東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

MLPhYs 学術変革領域研究(A) 学習物理学の創成
Foundation of "Machine Learning Physics"

格子上の場の理論ハンズオン
part1 Julia入門

東京大学情報基盤センター学際情報科学研究部門　永井佑紀

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

自己紹介

永井佑紀

経歴

2005: 北海道大学工学部応用物理学科卒業
2010: 博士（理学）東京大学　（指導教官：加藤雄介氏）
2010-2019 日本原子力研究開発機構　常勤研究員
2016-2017 米国マサチューセッツ工科大学客員研究員@ボストン
2018-2023　理研革新知能統合研究センター (AIP)客員研究員
2019-2024 日本原子力研究開発機構　副主任研究員
2024- 東京大学情報基盤センター学際情報科学研究部門 准教授

物性理論（主に超伝導）

機械学習と物理学

(最近の)専門分野

物性理論x機械学習　材料科学x機械学習　格子QCDx機械学習

精度が保証された機械学習：自己学習ハイブリッドモンテカルロ法の開発etc

著書など

2008

中学生のお子さんを持つお母さん向けの物理の本

自由国民社

2019

機械学習の物理学への応用の書籍

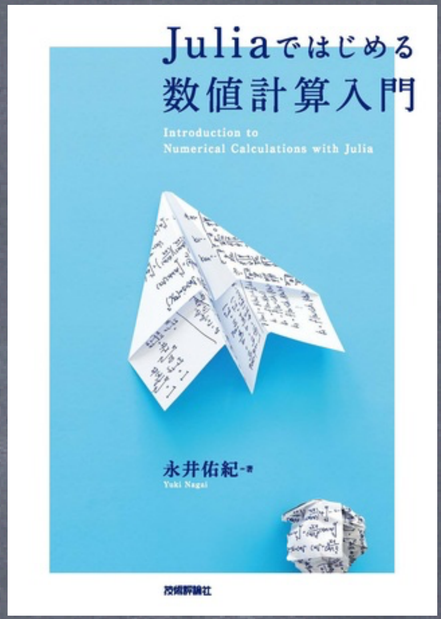
朝倉書店

2022

Julia言語の書籍(2022年6月)

講談社

2024



Julia言語の書籍(2024年6月頃)

技術評論社

Julia言語:Pythonのように書きやすく、CやFortranのように高速な言語

お手軽に本格的なプログラミングができるので最近推している

数値計算とは？

数値計算　人間ができないor面倒or大変なことを計算機にやらせる

物理系の学科の学部で習うこと

力学、熱力学、統計力学、電磁気学、量子力学etc...

紙と鉛筆で学習：講義に数値計算は出てこない

何故？

容易に手で解ける問題しか扱っていないから

手で解ける領域

残りは解けない

物理で現れる計算

2つの物体同士に働く万有引力

$$F = -G\frac{Mm}{r^2}$$

r:2つの物体間の距離

運動方程式を立てると

$$m\vec{a} = \vec{F} \qquad \vec{F} = -\frac{GMm}{r^3}\vec{r}$$

2本の微分方程式となる

$$\frac{d\vec{v}}{dt} = -\frac{GM}{r^3}\vec{r} \qquad \frac{d\vec{r}}{dt} = \vec{v}$$

これを解けば2つの物体の運動がわかる

2つなら手で色々やることで性質がわかる

N個の物体がある場合は？

物理で現れる計算

N個の物体同士に働く万有引力

$$F_i = -G\sum_j \frac{m_i m_j}{|\vec{r}_i - \vec{r}_j|^2}$$

r_i: i番目の物体の位置

運動方程式を立てると

$$m\vec{a}_i = \vec{F}_i \qquad \vec{F}_i = -G\sum_j m_i m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^2}$$

2N本の微分方程式となる

$$\frac{d\vec{v}_i}{dt} = -G\sum_j m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^2} \qquad \frac{d\vec{r}_i}{dt} = \vec{v}_i$$

これを解けばN個の物体の運動がわかる

->手では解けない

-> 自然界の振る舞いを調べるには、紙と鉛筆だけでは足りない

数値計算をする、とはどういうことか

手で解ける領域

残りは解けない

手で解けない面白い問題が沢山ある

計算機は道具である

物理屋は数学と計算機を道具とする

数学科ほど数学を深めない

情報系学科ほど計算機を深めない

どちらも例外あり

現実世界に存在する面白いものを対象とする

どんな方法であれ解ければ良い

物理学者が数学者を刺激する場合もあり

(多くの)物理屋にとって数値計算とは

物理の問題に注力したい

物理以外の問題に手間をかけ過ぎたくない

自動車を運転して遠くに行きたいのであって、そのために一から車の部品を組み立てたいわけじゃない

現実世界に存在する面白いものを対象とする

どんな方法であれ解ければ良い

一方、

目的地に到達するためにはカスタマイズした移動手段が必要な場合もある

カスタマイズが楽しくなってくる場合もある

計算機科学者との違い：見据えているのは目的地

自由度が高い、書きやすい、そして速いプログラミング言語がいい

Julia言語??

数値計算の種類

1. 理論はすでにあり、適応する対象が無数にあるケース　**Fortran,C++**

例：固体物理における「第一原理計算」

2.どのような理論を使うか試行錯誤が必要なケース　**Fortran,C++,Python**

いっぱいある

グリーン関数による摂動論

ハミルトニアンの数値的対角化

3. 機械学習が絡むケース　**Python**

最先端の機械学習の手法は個人で実装するには大変すぎる -> Pythonライブラリの使用

4. 上記の様々な組み合わせ

Juliaは2,3,4で有用

Juliaはどんなプログラミング言語か

数値計算のためのプログラミング言語

数値計算をする場合に、どのプログラミング言語を選べば良いのか

原理的には、どんな言語を選んでも数値計算は可能

物理の問題に注力したい　　どんなプログラミング言語が良いか

物理の数値計算では線形代数を多用する

特殊関数も使う

積分も微分もする

-> 数値計算のできる「電卓」が欲しい！

使うのが簡単がいい

インストールとかコンパイルとかに時間を溶かしたくない

バグとりが楽ならなお良い

Julia言語が最適

Juliaはどんな言語か

1次元シュレーディンガー方程式

$$\hat{H}\psi_n(x) = E_n\psi_n(x) \qquad \hat{H} = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2} + V(x,y,z)$$

差分化して解いて、V=0の解析解と比較してみる

```
function make_H(N,L,V)
    Δx = L/(N+1)
    H = zeros(Float64,N,N)
    for i=1:N
        x = i*Δx
        H[i,i] = V(x)
        j = i+1
        dij = -1/Δx^2
        if 1 ≤ j ≤ N
            H[i,j] += dij
        end

        j=i
        dij = 2/Δx^2
        if 1 ≤ j ≤ N
            H[i,j] += dij
        end

        j=i-1
        dij = -1/Δx^2
        if 1 ≤ j ≤ N
            H[i,j] += dij
        end

    end
    return H
end
```

```
using LinearAlgebra

using Plots
function test()
    V(x) = 0
    N = 1000
    L = 1
    H = make_H(N,L,V)
    e,v = eigen(H)
    e0 = zeros(Float64,N)
    for n=1:N
        e0[n]=n^2*π^2/L^2
    end
    plot(1:N,[e,e0],labels=["Numerical result" "Analytical result"],xlabel="n",ylabel="energy")
    savefig("eigen.png")
    println(e0[1],"\t",e[1])
end
test()
```

本当にこれだけのコード
他に”おまじない”はいらない

Juliaはどんな言語か

1次元シュレーディンガー方程式

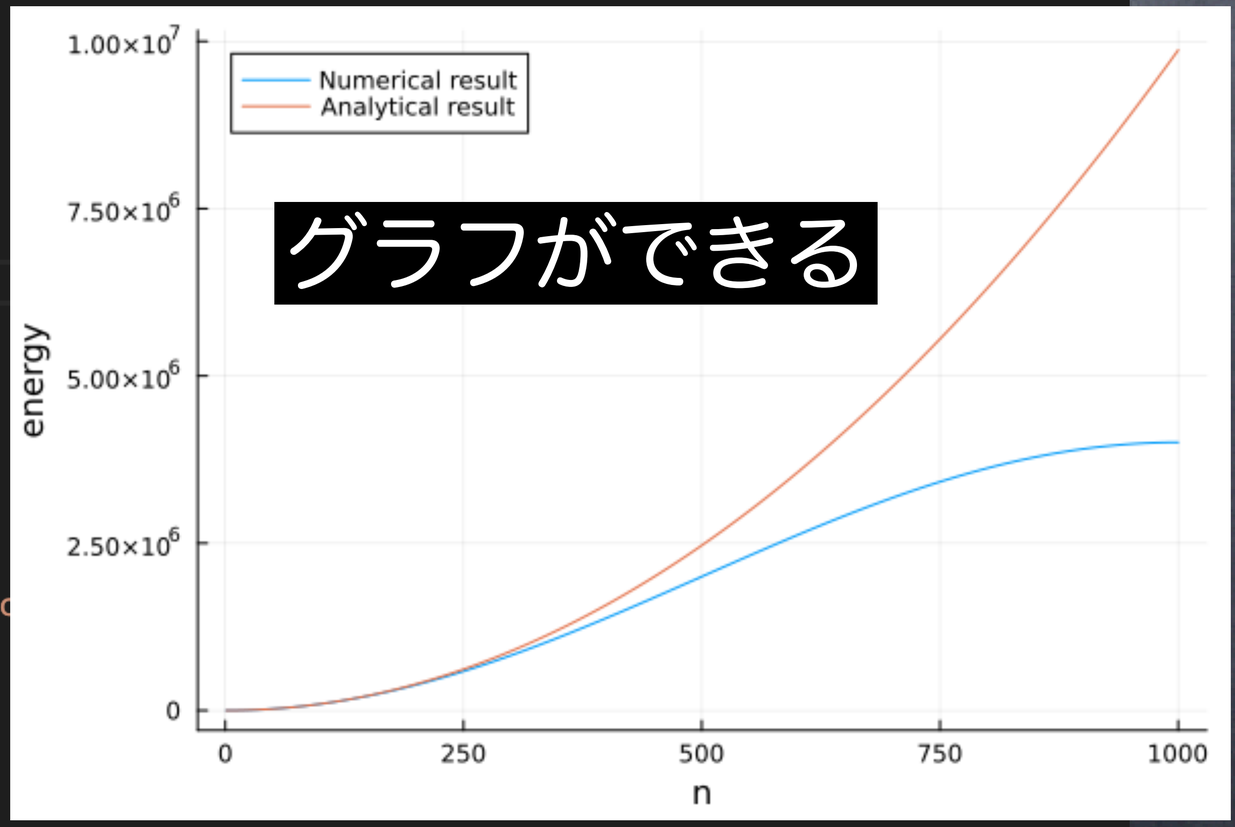
$$\hat{H}\psi_n(x) = E_n\psi_n(x) \qquad \hat{H} = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2} + V(x,y,z)$$

差分化して解いて、V=0の解析解と比較してみる

```
function make_H(N,L,V)
    Δx = L/(N+1)
    H = zeros(Float64,N,N)
    for i=1:N
        x = i*Δx
        H[i,i] = V(x)
        j = i+1
        dij = -1/Δx^2
        if 1 ≤ j ≤ N
            H[i,j] += dij
        end

        j=i
        dij = 2/Δx^2
        if 1 ≤ j ≤ N
            H[i,j] += dij
        end

        j=i-1
        dij = -1/Δx^2
        if 1 ≤ j ≤ N
            H[i,j] += dij
        end

    end
    return H
end
```

```
using LinearAlgebra

using Plots
function test()
    N=1000
    L=1000
    e,v = eigen(make_H(N,L,x->0))
    e0 = zeros(Float64,N)
    for n=1:N
        e0[n]=n^2*π^2/L^2
    end
    plot(1:N,[e,e0],labels=["Numeric
    savefig("eigen.png")
    println(e0[1],"\t",e[1])
end
test()
```

julia qm.jl
と実行すると



グラフができる

Julia言語とは

数値計算に最適な言語

Pythonのようなシンプルさ

C,Fortranに匹敵する速さ

matlabのような線形代数操作

Mathematicaのように積分したりできる

数値計算しようぜ

新しい言語なので、数値計算における”痒いところに手が届く”ような言語設計になっている

Juliaに関する質問はいつでも受け付けます

インストール

インストール

Windows OSの場合(Windows 11　Pro 22H2で確認)

　JuliaはWindowsストアにあるため、ストアの検索で「Julia」と入れ、指示に従ってインストールすることができます。Windows PowerShellを使う場合には

```
winget install julia -s msstore
```

とすることでインストールすることができます。どちらのインストール方法もやっていることは同じで、最新版のJuliaをインストールし、環境変数も設定してくれます。これで、Windows PowerShellで

```
julia
```

と入れれば実行する準備が整います。

Windows PowerShellを立ち上げる

Mac OSやLinux OSの場合

　端末（Macであればターミナル、Linuxであれば好きな端末）において、

```
curl -fsSL https://install.julialang.org | sh
```

を実行することでインストールができます。そのあとは.zshrcをリロードするか、新しい端末を開き直し

```
julia
```

ターミナルを立ち上げる

REPL

```
yuki@YukiMacBook-Pro-M2 ~ % julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.10.3 (2024-04-30)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

REPLと呼ばれる
電卓のように使える

Jupyter (Mathematicaみたいなノートブック)のインストール

```
julia
```

と入れると、JuliaのREPLが立ち上がる。次に、]キーを押して青文字の

```
(@v1.10) pkg>
```

のような状態になったら、

```
add IJulia
```

とする。これによって、Julia用のJupyter ノートブックがインストールされる。動くか確認するため、back spaceキーを押して、

```
julia>
```

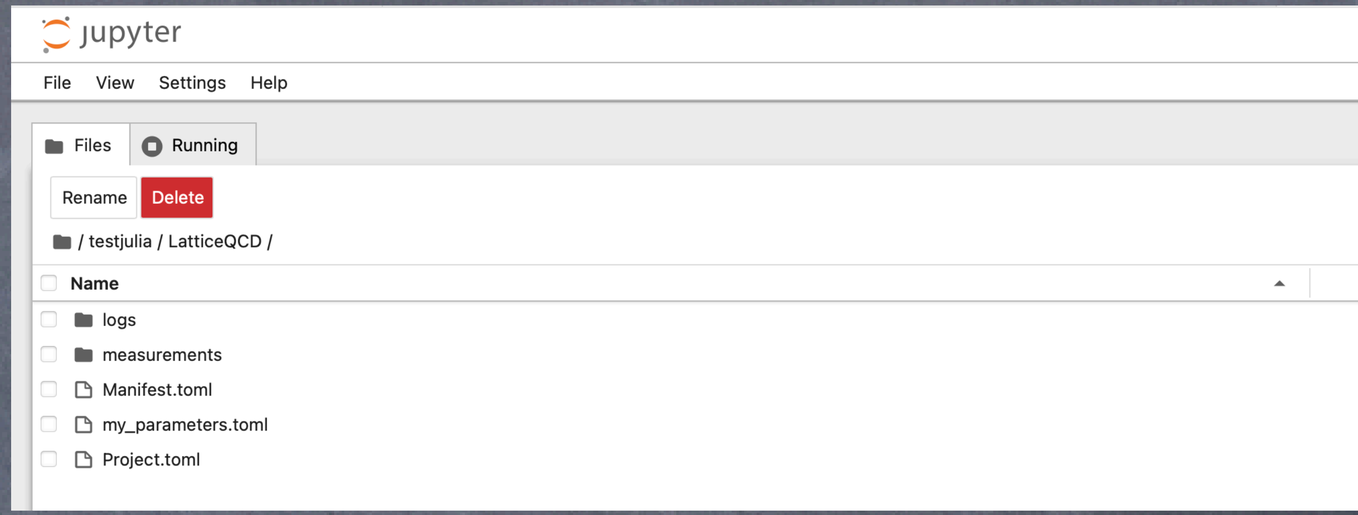
という緑色の文字の状態にしてから、

```
using IJulia
notebook()
```

とする。すると、

```
install Jupyter via Conda, y/n?
```

と聞かれるので、yを押す。
これでJupyterノートブックが立ち上がる。



こんな感じ

2. Jupyter notebookの使い方
   Newのボタンから**New Folder**を選ぶと新しいファイルができるので、**QCDtest**という名前をつける。そしてダブルクリックしてそのフォルダに入る。
   そして**New**ボタンを押すと「**Julia 1.10.5**」（バージョンは人によって異なるかもしれないが気にしない）があるのでそれをクリックする。すると、**notebook**が立ち上がる。

これで、

```
1 + 2
```

のように打ってから、**shift+enter**キーで、実行できる。さて、ここでは電卓的に様々な計算ができるので試してみよう。例えば、円周率は**pi**で使えるので、

```
sin(pi)
```

とすると**0.0**が返ってくるし、

```
sin(0.1)^2+cos(0.1)^2
```

とすると**1.0**が返ってくる。

3. Juliaの使い方
   Jupyter notebook上で色々なことができる。まず、可視化してみよう。Plotsパッケージを入れてみよう。notebookでは、

```
import Pkg
```

とした後に、

```
Pkg.add("Plots")
```

とすると、プロット関連のことができるPlotsというパッケージがインストールされる。

残りはnotebookで実際にやってみましょう