Computer Physics Communications 184 (2013) 1150–1154

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Accelerating modified Shepard interpolated potential energy calculations using graphics processing units

Hong Fu, Limin Zheng, Minghui Yang [*]

Key Laboratory of Magnetic Resonance in Biological Systems, State Key Laboratory of Magnetic Resonance and Atomic and Molecular Physics, Wuhan Centre for Magnetic Resonance, Wuhan Institute of Physics and Mathematics, Chinese Academy of Sciences, Wuhan, 430071, People's Republic of China

A R T I C L E I N F O

Article history:
Received 18 September 2012
Received in revised form
5 December 2012
Accepted 8 December 2012
Available online 14 December 2012

Keywords:
Graphics processing unit acceleration
Chemical reactive dynamics
Potential energy surface
Modified Shepard interpolation scheme

A B S T R A C T

The potential energy surfaces constructed with the modified Shepard interpolation scheme have been widely used in studies of chemical reaction dynamics. However, computational costs of interpolation increase rapidly with the size of the system and the number of data points needed to achieve a given accuracy. In this work, we present a naive Graphics Processing Unit (GPU)-accelerated algorithm for modified Shepard interpolated potential energy calculations and its implementation with the PGI CUDA Fortran language. The benchmark tests on a NVIDIA Tesla C2050 using four interpolated potential energy surfaces (one for $H + H_2O \leftrightarrow H_2 + OH$, two for $H + NH_3 \leftrightarrow H_2 + NH_2$ and one for $H + CH_4 \leftrightarrow H_2 + CH_3$) demonstrated a speedup of 50-fold over the original CPU implementation on an Intel E5620 processor and the speedup increases with the system size and the number of data points. This work presents a promising GPU application in the field of chemical reaction dynamics.

1. Introduction

Graphics processing units (GPUs) are designed for acceleration in image building in a frame buffer in order to address the demands of real-time high-resolution 3D graphics computational-intensive tasks. The highly parallel multi-core structure of GPUs makes them very effective to process large blocks of data in parallel. Recently, the graphics card manufacturer NVIDIA released the "compute unified device architecture" (CUDA) development toolkit for high-end graphics cards. CUDA allows developers to program in a C-like language and to take full advantage of NVIDIA's accelerator so that the complexity for using a GPU in general-purpose scientific computing is reduced substantially. Nowadays, the GPU has emerged as a popular platform for high performance computing in many different areas [1].

One of the main applications of GPU in computational chemistry is molecular dynamics (MD) simulations which describe the motions of atoms in bio-molecules or complex molecular systems by solving Newton's second law of motion and provide important information on dynamics and thermodynamics properties [2–11]. One of the main challenges of MD simulations is that the simulation time should be long enough to sample most important conformation spaces. GPUs provide an economical means to accelerate MD simulations [2,6]. As a result, many of the mainstream MD simulation software packages (Amber [11], Gromacs [12], NAMD [13], LAMMPS [4], etc.) have incorporated GPU acceleration and some new software packages such as HOOMD-Blue [14] and ACEMD [2] have been developed especially on GPU architecture. Recently, Pande and coworkers have released the OpenMM library of GPU kernels for MD simulations [15]. So far, GPU acceleration has been demonstrated to achieve speedups from around 10 to 100 for MD simulations [13].

Another important application of GPUs is quantum chemistry [16–29]. The first application in this area is the GPU implementation of the Quantum Monte Carlo method by Anderson and coworkers [18]. Afterwards, Yasuda reported GPU applications to accelerate two-electron integral evaluation (10× speedup) and density functional calculations [19]. Martinez and coworkers have implemented a GPU algorithm for two-electron integrals (with a 130× speedup), direct self-consistent field calculations (with speedups ranging from 28× to 650×), analytical energy gradients, geometry optimization and first principles molecular dynamics on GPU [16,17]. They also released "TeraChem"—the first general-purpose quantum chemistry software package [20–22]. Others include Asadchev and coworkers who implemented the two-electron integral calculations up to *G* functions on GPUs [23], and Luigi Genovese et al. who implemented the GPU accelerating code based on BigDFT and achieved a speedup factor of 6 [24]. More recently, efforts have also been made to accelerate electronic correlation calculations [25–29].

In this work, GPU acceleration is applied to the field of chemical reaction dynamics in which the nuclear motion in a molecular system is solved quantum mechanically (e.g. the time-dependent

* Corresponding author. Tel.: +86 27 87197783; fax: +86 027 87199291.
E-mail address: yangmh@wipm.ac.cn (M. Yang).

wave-packet quantum dynamics method) or classical mechanically (e.g. the quasi-classical trajectory method) under the Born–Oppenheimer approximation. The concept of potential energy surface (PES) is introduced to describe the relation between energy and geometry of the system. As the number of geometries needed to construct an accurate PES increases exponentially with the size of the system, the computational costs for *ab initio* calculations and validation of the PES also increase rapidly. In fact, the number of geometries denoted as "grid points" in quantum dynamics calculations for the potential integral is often much larger than the number of expansion coefficients of wavefunctions in a basis. For example, the number of grid points exceeds 3.1×10^{10} in full-dimensional quantum dynamics studies of the $H + NH_3$ reaction, whereas the number of expansion coefficients is 3.9×10^9, resulting in a dramatic increase of computational costs.

The PES could be generated with various methods, including analytic functional fitting [30,31], spline interpolation [32], modified Shepard interpolation [33], interpolating moving least squares [34], and neural networks [35]. In this work, the GPU acceleration is applied to the modified Shepard interpolation scheme developed by Collins and coworkers [33,36–38]. This scheme calculates potential energy from *ab initio* energies and energy derivatives of known points in the PES data set and produces these points with the aid of classical trajectory simulations of reaction dynamics. Crittenden and Jordan have summarized the desirable properties of the modified Shepard interpolation scheme: it has relatively low costs in computation and exactly reproduces the original *ab initio* data, and can be applied to a large range of chemical problems [39]. However, because the potential energy of an arbitrary configuration is expressed as a weighted average of Taylor series expansions about the points in the PES data set, the computational costs scale linearly with the points in the PES data set and exponentially with the size of the system. As a result, the computational costs of the potential energy calculation with the modified Shepard interpolation scheme is usually much larger than that with an analytic potential energy function.

In this work, we present a GPU-based implementation of the modified Shepard interpolation scheme and show that this GPU acceleration could be useful for the development or evaluation of the interpolated PES and in the studies of chemical reaction dynamics.

2. The modified Shepard interpolation scheme and CPU/GPU algorithms

Because details of the modified Shepard interpolation scheme have been described in many publications [33,36–39], this scheme together with its CPU algorithm will be introduced briefly in the first subsection. In the next two subsections, we present the GPU algorithms for the exact implementation and an approximate implementation of the interpolation scheme. For simplicity, the molecular configuration whose potential energy to be calculated is denoted as the "grid point" and is distinguished from the "data point" defined in the PES data set. The term "grid point" comes from the grids defined in the quantum reaction dynamics calculations, although the GPU algorithm presented here could also be applied in quasi-classical trajectory (QCT) calculations.

2.1. The interpolation scheme and CPU algorithm

For an arbitrary grid point, its energy is expressed as a weighted average of Taylor series expansions of data points in the PES data set:

$$V(Z) = \sum_{i=1}^{ndata*ngroup} w_i(Z) T_i(Z), \qquad (1)$$

where *ndata* is the number of original *ab initio* data points in the PES. By applying the molecular symmetry group elements to the original data points, the total number of data points used in the interpolation is *ndata* ∗ *ngroup*, where *ngroup* is the number of elements in the symmetry group. $Z = \{Z(X)\}$ are the interpolation coordinates and X is the $3 \times natom$ Cartesian coordinates of atoms in the molecule. Because the bond lengths are the smallest set of invariants that can give a global description of the molecular structure and their inverses were found to result in more accurate Taylor expansions for bond stretching potential functions, the PES is constructed with inverse inter-atomic distance coordinates in Collins' modified Shepard interpolation scheme,

$$Z = \{1/R_l\} \ (l = 1, \ldots, nbond), \qquad (2)$$

here $nbond = natom \times (natom - 1)/2$. $T(Z, i)$ is the second-order Taylor expansion about the near data point $Z(X(i))$,

$$T_i(Z) = V|_{Z(i)} + \sum_{l=1}^{nbond} \left.\frac{\partial V}{\partial Z_l}\right|_{Z(i)} [Z - Z_l(i)] + \frac{1}{2} \sum_{k=1}^{nbond} \sum_{l=1}^{nbond} \left.\frac{\partial^2 V}{\partial Z_k \partial Z_l}\right|_{Z(i)} [Z - Z_k(i)][Z - Z_l(i)] \qquad (3)$$

$v_i(Z)$ and $w_i(Z)$ represent the weight and relative weight of grid point $\boldsymbol{Z}$ with respect to the i-th data point, respectively,

$$w_i(Z) = \frac{v_i(Z)}{v_{\text{tot}}(Z)} \qquad (4)$$

and $v_{\text{tot}}(Z)$ is the summation of the weights

$$v_{\text{tot}}(Z) = \sum_{j}^{ndata*ngroup} v_j(Z). \qquad (5)$$

In practical calculations, only the data points with relative weights larger than the tolerance *wtol* are used in the Taylor expansions and the number of these data points is denoted as *nforc*,

$$V(Z) = \sum_{i=1}^{nforc} w_i(Z) T_i(Z). \qquad (6)$$

However, the selection of data points by their relative weights also introduces discontinuities in the PES, as discussed in Jordan's work [40].

Usually a two-part weight function is used with parameters $p = 2$ and $q = 12$ [41],

$$v_i(Z) = \frac{1}{\left(\sum_{l}^{nbond} \left[\frac{Z_l - Z_l(i)}{d_l(i)}\right]^2\right)^p + \left(\sum_{l}^{nbond} \left[\frac{Z_l - Z_l(i)}{d_l(i)}\right]^2\right)^q} \qquad (7)$$

$d^l(i)$ is the confidence length of l-th coordinates for the i-th data point and is viewed as an accuracy estimate of the Taylor expansion in each coordinate direction.

The inverse inter-atomic distance coordinate system is a redundant set of coordinates as there are only $nint = 3 \times natom - 6$ independent internal coordinates. Thus a set of *nint* internal coordinates ζ could be constructed as linear combinations of $\boldsymbol{Z}$ for each data point in the PES data set. Also, the *nint* local internal coordinates of grid points could be expressed as linear combinations of $\boldsymbol{Z}$, if it is sufficiently close to some data point $Z^0 = Z(X^0)$

$$\zeta^{X_0}(X) = U^T Z. \qquad (8)$$