3P115

量子力学的振動状態計算法に対する関数型プログラミング言語の適用

（慶大院・理工）　　○菅原道彦

Application of functional programming language to quantum mechanical vibrational state calculation

(Graduate School of Science and Technology, Keio Univ.)　○M. Sugawara

【序】近年、開発効率を意識したソフトウェア資産のモジュール化・並列プログラミングへの親和性などの理由により、関数型プログラミング言語が注目されている。一方で、科学技術計算用のプログラミング言語としては手続き型言語である FORTRAN、C などが広く用いられているが、数値計算用ライブラリなどを含むレガシーコードの活用など考慮すると、今後もこれらの手続き型言語が科学技術計算の主流である可能性は高い。しかし、後発の言語と比較するとライブラリのモジュール化（＝再利用における汎用性)、並列計算への適応性などにおいて不利な点も否めない。ライブラリ管理はコメントや、ドキュメントベースのマニュアルなどに依存する部分が多く、大規模なプロジェクトにおける生産性において非効率的である。特に、手続き型言語において、サブルーチン、関数、メソッド内で多くの状態変数をもつライブラリ群が複雑に関与する計算プログラムにでは、取り得る可能性がある状態すべてについてテストを行うのは困難であろう。また、FORTRAN、C などでもマルチコア CPU やネットワーク接続されたクラスター型計算機の普及にあわせて open-MP、MPI などの技術を駆使した並列プログラミング環境にも対応しつつあるが、並列計算を実装するためには既存のソースコードに対する大幅な追加改変を要求されがちであり、ハードウェア進化に対応するための保守管理に費やされる労力は少なくない。

一方で、純粋関数型言語は一切の状態変数を持たず、入力に一対一対応する出力を得る関数のみを使用して問題を解決する。このような参照透過性を有するため、本質的に関数自体の単体テストに重きを置くことができ、開発効率･保守管理における優位性は高い。特に、強く静的型付けされた関数型原語である Haskell などでは、キャストなどを基本的に許さず不正なデータ処理する関数が原理的に書けないため、ロジックに関する大部分のバグをコンパイラによる型チェックで検出する事が可能である[1]。また、状態変数の値が実行中に変化する「副作用」が存在しないため、同期が必要とされる MPI などと比較してもマルチコア上の並列プログラミングに対する親和性が高いことも期待される。

本研究では、純粋関数型言語として普及している Haskell を量子力学的振動状態計算法の開発に適用し、その応用可能性（モナド概念の有効利用、ウェブサービスとして

のインターフェース構築、並列計算への展開など）を議論する。

【方法論】一般的な振動状態の量子力学計算では

$$\left[-\frac{d^2}{dx^2}+V(x)\right]\Psi(x)=E\Psi(x) \qquad (1)$$

のような、シュレディンガー方程式を解くことになる。多くの手法では有限個数の基底関数系 $\{\phi_i(x)\}(i=1\cdots N)$ を導入し、波動関数を

$$\Psi(x)=\sum_i^N c_i\phi_i(x) \qquad (2)$$

と展開することによって、（1）式の微分方程式を展開係数ベクトル $\mathbf{c}\equiv\left(c_1,c_2,\cdots c_N\right)$ とハミルトニアン行列（但し、$\left(\mathbf{H}\right)_{ij}\equiv\int\phi_i(x)\phi_j(x)dx$）からなる固有値問題

$$\mathbf{H}\cdot\mathbf{c}=E\mathbf{c} \qquad (3)$$

に帰着される。

本研究では、基底関数の選択として、

1）グリッド基底：波動関数を各グリッド点における離散値で表現し、運動エネルギー演算子を差分近似で表現する手法。

2）DVR 基底：各種 DVR 基底を定義する特性多項式の 0 点における離散値で波動関数を表現する手法。

3）EFG 基底：移動最小２乗近似を用いた関数補完によって波動関数を表現する手法。基底関数として局所性と大域性を併せ持っているため、波動関数の局所性と運動エネルギー項における微分評価に優れる。

など種々の基底を採用し、純粋関数型言語 Haskell での実装を試みた。

この際、状態依存計算（＝State モナド）などの関数型言語特有の計算の構造を抽象化する手法を用いて、方法論（＝基底）の変更に依存する計算アルゴリズムを表現し、基底選択に依存しない部分である行列作成･固有値問題の数値解法部分を共通コードで処理可能にした。インターフェース部分に関しては、ウェブサービス公開との親和性を考えて yesod プラットフォーム[2]を採用し、結果の可視化についてはブラウザ上での SVG 表現が可能である d3.js を用いた[3]。この際、グラフ化ライブラリとして d3.js より上の階層で簡便に使用できる rickshaw を利用している[4]。

【参考】

[1]Haskell : An advanced purely-functional programming language) : https://www.haskell.org/

[2] Yesod Web Framework : http://www.yesodweb.com/

[3]3d.js : Data-driven documents : http://d3js.org/

[4] rickshaw : http://code.shutterstock.com/rickshaw/