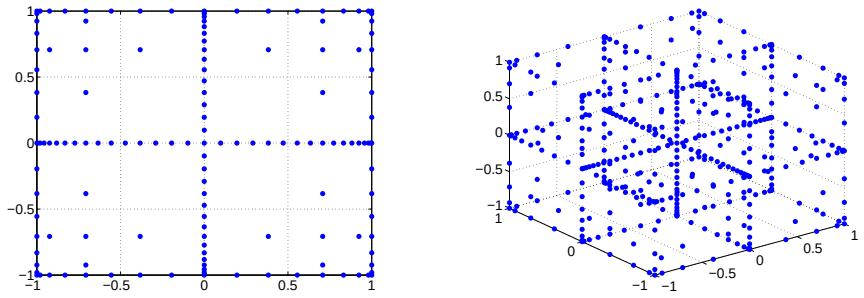

FIG. 1. *Examples of sparse grids for $H(7,2)$ (left) and $H(8,3)$ (right) on the domain $[-1,1]^d$. Notice that the nodes are well-dispersed throughout the domain.*

tensor product grids grows exponentially in d [20]. In this sense, Smolyak's algorithm helps alleviate the curse of dimensionality.

General error and cost bounds for Smolyak's algorithm were derived by Smolyak in [24] and more explicitly by Wasilkowski and Wozniakowski in [25]. Barthelmann et al. also derive error bounds for Smolyak's algorithm in [1] for a certain class of functions and are the ones presented here. Consider the space

$$F_d^k = \left\{ f : [-1,1]^d \to \mathbb{R} \mid D^\alpha f \text{ continuous if } \alpha_i \leq k \; \forall \; i \right\}$$

with norm

$$\|f\| = \max \left\{ \|D^\alpha f\|_\infty \mid \alpha \in \mathbb{N}_0^d, \; \alpha_i \leq k \right\}$$

for $d > 1$. Let I_d denote the embedding $F_d^k \; C(-1,1]^d)$ and

$$\|S\| = \sup \left\{ \|S(f)\|_\infty | f \in F_d^k, \; \|f\| f \leq 1 \right\}$$

for $S : F_d^k \to C([-1,1]^d)$. Letting n be the number of nodes required by $A(q,d)$ and $c_{d,k}$ denote constants that depend only on d and k, then for the space F_d^k,

$$\|I_d - A(q,d)\| \leq c_{d,k} n^{-k} (\log n)^{(k+2)(d-1)+1}. \tag{2.9}$$

Furthermore, $A(q,d)$ will exactly reproduce all polynomials of the form

$$\sum_{|\mathbf{i}|=q} \left(\mathbb{P}_{m_{i_1}} \otimes \ldots \otimes \mathbb{P}_{m_{i_d}} \right)$$

where $\mathbb{P}_m$ is the space of one-dimensional polynomials of degree less than or equal to m [1].

3. Implementation in MATLAB. Previous research has to our knowledge focused on efficient algorithms for evaluating Equation 2.7 simultaneously at one or very few points [12, 18, 17]. Our goal is to evaluate the interpolant at several thousand points at once, and an algorithm that is efficient for, say, 5 points may not be efficient for 10^5 points depending on the algorithm's scalability. As noted by Judd and coworkers in [10], Smolyak's algorithm as written in Equation 2.7 is inefficient because the linear combination causes several basis functions to be evaluated more than

once. Since we are interested in evaluating Equation 2.7 at several thousand points simultaneously we wish to avoid these redundant calculations. For this reason we employ the implementation presented [10] where Smolyak's algorithm is reformulated with disjoint sets of nodes to generate the unidimensional basis functions instead of the nested sets used in the conventional algorithm.

Further computational savings can be found in the construction of the univariate Lagrange basis polynomials (Equation 2.2). Since the denominator does not depend on x, we can precompute the denominator for each basis function ℓ_j, a task which only needs to be completed once. On the other hand, to efficiently evaluate the numerator of Equation 2.2 we take advantage of the fact that the sets of nodes are nested. Consider the one-dimensional interpolation problem and let n be the current level for which we want to evaluate the basis polynomials and let χ_n be the set of m_n nodes for the n^{th} level. Instead of explicitly computing each level's basis polynomials, we can use the basis polynomials from the $(n-1)^{th}$ level. By first defining

$$\ell(x) = \prod_{i=1}^{m_n} (x - x_i), \tag{3.1}$$

the formula for the j^{th} Lagrange basis polynomial of the n^{th} level is

$$\ell_j^{(n)}(x) = \begin{cases} \ell_j^{(n-1)}(x) \displaystyle\prod_{i^{even}} \frac{x - x_i}{x_j - x_i} & x_j \in \chi_{n-1}, \\ \ell^{(n-1)}(x) \left[\displaystyle\prod_{i^{even} \neq j} (x - x_i) \right] \left[\displaystyle\prod_{i \neq j} \frac{1}{x_j - x_i} \right] & x_j \notin \chi_{n-1} \end{cases} \tag{3.2}$$

where i^{even} denotes the indices of nodes that are in χ_n / χ_{n-1}. Note that each $\ell^{(n)}$ can also be computed recursively as

$$\ell^{(n)}(x) = \ell^{(n-1)} \prod_{i^{even}} (x - x_i). \tag{3.3}$$

We use a similar recursive scheme to compute the analytic derivatives of the one-dimensional Lagrange polynomials. Noting that the derivative of the Lagrange polynomial can be written as

$$\frac{d}{dx} \ell_j(x) = \ell_j(x) \sum_{i \neq j}^{m_n} \frac{1}{x - x_i}, \tag{3.4}$$

and $\frac{d}{dx} \ell_j^{(n)}(x)$ can be expressed in a way similar to Equation 3.2. By computing the Lagrange basis polynomials and their derivatives in this way we are able to reduce the overall computational cost of Smolyak's algorithm.

3.1. Numerical Results. Klimke and Wohlmuth developed the Sparse Grid Interpolation Toolbox for MATLAB (see [11, 12] for documentation and algorithm details), and we use this Toolbox as a benchmark for our own algorithm. Consider the task of evaluating Smolyak's interpolation of the function $f(x)$, which we denote by $A(q, d)[f](x)$, at the point $x \in \mathbb{R}^d$. Our implementation and the Toolbox's both work in two steps. In the first step, everything that can be calculated without knowledge of x is computed. This step takes as input the function $f(x)$, the bounds of interpolation, and the degree of polynomial exactness k, and computes the sparse

grid points, the multi-index set $Q(q,d)$, the coefficients for Smolyak's algorithm, and any bookkeeping data structures. The second step evaluates Smolyak's interpolant at x and involves computing the univariate Lagrange basis polynomials and the tensor products (Equation 2.5) dictated by Smolyak's algorithm.

As detailed in §4, our application requires that we integrate dynamics on the interpolating polynomial continuously $A(q,d)[f](x)$, meaning we must evaluate the same interpolating polynomial and its derivative several thousand times during simulations. With this in mind, we compare performance times of MATLAB's Toolbox and our own implementation only for the second step of the implementation process. The first step is a one-time computational cost and is negligible compared to the cost of simulating dynamics on the interpolant. We note, however, that for larger values of d and k the cost of computing the Smolyak coefficients for our implementation can be quite cumbersome. We calculate the coefficients as suggested in [10], where Judd and coworkers compute them by solving a linear system of equations. As one would expect, the associated matrix of this system becomes larger and more ill-conditioned as d and k increase. Computing the coefficients in this way is acceptable for our application, though, since d and k remain relatively small ($d \leq 5$ and $k \leq 6$).

In all Figures shown the method presented in this paper is referred to as "New Method" and the MATLAB Toolbox is referred to as "Toolbox." Table 1A shows performance scalings in dimension d for evaluating Smolyak's interpolant and its gradient at one point with $k = 4$, and Table 1B shows performance scalings in degree of exactness k for evaluating Smolyak's interpolant and its gradient at one point in 4 dimensions.. It is clear that the New Method outperforms the MATLAB Toolbox for all dimensions $d \leq 8$ and degrees of exactness $k \leq 7$. Table 1C shows performance scalings in the simultaneous evaluation of N points, where N increases in powers of 10 from from 1 to 10^5. Each computation was performed with $d = 4$ and $k = 5$ and also evaluated gradients. Here the New Method greatly outperforms the Toolbox. While the Toolbox is useful for the simultaneous evaluation of a few points, it was not designed to approximate a function at a large number of points. The code does not vectorize interpolant evaluation of points, and thus the simultaneous computation of several thousand points is quite expensive.

TABLE 1
Performance times (in seconds) for the New Method and MATLAB Sparse Interpolation Toolbox.

A: Dimension

	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$	$d=7$	$d=8$
New Method	8.4E-3	1.6E-2	4.1E-2	1.0E-1	2.4E-1	4.8E-1	9.2E-1
Toolbox	1.8E-2	4.7E-2	9.3E-2	1.8E-1	3.5E-1	6.0E-1	9.6E-1

B: Degree of Exactness

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
New Method	4.2E-3	1.0E-2	2.5E-2	5.8E-2	7.5E-2	1.2E-1	1.9E-1
Toolbox	1.2E-2	3.2E-2	8.5E-2	2.2E-1	5.9E-1	1.2E+0	2.6E+0

C: Number of Points

	$N=1$	$N=10$	$N=10^2$	$N=10^3$	$N=10^4$	$N=10^5$
New Method	2.8E-1	1.1E-1	7.8E-2	1.2E-1	8.9E-1	9.6E+1
Toolbox	4.2E-1	3.5E-1	9.8E-1	7.8E+0	7.5E+1	7.4E+2

4. Application to Quantum Chemistry. In this section we describe our surrogate model for reaction path following. The model can be broken into two steps, the first of which is to approximate the potential energy surface (PES). The second step is to simulate dynamics on the approximated PES.

4.1. Potential Energy Surface Approximation. An N-atom molecule is a function of $3N-6$ geometric coordinates (bond lengths, bond angles, and dihedral angles), and for many reactions only a handful of these coordinates change significantly and the rest remain approximately constant [2]. As such, the first step towards reducing the dimensionality of our problem is to isolate these few, say d, molecular coordinates and study the reaction as a function of only these coordinates. We then construct the d-dimensional PES's on which the reactions take place with Smolyak's sparse interpolation algorithm [24, 1, 10].

For each point in the sparse grid we must compute the energy to interpolate the true PES. The energy at quantum state n, $E_n(p)$, is computed as a function geometric coordinates $p \in \mathbb{R}^{3N-6}$. The first step of our method is to partition $p = (x, \xi)$ into a vector of design variables $x \in \mathbb{R}^d$ and a vector of remainder variables $\xi \in \mathbb{R}^{3N-6-d}$, where chemical knowledge or intuition of the system guides the appropriate choice of design variables x. After specifying appropriate bounds and choosing a degree of exactness k, for each point $x_i \in H(q,d)$ we compute the ground state energy via the constrained optimization problem

$$\hat{E}_0(x_i) = \min_{\xi} E_0(x_i, \xi) \tag{4.1}$$

where the minimization is only over the remainder variables ξ. $E_0(p)$ approximates the energy of the associated time-independent Schrödinger equation using density functional theory (DFT), which is in itself an expensive iterative process. Excited state energies $\hat{E}_n(x_i)$ $(n \geq 1)$ are computed at the respective optimized ground state geometries using the time-dependent DFT method [22]. With the sets of sparse grid points x_i and energy values $\hat{E}_n(x_i)$ (now with $n \geq 0$) in hand, we use Smolyak's interpolation algorithm [24] to approximate the PES's. We will denote the surrogate model for the energy by $E_n^s(x) \approx \hat{E}_n(x)$ for any energy state $n \geq 0$.

4.2. Reaction Path Dynamics. The goal of our simulations is to successfully predict the natural relaxation of a molecule from excited states and track the entire reaction path. Each simulation begins at an equilibrium geometry on the ground state PES, from which the molecule is excited to the first specified energy state. The reaction path of a molecule moves in the direction of the negative gradient on PES's [15]. Large steps across the PES's may pass over local minima or be physically unnatural, so we integrate dynamics continuously via continuous steepest descent [13, 5]. This method integrates the dynamics

$$\dot{x} = -\nabla E_n^s(x) \tag{4.2}$$

until $\|\nabla \hat{E}_n^s(x)\|$ is sufficiently small. The solution x to Equation 4.2 is the reaction path and is approximated using a Runge-Kutta 4-5 method with Dormand-Prince coefficients and a variable time step [21]. The time step is arbitrary and in no way reflects the time scale of the physical reaction.

Once a local minimizer x_{min}^s of the excited state PES has been found, we simulate the relaxation of the molecule. Upon each relaxation to a lower energy level, thermal fluctuations and molecular vibrations are accounted for by reinitializing dynamics on

the succeeding energy state at $x^s_{min} + \gamma$ where γ is a random variable. The domain γ depends on the type of molecular coordinate: if the i^{th} design variable is a bond length, $\gamma_i \in [-.05,\ .05]$ and if the i^{th} design variable is a bond or dihedral angle, $\gamma_i \in [-30,\ 30]$. These intervals were chosen arbitrarily to approximate the effects of molecular vibrations and thermal fluctuations.

4.3. 2-Butene. A good test molecule for our simulations is 2-butene because it has a known photoisomerization transition path [26]. The two stable geometries for 2-butene are *cis*-2-butene and *trans*-2-butene, both of which are shown in Figure 2. The simulation consists of two design variables: the main isomerization coordinate

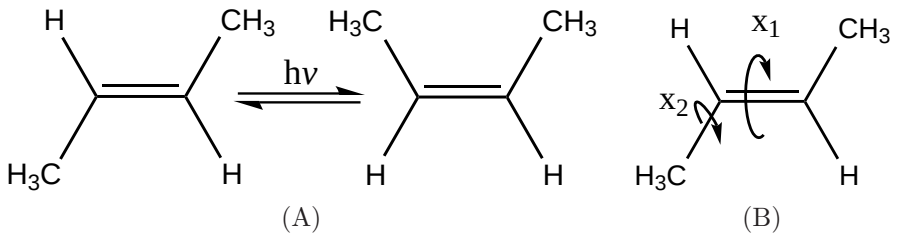

FIG. 2. *A: Photoisomerization of* trans-*2-butene (left) and* cis-*2-butene. B: Degrees of freedom for simulations.*

(x_1) is a torsion over the central carbon double bond [26], and the second coordinate (x_2) is the torsion of one of the other carbon bonds. Both of these dihedral angles are shown in Figure 2. The two stable geometries correspond to $x_1 = 0$ and $x_2 = 180$ for *cis*-2-butene and *trans*-2-butene, respectively. For each coordinate all associated dihedral angles are held constant during the constrained optimization energy calculations (Equation 4.1). The units for reported energies, angles, and bond lengths are electron volts, degrees, and Ångstroms, respectively. All quantum chemistry calculations are performed with the GAUSSIAN 09 software package [6].

The bounds for the two-dimensional PES's are $[-10, 190]$ for x_1 and $[-10, 130]$ for x_2, and the degree of polynomial exactness for Smolyak's algorithm is $k = 5$. Each optimization is calculated at with the B3LYP DFT method and the CEP-31G* basis set. The approximated PES's for both the ground and first singlet excited state are shown in Figure 3A. One can visually observe that the global minimum of the first excited state lies directly above the global maximum of the ground state, so it is clear that randomness caused by molecular vibrations and thermal fluctuations could play an important role in the reaction.

The reaction path is initialized on the ground state at $\bar{x} = (0.0,\ 7.1)$, corresponding to a *cis*-2-butene geometry of our surrogate PES. After excitation to the first excited singlet state, continuous steepest descent converges to a minimizer of $\bar{x}^1 = (90,\ 68)$ on the first excited state PES. Upon relaxation, the effects of thermal fluctuations and molecular vibrations either send the molecule back to its original *cis* structure or towards its *trans* counterpart. Twenty different reaction paths are shown on the PES's in Figure 3B. The simulations find three stable geometries for both *cis* and *trans* corresponding to three different values of x_2. These simulations not only demonstrate the role that x_2 plays in the photoisomerization reaction, but also reveal a large number of 2-butene local minima. Simulations have also been performed with three degrees of freedom and will be reported in separate paper.

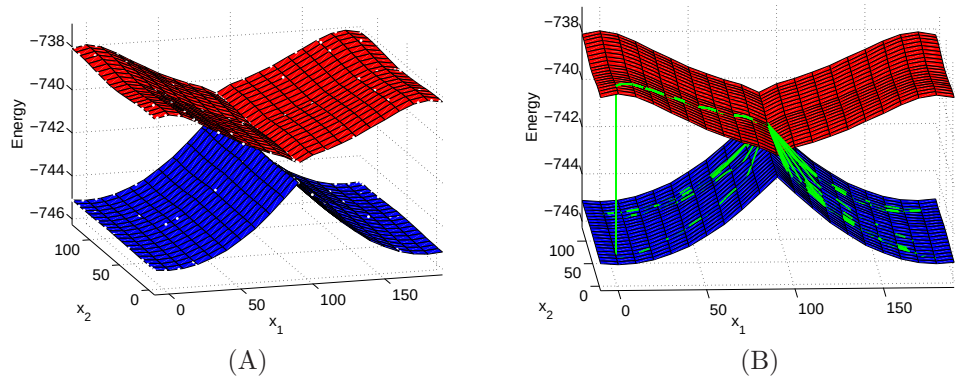

FIG. 3. *A: Ground and first excited state PES's generated with Smolyak's interpolation algorithm with $k = 5$. B: 20 simulation paths with effects of molecular vibrations and thermal fluctuations upon relaxation to ground state.*

5. Conclusion. In this paper we have reviewed interpolation on Chebyshev sparse grids using Smolyak's algorithm [24, 1] and described an efficient implementation based on the method from [10]. Numerical test comparing our method to MATLAB's Sparse Interpolation Toolbox [11] show that our method outperforms the Sparse Toolbox in all cases considered. The greatest computational savings come when simultaneously evaluating several thousand points. We have also described a surrogate model that uses Smolyak's algorithm to interpolate molecular PES's and integrate dynamics to simulate reaction paths. The surrogate model allows one to continuously follow reaction paths, a task that is traditionally too computationally burdensome for traditional quantum chemistry methods. Finally, we have presented simulation results for the photoisomerization of 2-butene. The potential energy surrogate model presented in this paper can be used to study isomerization reactions and other quantum phenomena.

REFERENCES

[1] VOLKER BARTHELMANN, ERICH NOVAK, AND KLAUS RITTER, *High dimensional polynomial interpolation on sparse grids*, Advances in Computational Mathematics, 12 (2000), pp. 273–288.

[2] ADAM B. BIRKHOLZ AND H. BERNHARD SCHLEGEL, *Coordinate reduction for exploring chemical reaction paths*, Theoretical Chemistry Accounts, 131 (2012), p. 1170.

[3] HANS-JOACHIM BUNGARTZ AND MICHAEL GRIEBEL, *Sparse grids*, Acta Numerica, 13 (2004), pp. 1–123.

[4] HANS-JOACHIM BUNGARTZ, ALEXANDER HEINECKE, DIRK PFLÜGER, AND STEFANIE SCHRAUFSTETTER, *Option pricing with a direct adaptive sparse grid approach*, Journal of Computational and Applied Mathematics, 236 (2012), pp. 3741–3750.

[5] R COURANT, *Variational Methods for the Solution of Problems of Equilibrium and Vibrations*, Bull. Amer. Math. Soc., 49 (1943), pp. 1–43.

[6] M J FRISCH, G W TRUCKS, H B SCHLEGEL, G E SCUSERIA, M A ROBB, J R CHEESEMAN, G SCALMANI, V BARONE, B MENNUCCI, G A PETERSSON, H NAKATSUJI, M CARICATO, X LI, H P HRATCHIAN, A F IZMAYLOV, J BLOINO, G ZHENG, J L SONNENBERG, M HADA, M EHARA, K TOYOTA, R FUKUDA, J HASEGAWA, M ISHIDA, T NAKAJIMA, Y HONDA, O KITAO, H NAKAI, T VREVEN, J A MONTGOMERY JR., J E PERALTA, F OGLIARO, M BEARPARK, J J HEYD, E BROTHERS, K N KUDIN, V N STAROVEROV, R KOBAYASHI, J NORMAND, K RAGHAVACHARI, A RENDELL, J C BURANT, S S IYENGAR, J TOMASI, M COSSI, N REGA, J M MILLAM, M KLENE, J E KNOX, J B CROSS, V BAKKEN, C ADAMO,

J Jaramillo, R Gomperts, R E Stratmann, O Yazyev, A J Austin, R Cammi, C Pomelli, J W Ochterski, R L Martin, K Morokuma, V G Zakrzewski, G A Voth, P Salvador, J J Dannenberg, S Dapprich, A D Daniels, . Farkas, J B Foresman, J V Ortiz, J Cioslowski, and D J Fox, *Gaussian 09, Revision A.1*, 2009.

[7] Jochen Garcke and Michael Griebel, *Data mining with sparse grids using simplicial basis functions*, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01, (2001), pp. 87–96.

[8] V. Gradinaru, *Fourier transform on sparse grids: Code design and the time dependent Schrödinger equation*, Computing, 80 (2007), pp. 1–22.

[9] Michael Griebel and Veronika Thurner, *The efficient solution of fluid dynamics problems by the combination technique*, International Journal of Numerical Methods for Heat & Fluid Flow, 5 (1995), pp. 251–269.

[10] Kenneth L. Judd, Lilia Maliar, Serguei Maliar, and Rafael Valero, *Smolyak Method for Solving Dynamic Economic Models: Lagrange Interpolation, Anisotropic Grid and Adaptive Domain*, Working Paper 19326, National Bureau of Economic Research, Cambridge, MA, 2013. Retrieved January 5, 2014, from http://www.nber.org/papers/w19326, pp. 1-47.

[11] Andreas Klimke, *Sparse Grid Interpolation Toolbox User's Guide*, tech. report, Berichte aus dem Institut fur Angewandte Analysis und Numerische Simulation, 2008.

[12] Andreas Klimke and Barbara Wohlmuth, *Algorithm 847: spinterp : Piecewise Multilinear Hierarchical Sparse Grid Interpolation in MATLAB*, ACM Transactions on Mathematical Software, 31 (2005), pp. 561–579.

[13] X.-L. Luo, C. T. Kelley, L.-Z. Liao, and H. W. Tam, *Combining Trust-Region Techniques and Rosenbrock Methods to Compute Stationary Points*, J Optim Theory Appl, 140 (2009), pp. 265–286.

[14] Benjamin A. Malin, Dirk Krueger, and Felix Kubler, *Solving the multi-country real business cycle model using a Smolyak-collocation method*, Journal of Economic Dynamics and Control, 35 (2011), pp. 229–239.

[15] R M Minyaev, *Reaction Path as a Gradient Line on a Potential Energy Surface*, International Journal of Quantum Chemistry, 49 (1994), pp. 105–127.

[16] D. S. Mokrauer, C. T. Kelley, and A. Bykhovski, *Simulations of Light-Induced Molecular Transformations in Multiple Dimensions with Incremental Sparse Surrogates*, Journal of Algorithms & Computational Technology, 6 (2012), pp. 577–592.

[17] Alin Murarasu, Gerrit Buse, Dirk Pfluger, Josef Weidendorfer, Bode Arndt, Alin Murarau, Dirk Püger, and Arndt Bode, *fastsg : A Fast Routines Library for Sparse Grids*, Procedia Computer Science, 9 (2012), pp. 354–363.

[18] Alin Murarasu, Dirk Pflüger, Josef Weidendorfer, Gerrit Buse, and Daniel Butnaru, *Compact Data Structure and Scalable Algorithms for the Sparse Grid Technique*, in PPoPP, ACM, 2011, pp. 1–10.

[19] Erich Novak and Klaus Ritter, *High dimensional integration of smooth functions over cubes*, Numerische Mathematik, 75 (1996), pp. 79–97.

[20] ———, *Simple Cubature Formulas with High Polynomial Exactness*, Constructive Approximation, 15 (1999), pp. 499–522.

[21] P J Prince and J R Dormand, *High order embedded Runge-Kutta formulae*, Journal of Computational and Applied Mathematics, 7 (1981), pp. 67–75.

[22] Erich Runge and E. K. U. Gross, *Densitfy-Funcational Theory for Time-Dependent Systems*, Physical Review Letters, 52 (1984), pp. 997–1000.

[23] Ch. Schwab and R.A. A. Todor, *Sparse Finite Elements for Stochastic Elliptic Problems - Higher Order Moments*, Computing, 71 (2003), pp. 43–63.

[24] S Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Soviet Math. Dokl., 4 (1963), pp. 240–243.

[25] Grzegorz W. Wasilkowski and Henryk Wozniakowski, *Explicit Cost Bounds of Algorithms for Multivariate Tensor Product Problems*, Journal of Complexity, 11 (1995), pp. 1–56.

[26] D. L. Woolard, R. Brown, M. Pepper, and M. Kemp, *Terahertz Frequency Sensing and Imaging: A Time of Reckoning Future Applications?*, Proceedings of the IEEE, 93 (2005), pp. 1722–1743.

[27] Dongbin Xiu, *Efficient Collocational Approach for Parametric Uncertainty Analysis*, Communications in Computational Physics, 2 (2007), pp. 293–309.

[28] Christoph Zenger, *Sparse Grids*, Notes on Numerical Fluid Mechanics, 31 (1991), pp. 241–251.