

水素原子に対するSchrödinger方程式 の数値解法

@dc1394

自己紹介

- Twitter: @dc1394
- C++, C#, F#そしてRubyが好きです(ただしプログラマーではありません)。
- 量子力学の数値計算とかやっています。
- 最も興味のある分野
 - ・第一原理計算
 - ・密度汎関数理論(Density Functional Theory, DFT)
- 第一原理計算やDFTについては、よろしければ拙作のスライドをご覧ください
(<http://www.slideshare.net/dc1394/ss-26378208>)。

概要

- Schrödinger方程式と用いる単位系について
- 3次元のSchrödinger方程式(偏微分方程式)を変数分離し、1次元の常微分方程式に還元
- 境界値の推定
- Boost.odeintを用いた常微分方程式の数値解法
- Brentアルゴリズム(GNU Scientific Library (GSL)を使用)を用いた固有値の探索
- 波動関数の構成と正規化

Schrödinger方程式とは

- 量子力学の(非相対論的な)基礎方程式で、1926年にErwin R. J. A. Schrödingerが提出。
- 単一粒子について、時間に依存しない定常状態でのSchrödinger方程式(最も解きやすい表式)は、

$$\hat{H}\psi(\vec{r}) = E\psi(\vec{r})$$
$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})$$



Erwin R. J. A. Schrödinger
(1887–1961)

Hartree原子単位系

- このスライドでは、Schrödinger方程式の表式を簡潔にするために、Hartree原子単位系を使用する。
- この単位系では、長さの単位はBohr半径 a_0 ($1 [a_0] = 5.29 \times 10^{-11} [m]$), 質量の単位は電子の質量 m_e , 電荷は電気素量 e , エネルギーはHartree ($1 [\text{Hartree}] = 4.36 \times 10^{-18} [J] = 27.2 [eV]$)を用いる。
- この単位系では、Dirac定数 \hbar と、Coulombポテンシャルの比例定数 $1 / (4 \pi \epsilon_0)$ が1となる。
- 単位を表す記号として、atomic unit の省略形である a.u. で表す。

水素原子のSchrödinger方程式

- 最も簡単な水素原子について、定常状態におけるSchrödinger方程式(以下Sch方程式と書く)を以下に示す。

$$-\frac{1}{2}\nabla^2\psi(\vec{r}) - \frac{1}{r}\psi(\vec{r}) = E\psi(\vec{r})$$

電子の運動エネルギーポテンシャル

Coulombポテンシャル

- ここで、 $r = \sqrt{x^2 + y^2 + z^2}$ である。
- この方程式は(少なくとも見かけ上は)単純であり、また解析的に解くことができる(ただし、非常に面倒である)。
- 今回は、この方程式を数値的に解くことを考える。

使用するプログラム言語、ライブラリ等

- プログラム言語はC++11を使用する。
- Boost C++ Librariesを使用する。
- 連立一次方程式のソルバーと、非線形方程式の根を求めるルーチンは、GNU Scientific Library (GSL)に含まれているものを使用する。
- さらに、並列計算のために、Threading Building Blocks (TBB)を使用する。

Sch方程式の変数分離

- 水素原子に対するSch方程式を、以下のように書く。

$$-\frac{1}{2}\Delta\psi_i(\vec{r}) + V(r)\psi_i(\vec{r}) = E\psi_i(\vec{r}), \quad V(r) = -\frac{1}{r}$$

- ここで、極座標におけるラプラシアン Δ は、

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\left[\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right]$$

- であり、 $V(r)$ は球対称であるので、

$$\psi_i(\vec{r}) = r^l L_{nl}(r) Y_{lm}(\theta, \phi)$$

- と変数分離が可能である。ここで、 n, l, m はそれぞれ主量子数、方位量子数、磁気量子数である。

Sch方程式の変数分離

- この変数分離により、以下の二つの微分方程式が得られる。

$$\begin{cases} \frac{d^2 L_{nl}(r)}{dr^2} + \frac{2(l+1)}{r} \frac{dL_{nl}(r)}{dr} = 2[V(r) - E] L_{nl}(r) \\ \hat{l}^2 Y_{lm}(\theta, \phi) = l(l+1) Y_{lm}(\theta, \phi) \end{cases}$$

- なお、この計算は、
https://github.com/dc1394/fukui_sono2/blob/master/fukui_sono2.pdf にアップロードしてあります。
- 第二式の解は、球面調和関数として解析的に得られる。従って、数值的に解くべき方程式は第一式である。

Sch方程式の数値解法上の困難

$$\frac{d^2 L_{nl}(r)}{dr^2} + \frac{2(l+1)}{r} \frac{dL_{nl}(r)}{dr} = 2[V(r) - E] L_{nl}(r)$$

- ここで、この常微分方程式の境界条件は、
- $L_{nl}(0)=\text{有限}$, $L_{nl}(\infty)=0$ である。
- しかし、これらの境界条件は、この常微分方程式を数値的に解く上で、何も言っていないのと同じである。
- さらに、この常微分方程式は固有方程式であり、 E も未知数であるが、 E が固有値以外の値では解が発散する。

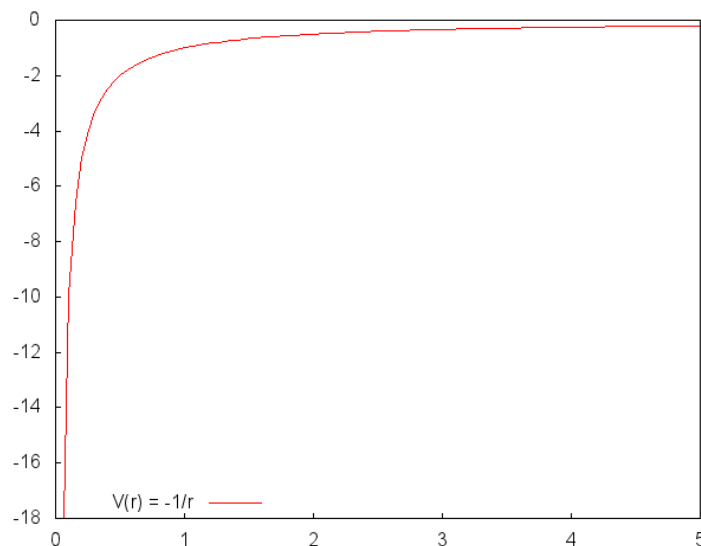
Sch方程式を二点境界値問題にする

$$\frac{d^2 L_{nl}(r)}{dr^2} + \frac{2(l+1)}{r} \frac{dL_{nl}(r)}{dr} = 2[V(r) - E] L_{nl}(r)$$

- 原点は確定特異点であるので、原点から微分方程式を数値的に解くことができない。
- また、コンピュータでは無限大を扱えないので、無限遠点から微分方程式を数値的に解くことも不可能である。
- 従って、原点に十分近い点と、原点から十分離れた点で、 $L_{nl}(r)$ とその微分 $dL_{nl}(r)/dr$ の値を調べ、その二つの点から微分方程式を数値的に解く。
- 最終的に、二点境界値問題に帰着する。

対数メッシュの導入

- ポテンシャル $V(r)$ は原点付近で大きく変化する。



- このような空間の異方性を考慮するために、対数メッシュ $r = \exp(x)$ を導入(変数変換)して微分方程式を変形する。

変数変換した微分方程式

- この変数変換により、 $L_{nl}(r)$ の一階微分と二階微分は以下の式となる。

$$\frac{dL_{nl}(x)}{dr} = e^{-x} \frac{dL_{nl}(x)}{dx}$$
$$\frac{d^2 L_{nl}(r)}{dr^2} = -e^{-2x} \frac{dL_{nl}(x)}{dx} + e^{-2x} \frac{d^2 L_{nl}(x)}{dx^2}$$

- これらを目的の微分方程式に代入することによって、次式が得られる。

$$\frac{d^2 L_{nl}(x)}{dx^2} = -2(l+1) \frac{dL_{nl}(x)}{dx} + 2e^{2x} [V(x) - E] L_{nl}(x)$$

連立一階常微分方程式に書き直す

- 二階の常微分方程式を直接、数値的に解くことは難しい。
- 従って、二階の常微分方程式を、二元連立一階常微分方程式に書き直す。すると、次の二つの式が得られる。

$$\begin{cases} \frac{dL_{nl}(x)}{dx} = M(x) \\ \frac{dM(x)}{dx} = -(2l+1)M(x) + 2e^{2x}[V(x) - E]L_{nl}(x) \end{cases}$$

- これらの式が、数値的に解くべき方程式である。
- 次に、 $L_{nl}(x)$ と $M(x)$ の境界値について考える。

$V(r)$ と $L_{nl}(r)$ を級数展開する

- まずは原点に十分近い点で、関数 $V(r)$ と関数 $L_{nl}(r)$ がどう振る舞うか調べてみる(Frobeniusの方法を用いる)。

- 関数 $V(r)$ と関数 $L_{nl}(r)$ は、以下のように級数展開できるはずである。

$$V(r) = \sum_{n=0}^{\infty} a_n r^n \quad L_{nl}(r) = \sum_{m=0}^{\infty} b_m r^m$$

- 従って、 $L_{nl}(r)$ の一階微分と二階微分は以下となる。

$$\frac{dL_{nl}(r)}{dr} = \sum_{m=1}^{\infty} m b_m r^{m-1} \quad \frac{d^2 L_{nl}(r)}{dr^2} = \sum_{m=2}^{\infty} m(m-1) b_m r^{m-2}$$

級数展開した式を代入する

$$\frac{d^2 L_{nl}(r)}{dr^2} + \frac{2(l+1)}{r} \frac{dL_{nl}(r)}{dr} = 2[V(r) - E] L_{nl}(r)$$

□ 前ページの結果を上式に代入すると、

□ 左辺=

$$\sum_{m=0}^{\infty} [m(m-1) + 2(l+1)m] b_m r^{m-2}$$

□ 右辺=

$$2 \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} a_m b_m r^{n+m} - 2E \sum_{m=0}^{\infty} b_m r^m$$

□ が得られる。

$L_{nl}(r)$ の級数展開の係数を求める

- 左辺と右辺の r のべき乗の係数を比較することによって、 $L_{nl}(r)$ の級数展開の係数 $b_0 \sim b_4$ は以下のように得られる。

$$b_0 = \text{arbitrary}$$

$$b_1 = 0$$

$$b_2 = \frac{(a_0 - E) b_0}{2l + 3}$$

$$b_3 = \frac{a_1 b_0}{3l + 6}$$

$$b_4 = \frac{a_0 b_2 + a_2 b_0 - E b_2}{4l + 10}$$

- ここで、 $V(r)$ の級数展開の係数 $a_0 \sim a_2$ (と E)は、未だ未知数であるので、別に求める必要がある。

原点付近の $L_{nl}(x)$ と $M(x)$ の近似値

- $V(r)$ の級数展開の係数 $a_0 \sim a_2$ は、以下の連立一次方程式から求められる。

$$\begin{pmatrix} 1 & r_0 & r_0^2 \\ 1 & r_1 & r_1^2 \\ 1 & r_2 & r_2^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} V(r_0) \\ V(r_1) \\ V(r_2) \end{pmatrix}$$

- この連立一次方程式の解は、GSLの連立一次方程式のソルバーを用いれば、簡単に得られる。
- その解を用いて、以下のように、原点付近($x = x_0$)での $L_{nl}(x_0)$ と $M(x_0)$ の近似値が求められる。

$$L_{nl}(x_0) = \{ \{ [(b_4 e^{x_0} + b_3) e^{x_0}] + b_2 \} e^{x_0} + b_1 \} e^{x_0} + b_0$$

$$M(x_0) = \{ [(4b_4 e^{x_0} + 3b_3) e^{x_0} + 2b_2] e^{x_0} + b_1 \} e^{x_0}$$

原点から十分離れた点での波動関数の振る舞い

- 波動関数を以下のように書くと、

$$\psi_i(\vec{r}) = \frac{P_{nl}(r)}{r} Y_{lm}(\theta, \phi)$$

- 次式が成り立つ。

$$\left[-\frac{1}{2} \frac{d^2}{dr^2} + V(r) + \frac{l(l+1)}{r^2} \right] P_{nl}(r) = E P_{nl}(r)$$

- 左辺中括弧の中の第二項と第三項の寄与は、原点から十分離れたところでは無視できる。従って、

$$-\frac{1}{2} \frac{d^2 P_{nl}(r)}{dr^2} \approx E P_{nl}(r)$$

- となる。この微分方程式の解析解は容易に分かり、

$$P_{nl}(r) = \exp \left[- \left(\sqrt{-2E} \right) r \right]$$

- である。

原点から十分離れた点での $L_{nl}(x)$ と $M(x)$ の近似値

□ ここで、

$$L_{nl}(r) = \frac{P_{nl}(r)}{r^{l+1}}$$

□ である。

□ 従って、上式に前ページの結果を代入すると、原点から十分離れた点($x = x_{\max}$)での $L_{nl}(x_{\max})$ と $M(x_{\max})$ の近似値は次式となる。

$$L_{nl}(x_{\max}) = \frac{\exp[-(\sqrt{-2E})e^{x_{\max}}]}{e^{(l+1)x_{\max}}}$$

$$M(x_{\max}) = -L_{nl}(x_{\max}) \left\{ \sqrt{-2E} + \frac{(l+1)}{e^{x_{\max}}} \right\}$$

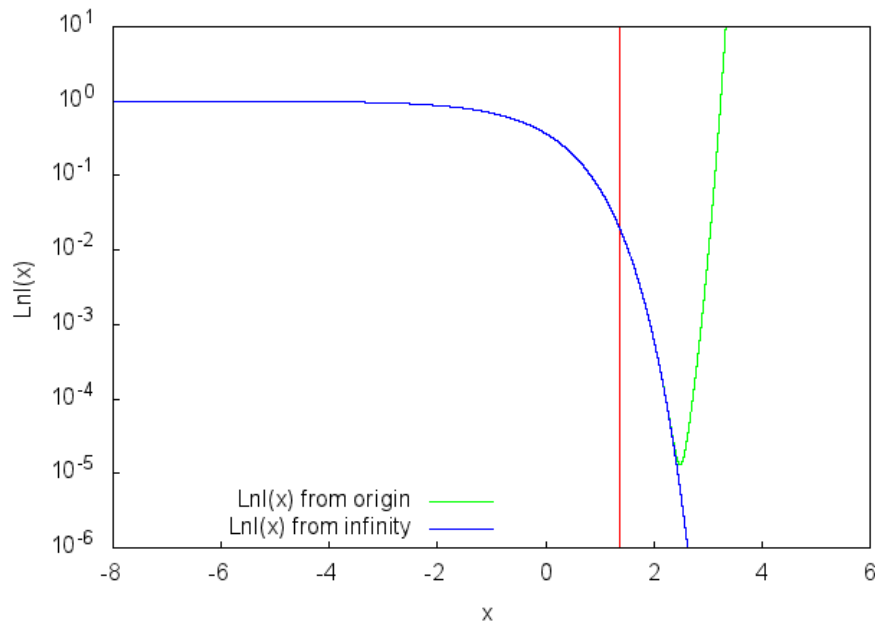
常微分方程式の解法

- これらの近似値を初期値として用いて、あるエネルギー E を仮定し、原点に十分近い点と、原点から十分離れた点から、常微分方程式を数値的に解いていく。
- 常微分方程式のソルバーには、Boostに含まれるBoost.odeintを用いる。
- ソルバーのアルゴリズムは、Adams-Bashforth-Moulton法(予測子修正子法)、Burilrsh-Stoer法(補外法)、Controlled Runge-Kutta法(誤差とステップ幅を制御したRunge-Kutta法)のいずれかを使う(私のコードでは、インプットファイルでアルゴリズムを指定するようにしている)。

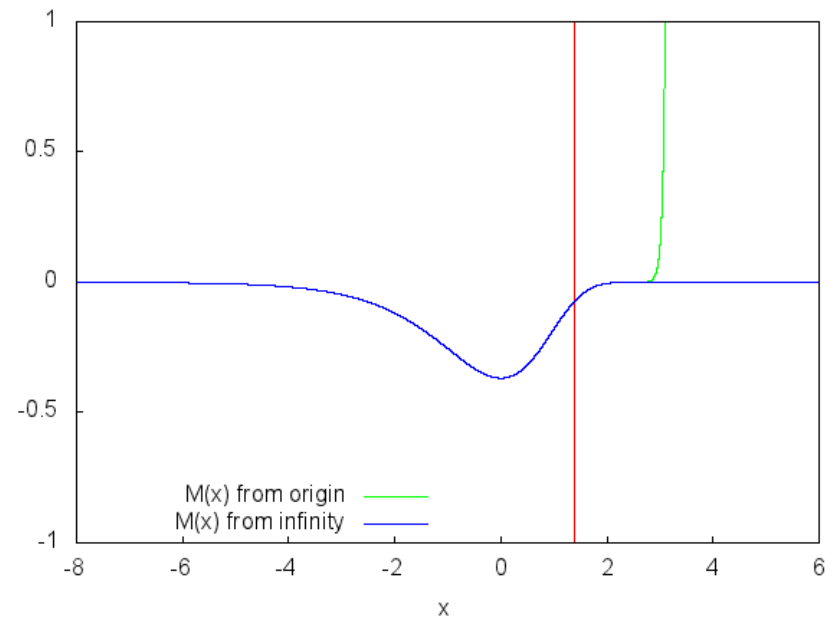
$L_{nl}(x)$ と $M(x)$ のグラフ

- $l = 0$, $E = -0.5$ (Hartree)に対して、 $x = -8.0$ (緑線)及び $x = 6.0$ (青線)から解くと、以下のようなになる。

$L_{nl}(x)$ for $E = -0.5$ (Hartree)

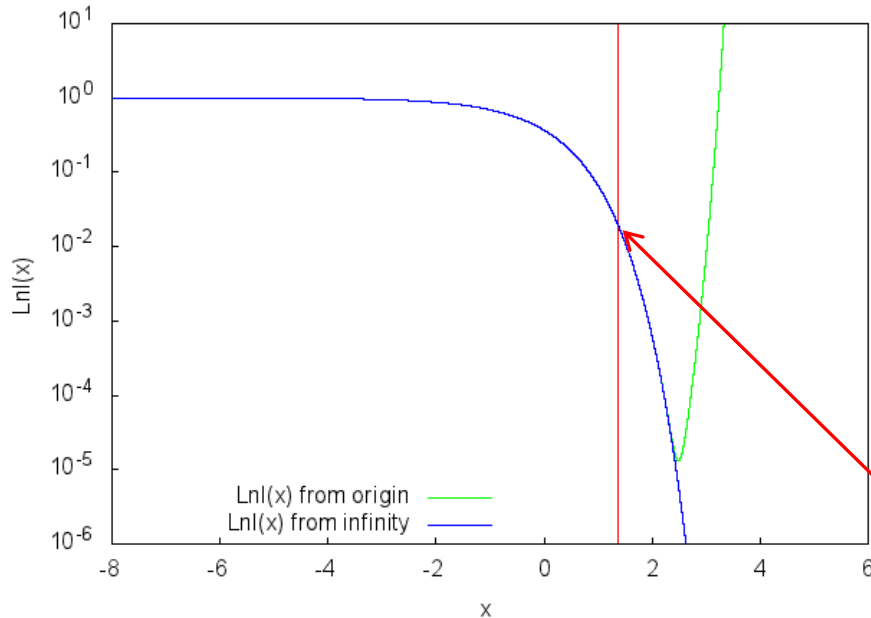


$M(x)$ for $E = -0.5$ (Hartree)

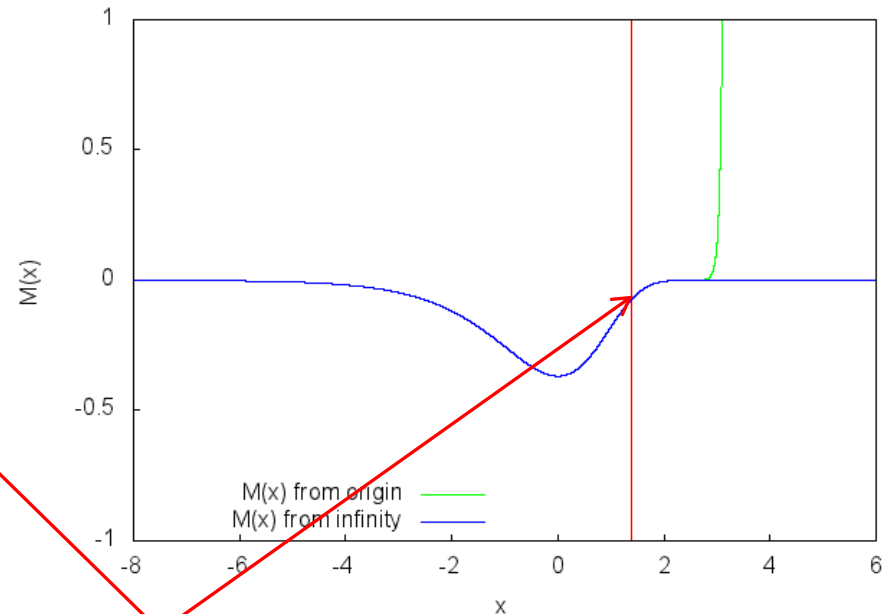


マッチングポイント

$\text{Lnl}(x)$ for $E = -0.5$ (Hartee)



$M(x)$ for $E = -0.5$ (Hartee)



マッチングポイント

- 二つの $\text{L}_{nl}(x)$ あるいは $M(x)$ を比較する点を、マッチングポイントと呼ぶ。

関数 $\Delta D(E)$ を定義する

- もし選んだ E が固有値であるならば、次式が成り立つ。

$$\frac{L_{nl,O}(x_{MP})}{L_{nl,I}(x_{MP})} = \frac{M_O(x_{MP})}{M_I(x_{MP})}$$

- ここで、以下の関数 $\Delta D(E)$ を定義する。

$$\Delta D(E) = M_O(x_{MP}) - \alpha M_I(x_{MP}), \quad \alpha = \frac{L_{nl,O}(x_{MP})}{L_{nl,I}(x_{MP})}$$

- この関数 $\Delta D(E)$ がゼロになる E (つまり非線形方程式 $\Delta D(E) = 0$ の根) が、固有値である。

固有値Eを探索する

- $\Delta D(E)$ は固有値の前後で符号が変化する。
- 従って、固有値を探索するアルゴリズムは以下のようになる。
 - (1) Eをスキャンし、 $\Delta D(E)$ の符号が変化する領域を探す。
 - (2) 符号が変化する領域が見つかったら、その領域内で、Brent法(これはGSLに含まれるルーチンを使う)を用いて、 $\Delta D(E) = 0$ の根を求める。なお、Brent法は、非線形方程式の根を非常に効率的に求めるアルゴリズムである。
 - (3) 求まった根が目的の固有値Eである。

波動関数の構成と正規化

- 見つかった固有値Eに対して、以下の順序で波動関数を求める。

- (1) $L_{nl,O}(x)$ と $L_{nl,I}(x)$ を接合して $L_{nl}(x)$ を構成する。

$$L_{nl}(x) = \begin{cases} L_{nl,O}(x) & (x \leq x_{MP}) \\ \frac{L_{nl,O}(x_{MP})}{L_{nl,I}(x_{MP})} L_{nl,I}(x) & (x > x_{MP}) \end{cases}$$

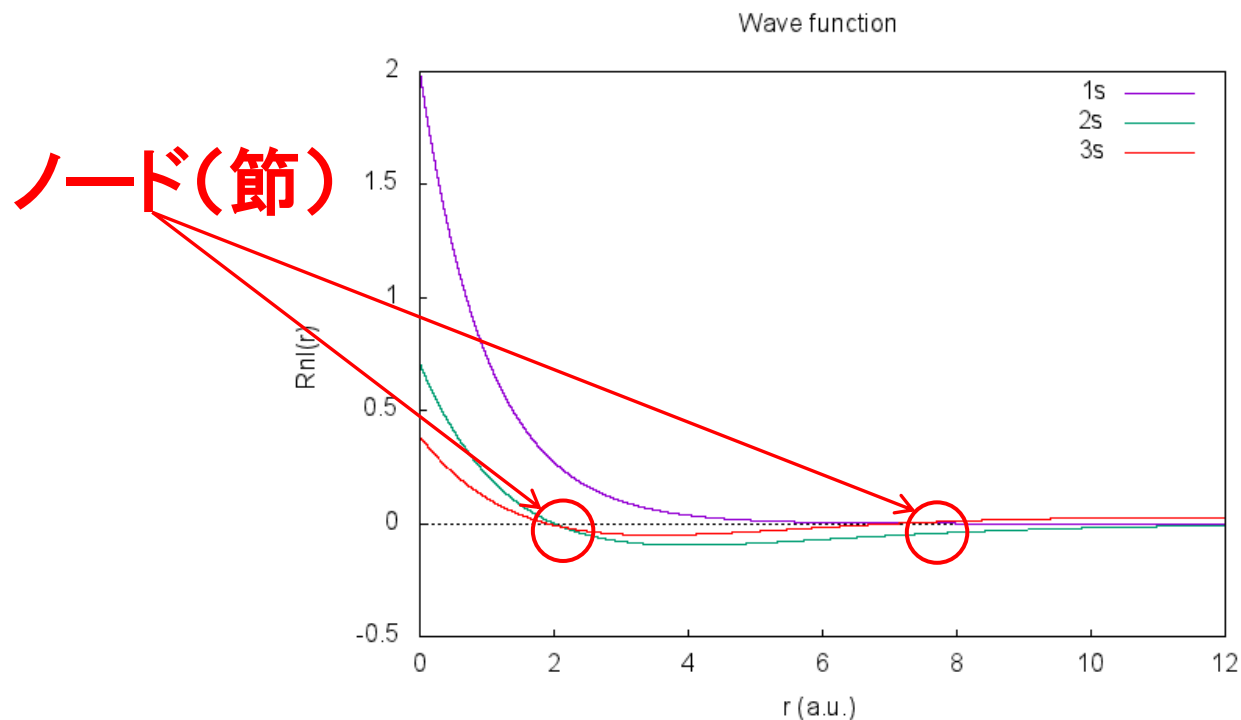
- (2) 以下の式により、 $L_{nl}(x)$ を正規化する。

$$N = \int_{x_{\min}}^{x_{\max}} [P_{nl}(x)]^2 e^x dx$$

$$L_{nl}(x) := \frac{1}{\sqrt{N}} L_{nl}(x)$$

波動関数のノード数

- それぞれの波動関数のノード数は $n - l - 1$ となる。
- 重複して解を求めてしまわないように、ノード数が適切なものになっているか調べる必要がある。



最終的に得られる波動関数

- 最終的に得られる波動関数は(変数をxからrに戻した)、

$$R_{nl}(r) = r^l L_{nl}(r)$$

- と、及びそれにrを乗じた形

$$P_{nl}(r) = r R_{nl}(r)$$

- である。これらをrのメッシュの値と共にファイルに出力する。

厳密な固有値と計算で求めた固有値の比較

- 水素原子の場合、厳密な固有値は解析的に求められ、

$$E_{exact} = -\frac{1}{2n^2}$$

- となる。上式より、1s軌道の場合、厳密な固有値は-0.5 (Hartree)である。
- 私のコードでの計算値は(インプットファイルで与えるパラメータにもよるが)、-0.499999985 (Hartree)であり、非常に良く一致している。

ポテンシャルエネルギーと運動エネルギー

- ポテンシャルエネルギーの期待値 $\langle V \rangle$ は、次式で与えられる。

$$\langle V \rangle = \int_{x_{\min}}^{x_{\max}} [P_{nl}(x)]^2 dx$$

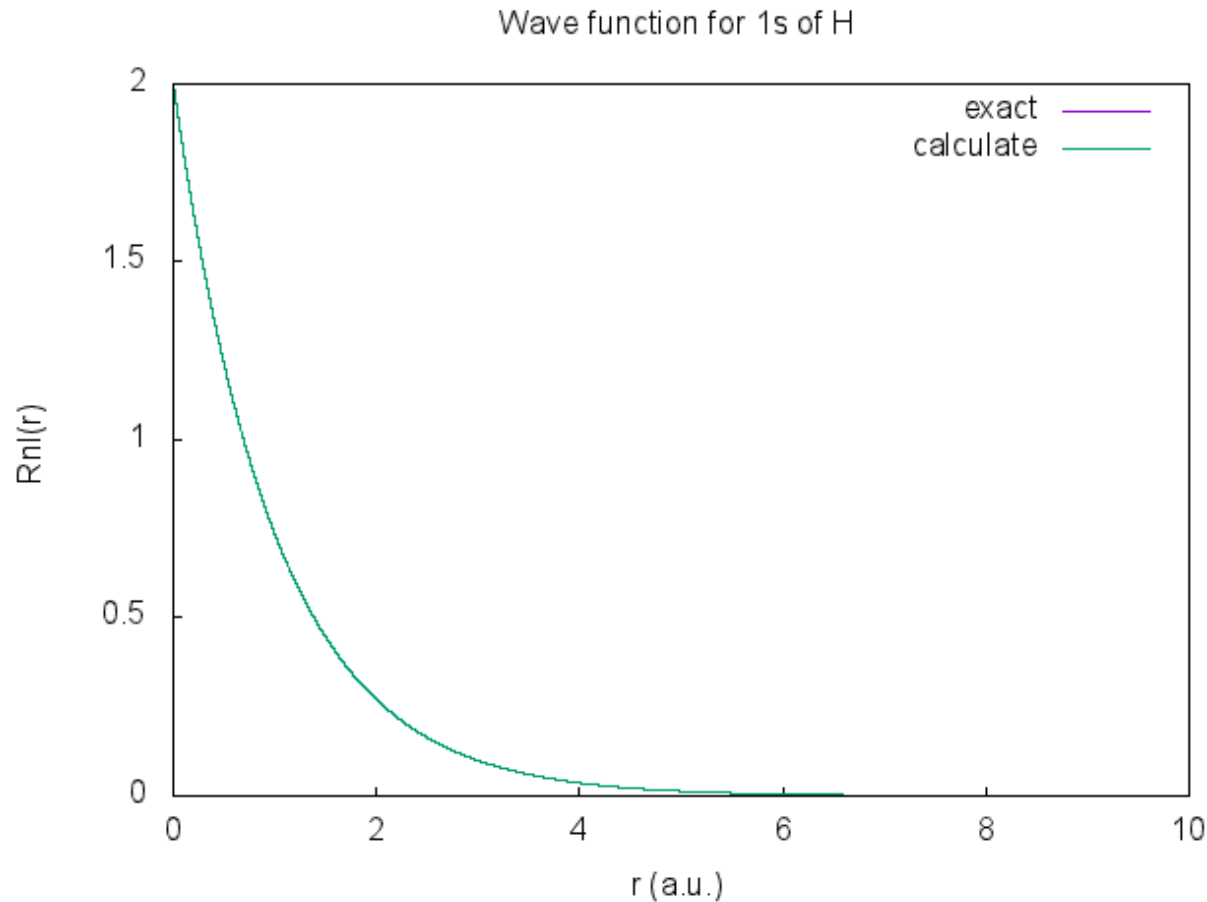
- 水素原子の1s軌道の場合、厳密なポテンシャルエネルギーの期待値は -1.0 (Hartree)であるが、私のコードによる計算値は、 -0.99999997 (Hartree)である。

- 運動エネルギーの期待値 $\langle T \rangle$ は、virial定理から、

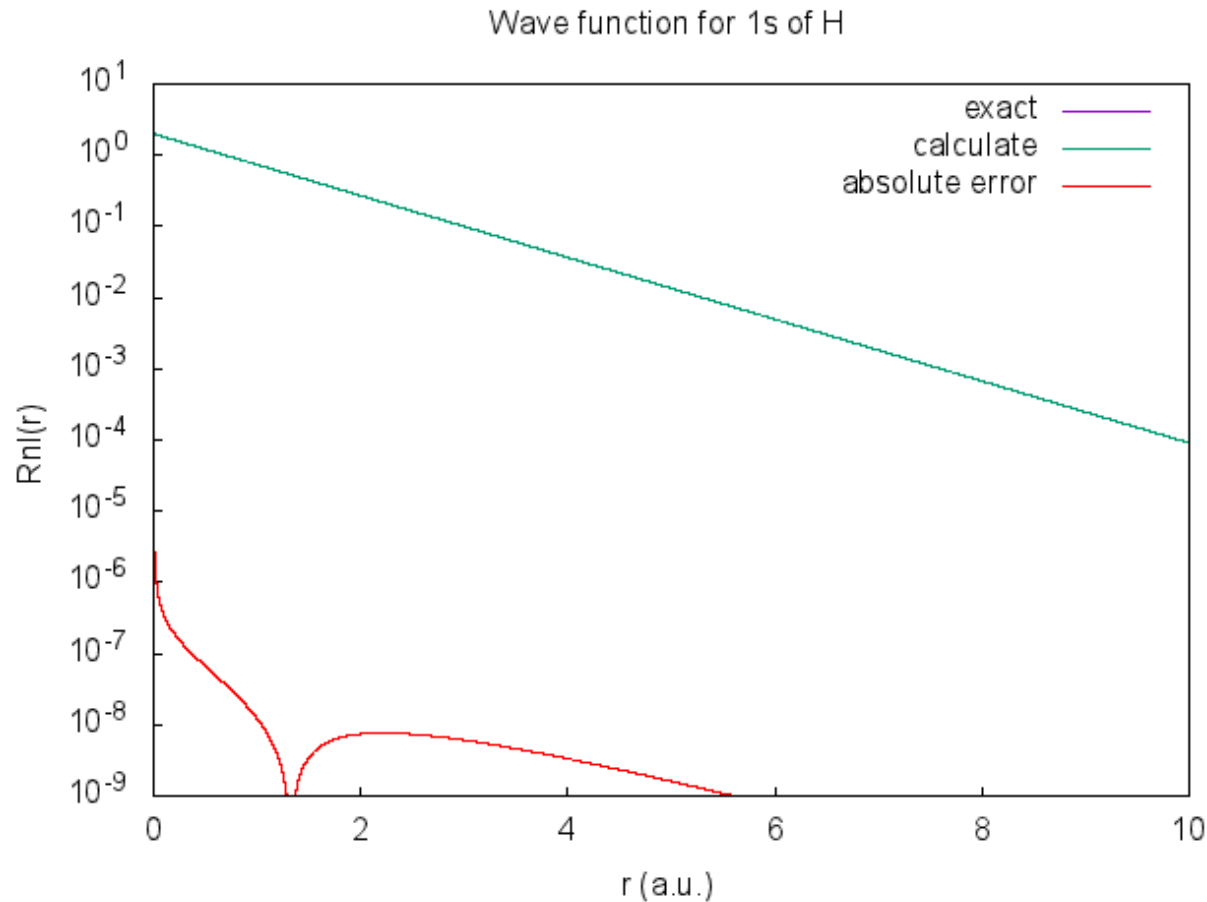
$$\langle T \rangle = -\frac{1}{2} \langle V \rangle$$

- であるので、ポテンシャルエネルギーから簡単に求められる。

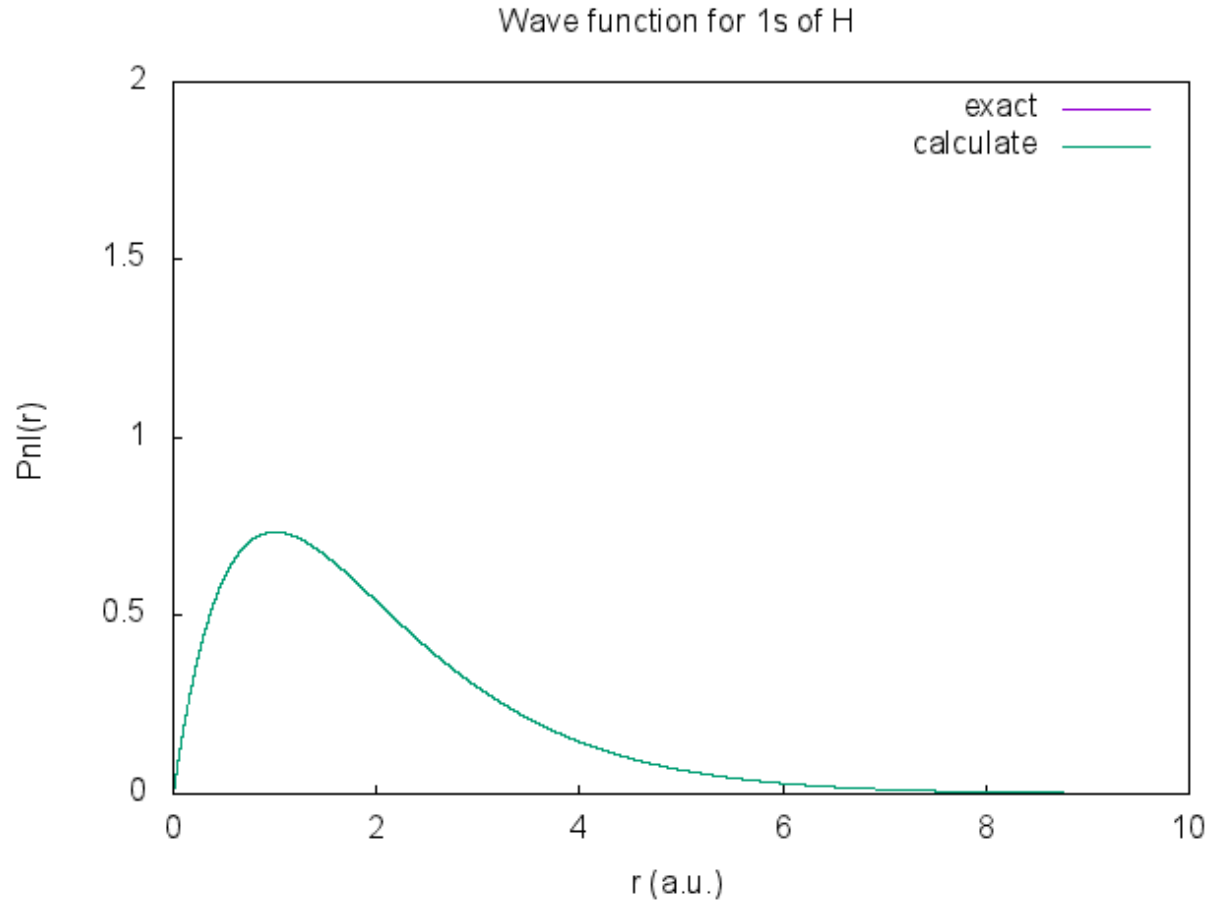
厳密な $R_{nl}(r)$ と、計算で求めた $R_{nl}(r)$ を比較したプロット(H原子の1s軌道)



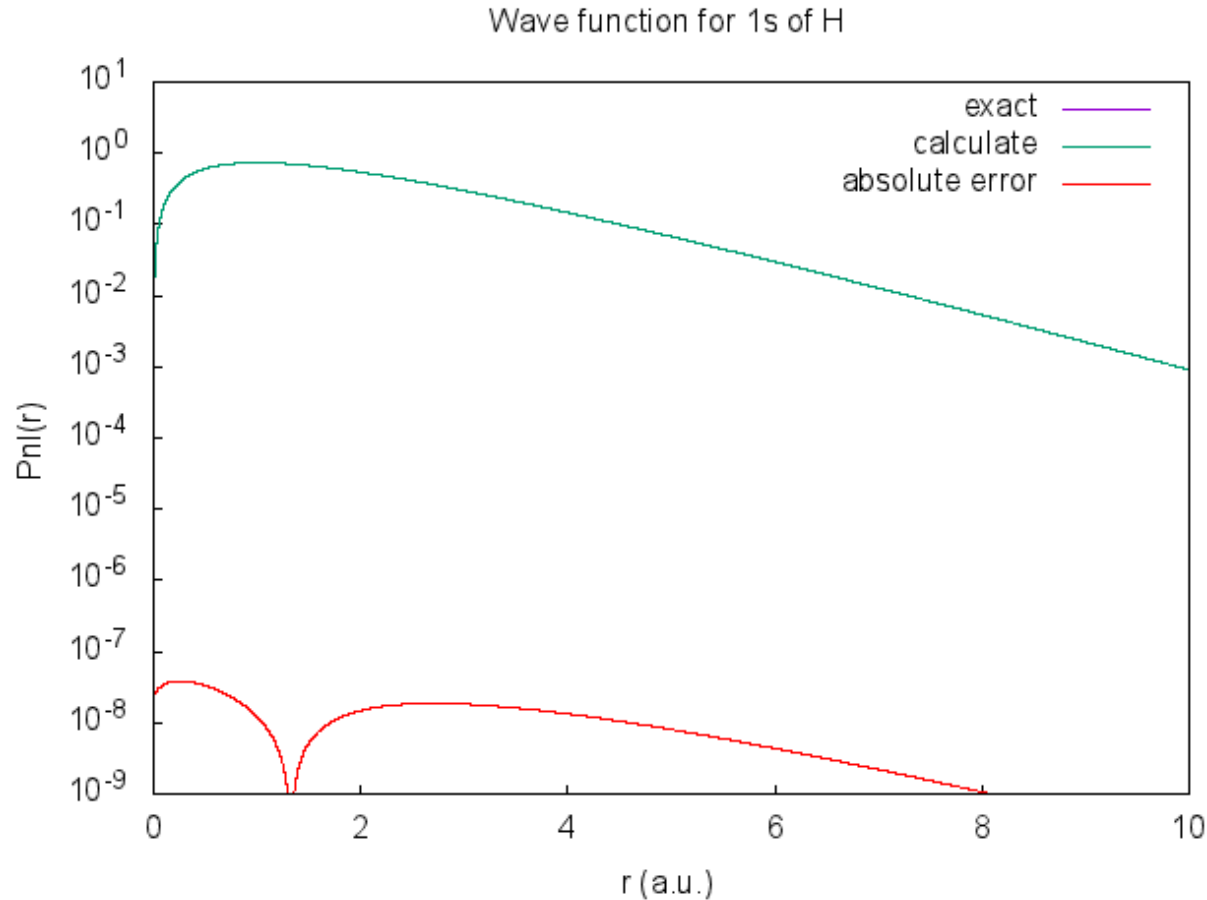
厳密な $R_{nl}(r)$ と、計算で求めた $R_{nl}(r)$ を比較したプロット(y軸対数目盛)



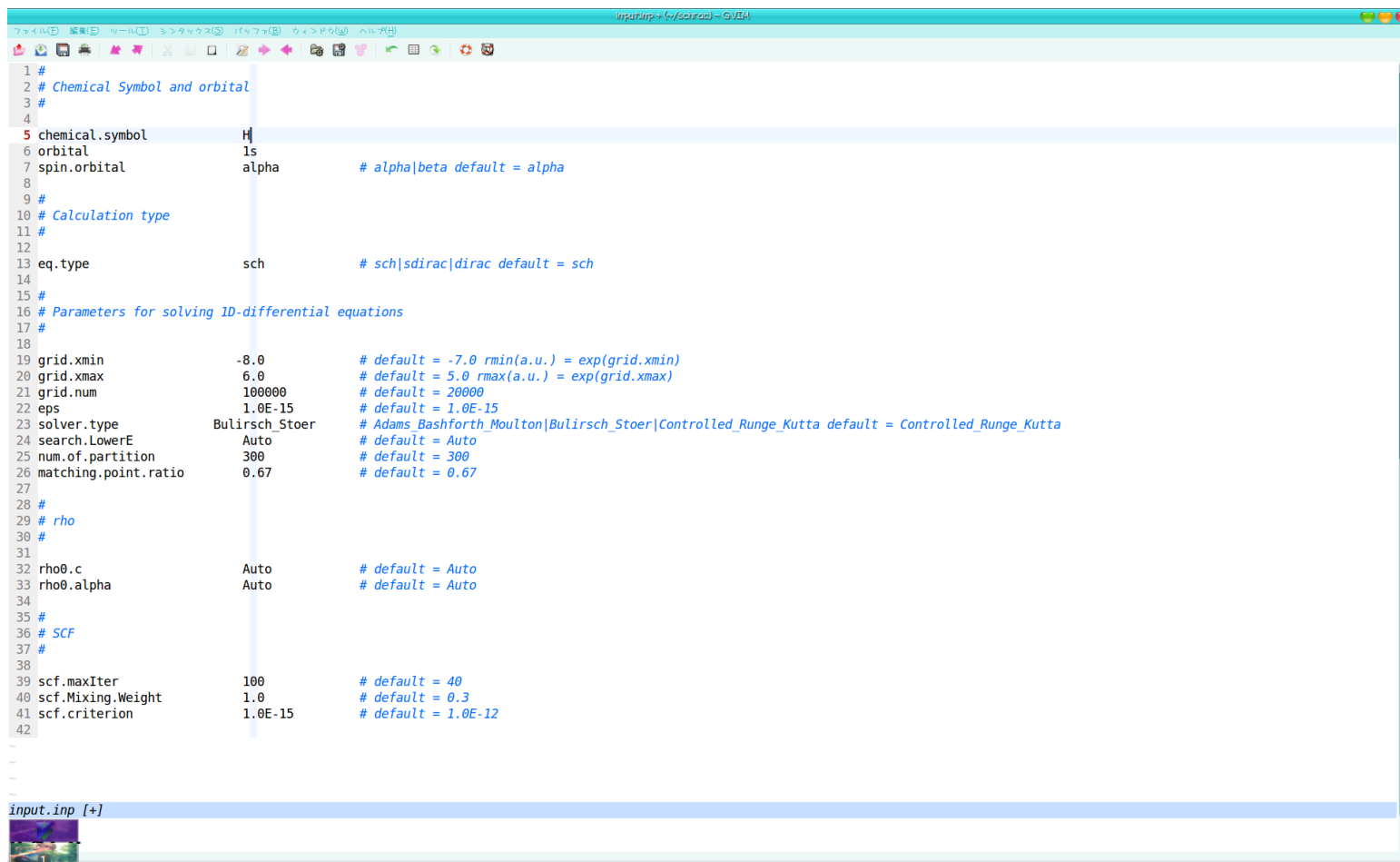
厳密な $P_{nl}(r)$ と、計算で求めた $P_{nl}(r)$ を比較したプロット(H原子の1s軌道)



厳密な $P_{nl}(r)$ と、計算で求めた $P_{nl}(r)$ を比較したプロット(y軸対数目盛)



インプットファイル



```
input.inp - Gvim
1 #
2 # Chemical Symbol and orbital
3 #
4
5 chemical.symbol      H
6 orbital              1s
7 spin.orbital         alpha      # alpha|beta default = alpha
8
9 #
10 # Calculation type
11 #
12
13 eq.type              sch        # sch|sdirac|dirac default = sch
14
15 #
16 # Parameters for solving 1D-differential equations
17 #
18
19 grid.xmin             -8.0      # default = -7.0 rmin(a.u.) = exp(grid.xmin)
20 grid.xmax             6.0       # default = 5.0 rmax(a.u.) = exp(grid.xmax)
21 grid.num              100000    # default = 20000
22 eps                  1.0E-15    # default = 1.0E-15
23 solver.type           Bulirsch_Stoer # Adams_Bashforth_Moulton|Bulirsch_Stoer|Controlled_Runge_Kutta default = Controlled_Runge_Kutta
24 search.LowerE         Auto      # default = Auto
25 num.of.partition      300       # default = 300
26 matching.point.ratio  0.67      # default = 0.67
27
28 #
29 # rho
30 #
31
32 rho0.c               Auto      # default = Auto
33 rho0.alpha            Auto      # default = Auto
34
35 #
36 # SCF
37 #
38
39 scf.maxIter           100       # default = 40
40 scf.Mixing.Weight      1.0      # default = 0.3
41 scf.criterion          1.0E-15  # default = 1.0E-12
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

input.inp [+]

実行画面

```
dc1394@dc1394-Notebook-PC ~/schrac $ ./schrac -I input.inp -T 1
i = 1, D = 0.034177600716728, node = 0 (OK)
i = 2, D = 0.022634644714076, node = 0 (OK)
i = 3, D = 0.011242476681066, node = 0 (OK)
i = 4, D = 0.000000151249277, node = 0 (OK)
i = 5, D = -0.011093272653194, E = -0.498333333333333, node = 0 (OK)
i = 6, D = -0.000000001012951, E = -0.499999977276436, node = 0 (OK)
i = 7, D = 0.000000000000045, E = -0.499999977427608, node = 0 (OK)
i = 8, D = -0.000000000000003, E = -0.499999977427602, node = 0 (OK)
E(Kinetic Energy)      = 0.499999853697015
E(Potential Energy)    = -0.999999707394030
E(Eigenvalue)          = -0.499999977427602
E(Total Energy)        = -0.499999977427602

wavefunction_H_1s.csvに波動関数を書き込みました。
コマンドラインオプション解析処理 elapsed time = 0.4451 (msec)
初期化処理 elapsed time = 8.9673 (msec)
微分方程式の積分と固有値探索処理及び正規化処理 elapsed time = 1214.6309 (msec)
エネルギー出力処理 elapsed time = 2.5569 (msec)
ファイル書き込み処理 elapsed time = 140.5684 (msec)
Total elapsed time = 1367.1686 (msec)
Used Memory Size: 13976(kB)
終了するには何かキーを押してください...
```



Terminal

He原子に対する実行画面(参考)

```
dc1394@dc1394-Notebook-PC ~/schrac $ ./schrac -I input.inp -T 1
SCF = 1, NormRD = 36.473672058001064, Energy = -3.071777963678596
SCF = 2, NormRD = 3.253866567885526, Energy = -2.752397411042186
SCF = 3, NormRD = 0.269082031037238, Energy = -2.893353103592523
SCF = 4, NormRD = 0.023423543308962, Energy = -2.850240581354331
SCF = 5, NormRD = 0.002016177632398, Energy = -2.862744341426661
SCF = 6, NormRD = 0.000174200355797, Energy = -2.859056652035135
SCF = 7, NormRD = 0.000015035298330, Energy = -2.860138981450108
SCF = 8, NormRD = 0.000001298112867, Energy = -2.859820865645627
SCF = 9, NormRD = 0.000000112065713, Energy = -2.859914326242244
SCF = 10, NormRD = 0.000000009674856, Energy = -2.859886864679734
SCF = 11, NormRD = 0.000000000835244, Energy = -2.859894933432290
SCF = 12, NormRD = 0.000000000072108, Energy = -2.859892562645613
SCF = 13, NormRD = 0.000000000006225, Energy = -2.859893259242863
SCF = 14, NormRD = 0.000000000000538, Energy = -2.859893054545039
SCF = 15, NormRD = 0.000000000000046, Energy = -2.859893114688969
SCF = 16, NormRD = 0.000000000000004, Energy = -2.859893097022971
SCF = 17, NormRD = 0.000000000000000, Energy = -2.859893102213666
E(Kinetic Energy)      = 1.687231374889430
E(Coulomb Energy)      = -3.374462749778861
E(Hartree Energy)      = -1.027521869779237
E(Eigenvalue)          = -0.916185616217215
E(Total Energy)        = -2.859893102213666

wavefunction_He_1s.csvに波動関数を書き込みました。
コマンドラインオプション解析処理 elapsed time = 0.2231 (msec)
初期化処理 elapsed time = 54.9874 (msec)
微分方程式の積分と固有値探索処理及び正規化処理 elapsed time = 272617.1897 (msec)
エネルギー出力処理 elapsed time = 2.6165 (msec)
ファイル書き込み処理 elapsed time = 121.8896 (msec)
Total elapsed time = 272796.9063 (msec)
Used Memory Size: 38748(kB)
終了するには何かキーを押してください...
```

ソースコードへのリンク

- このプログラムのソースコードは、GitHub上で公開しています
- <https://github.com/dc1394/schrac>
- ライセンスは修正BSDライセンスとします。

まとめ

- Sch方程式を、原点付近と原点から十分遠い点から数値的に解いた。
- それぞれの解を接合することにより、固有値及び波動関数を得た。
- 計算によって得られた波動関数から、運動エネルギー及びポテンシャルエネルギーを計算した。
- 計算で求めた固有値及び波動関数のいずれも、解析的に求められる値とほとんど完全に一致していた。