3P124

Multiresolution Multiwavelet 基底による Ehrenfest Dynamics プログラム高速化の検討

（豊橋技術科学大） ○濱田信次、関野秀男

化学現象のメカニズムを基本から理解し、材料設計への応用を目指す際に、核と電子の同時シミュレーションは非常に重要な道具である。我々はその中でも、wavelet基底を用いてEhrenfest Dynamicsを効率的に計算することに取り組んできた[1]。wavelet基底はシステマティックで高効率な関数展開を可能にするものであり、特に電子状態に関する先見的情報が得にくいDynamicsにおいては非常に有効であると考えられる。核を古典、電子を量子で取り扱うDynamics手法としては、Born-Oppenheimer MD, Car-Parrinello MD, Surface Hoppingなどがあるが、Ehrenfest Dynamicsは、数学的には最も首尾一貫したDynamics手法である。なお、我々は最終的にはミクロからマクロまで種々の階層でのシミュレーションを目指して、より一般的な時間発展方程式をwaveletで効率的に解くことを試みている。[2,3]。waveletにも様々な種類があるが、我々はAlpertのmultiwaveletと呼ばれるwaveletを使用している。このwaveletはDaubechiesのwaveletと並んで、数値計算上非常に好ましい、compact support(=空間的にlocal)という性質をもっている。

waveletを使った量子化学計算を高速に行う場合、もちろんアルゴリズムの選択がもっとも重要である。waveletの表現形式（Standard表現、Non-Standard表現、圧縮表現、再構成表現)、変換軸の分離による高速変換技法、truncate（=重要なwavelet基底のみを抽出すること）方法などを状況に応じて細心の注意を払って選択する必要がある。一連の計算処理において精度と計算量のバランスが均等に分配されるようにしなければならない。

これらのアルゴリズムの検討がある程度行き着くところまで行った後には、言語、コンパイラ、CPU、キャッシュ、並列化、ハードウェア化などの物理的な高速化を考える段階となる。正確には、通常、量子化学の計算量は莫大であるので、アルゴリズムの検討段階でも、高速計算出来なくては、効率的な検討ができない。

量子化学計算におけるプログラミング言語としてはC/C++やfortranが主流であるが、我々は学習の容易さ、記述の美しさ、生産性の高さ、実用的ライブラリーの豊富さなどを兼ね備えた非常に優れた言語であるpythonを使ってプログラム開発を行っている。pythonはアルゴリズム検討段階では最適な言語の一つだといえる。然し、pythonのような動的スクリプト系の言語は利便性の代償として実行時に種々のチェックを行うため、コンパイル型の言語であるC/C++やfortranに比べると、実行速度は一桁程度遅くなる場合が多い。このため律速処理のみをC/C++に置き換える手法がいろいろと試みられている。Cython言語を使う手法もその中の一つである。Cython言語とはPython言語に型宣言を付加可能にしたPython-likeな言語である。Cythonコンパイラによって、Cython言語コードがC言語コードに変換され、これからCコンパイラ(MSC、GCCなど)を用いて、PythonのC拡張モジュールと呼ばれるものを生成する。Cythonを使用する最大のメリットは、最小限の手間で、Pythonコードをコンパイラ型言語コードに変換できるということである。すなわち高生産性アルゴリズム開発と高速計算とを両立できるということである。

なお、Pythonの世界ではnumpyという非常に優れた配列(=テンソル)処理モジュールが存在するが、Cythonはこのnumpyに対応していることも大きな利点である。

[高速化の具体的検討]

まずは、我々のPythonプログラムのうち、高速化検討が必要と思われる全モジュールについて、Cython版モジュールを作成した。Cython言語はPython言語の上位互換を目指した言語であるため、基本的にはこれは容易な作業である。（ただし、Cythonはまだ開発途上であるため、いくつかの制限事項に注意する必要がある。）なお、各Python版モジュールとそれに対応するCython版モジュールとは簡単に切り替え可能にしておく。（少し手間は増えるが、現実には２つの相互互換な版を作っておくことがいろいろな意味でメリットがある。）

pythonのloggingツール、profilingツール等を使って、各処理の実行時間を分析し、律速処理を同定する。律速処理について、Cython版の対応する部分に型宣言を付加して、高速化を試みる。

[高速化可能な部分]

当然ながら、Cythonに変換することによる効果の大きなコード部分と、それほどでもないコード部分が存在する。我々のプログラムはpythonの辞書やリストなどの高機能データ型を多用している。

これらをCythonに置き換えても、大きな効果はない。正確にはPythonはもともと十分高速である。

numpyの縮約計算なども多用しているが、これも元々Cで作成されており、あまり高速化の余地はない。効果が大きいのは、通常のスカラー処理や、numpyのarray要素を直接操作する処理である。

具体的例としては関数のsamplingやDFTのfunctionalの計算であるが、詳細は当日報告予定である。

[精度検証]

精度と計算速度は相反するので、計算速度の検討の際には、精度にも細心の注意を払う必要がある。当然ならが、各要素処理ごとの精度と計算速度を検証しながら検討しているが、Ehrenfest Dynamicsの場合、総合的な精度検証としては、おもに全Energy保存で行っている。

Fig. 1, 2にそれぞれ2次元水素分子の振動回転運動におけるEnergy保存のグラフと２個の核の軌跡を示す。電子運動部分はTDRHF(Time Dependent Restricted Hatree-Fock)を使用している。

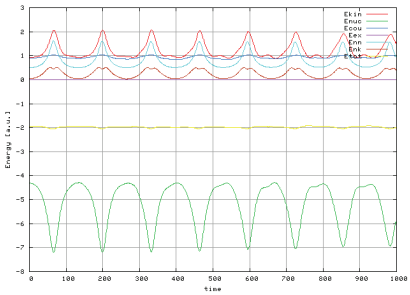

Fig.1　全Energy保存のグラフ

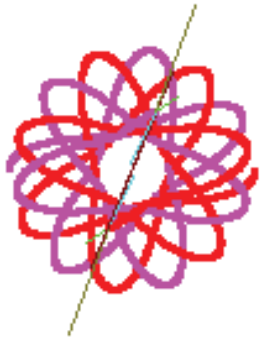

Fig.2　２個の核の軌跡

【参考文献】

[1] 第10回理論化学討論(2007)

ウェーブレットを用いたTDHF/TDDFT実時間シミュレーション　濱田信次、関野秀男

[2] 第11回理論化学討論(2008)

ウェーブレットを用いた流体力学形式の時間依存方程式　濱田信次、関野秀男

[3] IEEE, *2(2)* (2008) 654-657,　Wavelet Analysis and Pattern Recognition

“Solution of advection equation using wavelet basis sets”, Hideo Sekino and Shinji Hamada