# テンソルネットワーク法入門（実習）

東大理　大久保　毅

# 実習の流れ

- 実習の準備
  - 講義のgithubサイトを開く
  - ノートブックの使い方練習
- 実習０：モンテカルロシミュレーション
  - Snap shotの観察
  - 正方格子イジング模型の物理量計算
- 実習1：テンソル繰り込み
  - 正方格子イジング模型の自由エネルギー計算
  - 有限系でのモンテカルロ法との比較
- 実習2：行列積状態
  - 無限系の正方格子イジング模型の自由エネルギー計算
  - テンソル繰り込みとの比較
- 全体のまとめ

# 実習の準備１：環境の準備

- ブラウザで講義のgithubに行く
  - https://github.com/TsuyoshiOkubo/Introduction-to-Tensor-Network
- このスライドのダウンロード
  - 上記サイトから「計算物理春の学校2023：テンソルネットワーク法入門（実習）.pdf」というファイルをダウンロード

# 実習の準備２：ノートブックのgoogle colabでの実行

- **チュートリアルファイルの準備**
  - github の中身
    - https://github.com/TsuyoshiOkubo/Introduction-to-Tensor-Network
  - Exercise フォルダの中身
    - Ex0-1.ipynb, Ex0-2.ipynb, Ex0-3.ipynb
    - Ex1-1.ipynb, Ex1-2.ipynb, Ex1-3.ipynb
    - Ex2-1.ipynb, Ex2-2.ipynb

    ］実習ファイルの本体
    - Ising_lib.py
    - TRG_lib.py
    - classical_iTEBD_lib.py

    ］アルゴリズムのモジュール
    - exact_output/ ］厳密解の結果
    - outputs/ ］実行済み計算結果（の一部）

# 実習の基本的な流れ

1. Ex*.ipynb をgoogle colabで開く

   - 「ドライブにコピー」

2. Ex*.ipynbをそのまま実行して結果を確認

3. インプットなどを変えて、結果の変化を確認

＊慣れている方はlocal の環境で実行してもOKです。

必要なモジュール：
- numpy
- scipy
- numba

# 実習０：（比較のため）
# モンテカルロシミュレーション

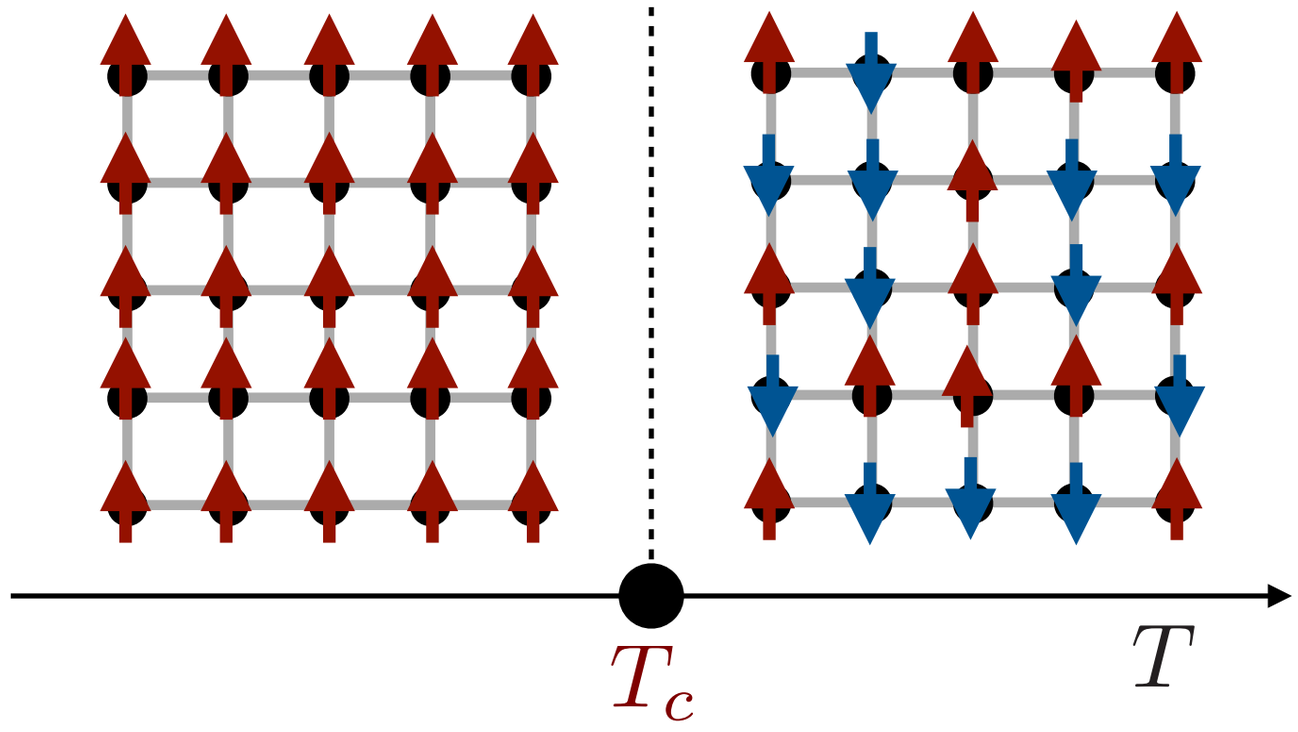
# MCMCの実習：平衡シミュレーション

- 正方格子イジング模型のシミュレーション

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} S_i S_j$$

  - $T=T_c$ で連続相転移

$$T_c/J = \frac{2}{\ln(1+\sqrt{2})} = 2.26918531\ldots$$

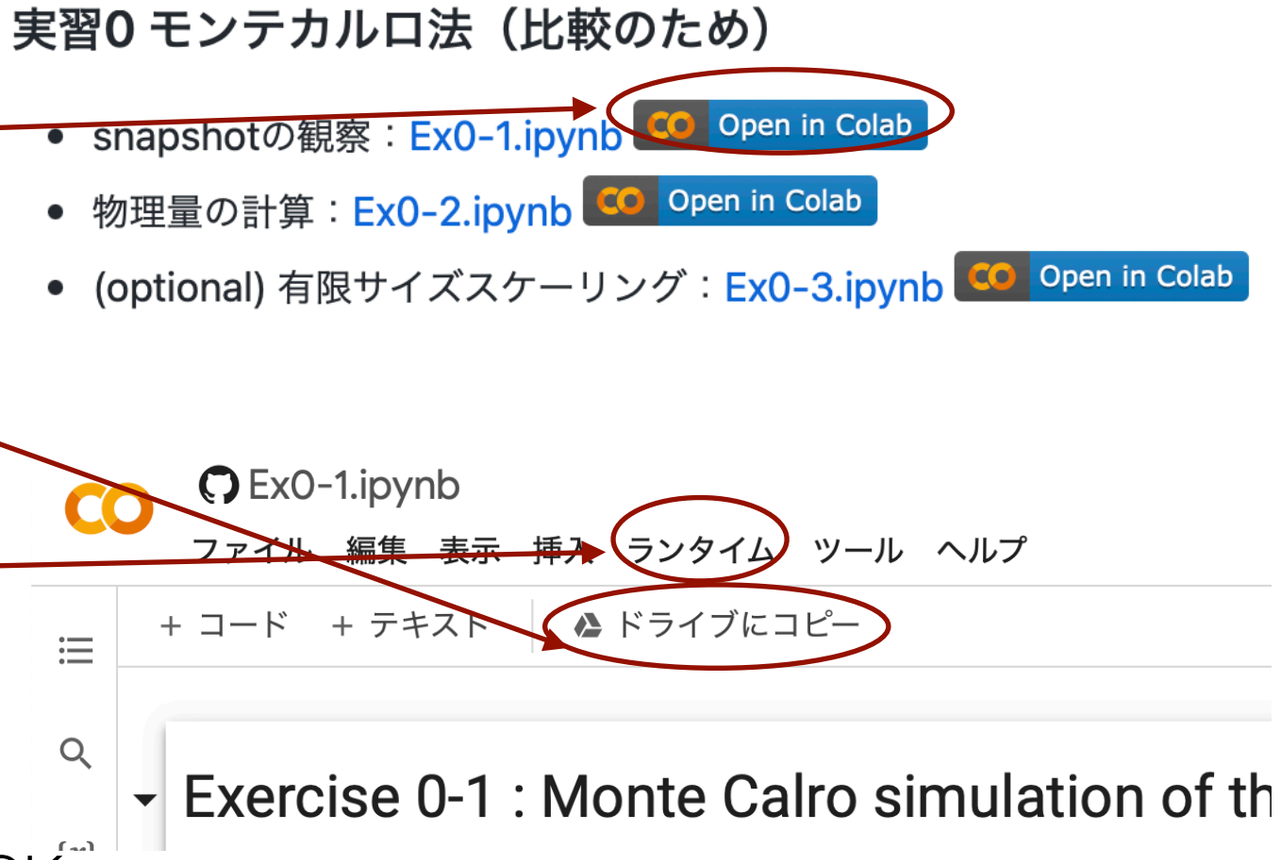


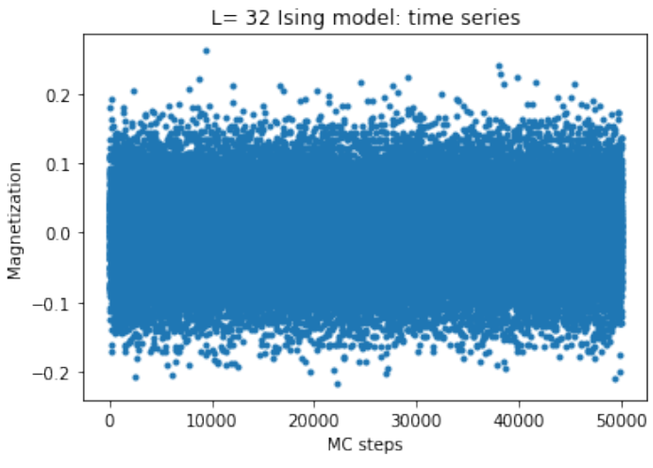

  - $T>T_c$：常磁性相
  - $T<T_c$：強磁性相

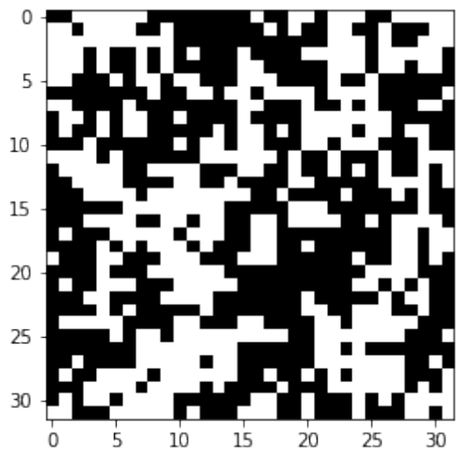
- 古典モンテカルロ法（メトロポリス法）のシミュレーションを実行して、平衡状態の物理量、時間変化、スピン配置をみる

# Ex0-1：イジング模型のシミュレーション

1. Google colabで*Ex0-1.ipynb*を開く
   - これをクリック
2. 「ドライブにコピー」
3. ランタイム/全てのセルを実行
   - 30~40秒くらいで終わるはず
   - 以下の図などが出力されていればOK



# Ex0-1.ipynbのinput

## MCMCで正方格子イジング模型の物理量を計算します

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

```
[2]  #Parameters for calculation
     Tc = 2.0/np.log(1.0+np.sqrt(2.0))

     ### input parameters ###
     L=32
     T=2.0*Tc ## T means T/J
     h = 0

     algorithm="metropolis"
     #algorithm="heatbath"
     #algorithm = "cluster"

     random_seed = None # 11
     thermalization = 10000
     observation = 50000

     ########################
```

厳密なTc（変更しない）

*L*: システムサイズ

*T*: 温度

*h*: 磁場（今回は基本的に0）

講義で説明した
・メトロポリス法
の他に、
・熱浴法（*heatbath*)
・*Swendsen-Wang* 法（*cluster*）
も選べます

アルゴリズム

乱数のタネ

（注）同じタネだと、
同じ結果
（*None*だと毎回違う）

物理量を計算せずに、*MCMC*を行う*step*数
（初期状態依存性を少なくするため）

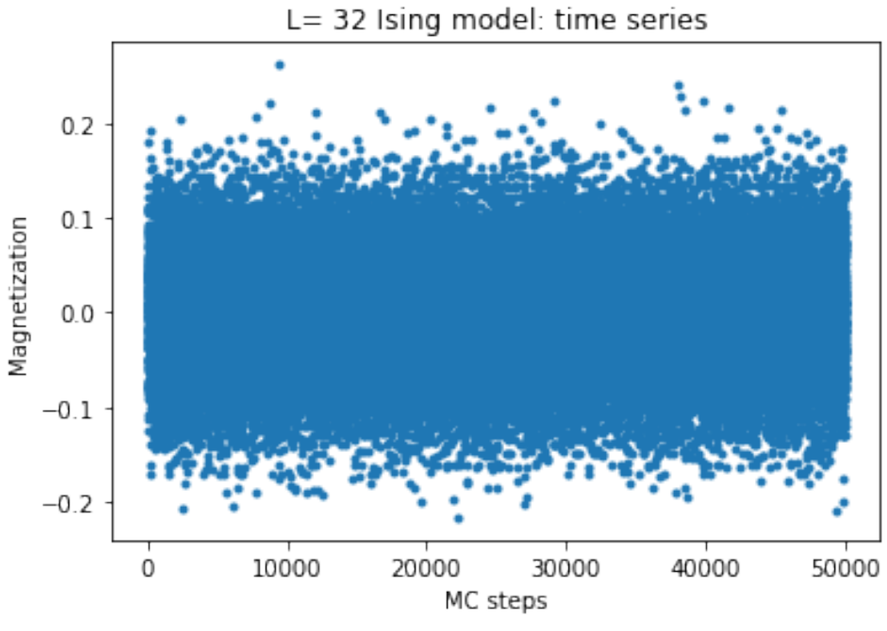
物理量を計算する*step*数

# Ex0-1.ipynbのoutput

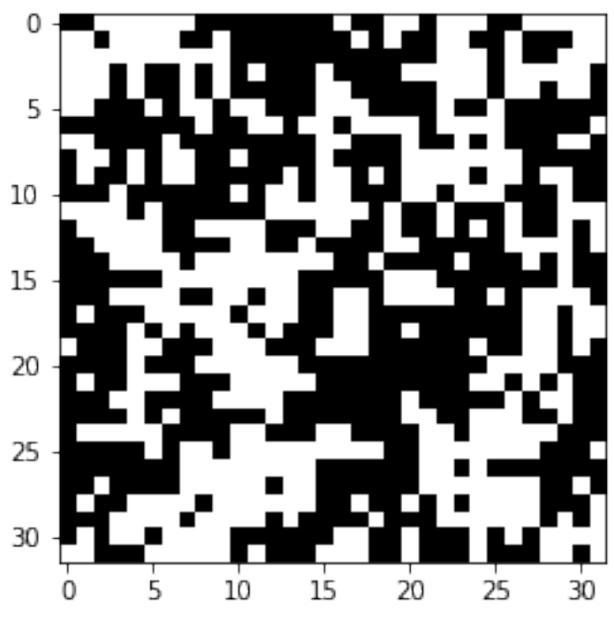
```
[Outputs with errors estimated by Jackknife method]
T = 4.538370628426044
Energy = -0.47929273437500003 +- 0.0003877741513014861
Energy^2 = 0.2322261947631836 +- 0.0003749347065671236
Magnetization = -4.5156250000000064e-05 +- 0.00029356012661862124
Magnetization^2 = 0.003279116973876952 +- 1.9936339089795468e-05
Magnetization^4 = 3.2012081598513764e-05 +- 4.408930118636691e-07
Specific heat = 0.12452318130922689 +- 0.0009802312918727957
Susceptibility = 0.7398725349177852 +- 0.0044982688500764065
Connected Susceptibility = 0.2671902989849985 +- 0.0019568959196030276
Binder ratio = 2.9771448026203218 +- 0.021618961306956737
```

物理量と誤差（Jackknife法）

物理量のダイナミクス




最終状態のsnapshot

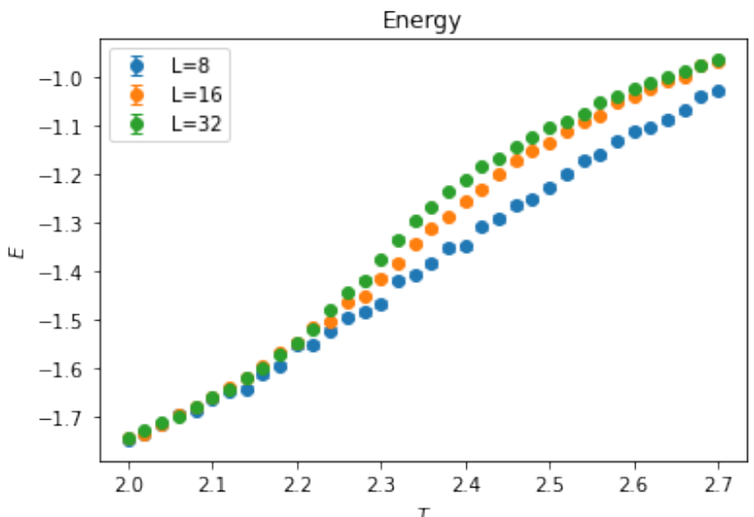


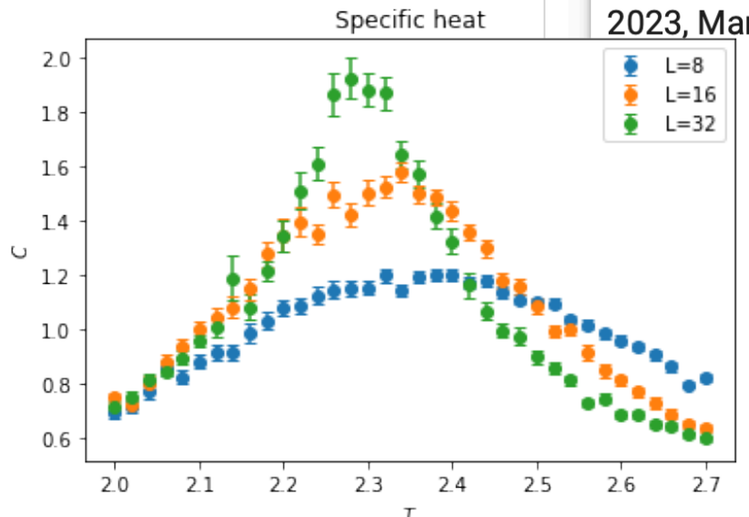
# 実習0-1

- 温度Tを $T > T_c$, $T = T_c$, $T < T_c$ などで変えてみて、ダイナミクス、スナップショットの違いを観測する
  - 物理量によってダイナミクスが違う場合（例：低温の磁化）
- (optional) 温度を固定して$L$を変えてみて、ダイナミクスの違いを観測する
  - 特に$T=T_c$ 近傍で、$L$の増大でダイナミクスが遅くなるはず
  - ダイナミクスが"ゆっくり"になると物理量の誤差が大きくなる

# Ex0-2：温度依存性の解析

1. Google colabで*Ex0-1.ipynb*を開く
   - これをクリック
2. 「ドライブにコピー」
3. ランタイム/全てのセルを実行
   - 1分程で終わるはず
   - 以下の図などが出力されていればOK

# Ex0-2.ipynbのinput

## MCMCで正方格子イジング模型の物理量を計算します

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

**Ex0-1.ipynbから追加・変更があった部分**

```
Read_simulation_data = False ## I
data_file = "mcdata_ex1-2.dat" ##
```

Trueにすると計算はせずplotだけ

上記がTrueの時はinput_file名、
Falseの時はoutput file名

```
#Parameters for calculation
L_list=[8,16,32]
T_list=np.arange(2.0,2.7,0.02) #
T_list_all = []
for L in L_list:
    T_list_all.append(T_list)
```

計算するLのリスト（この例では、L=8, 16,32を計算）

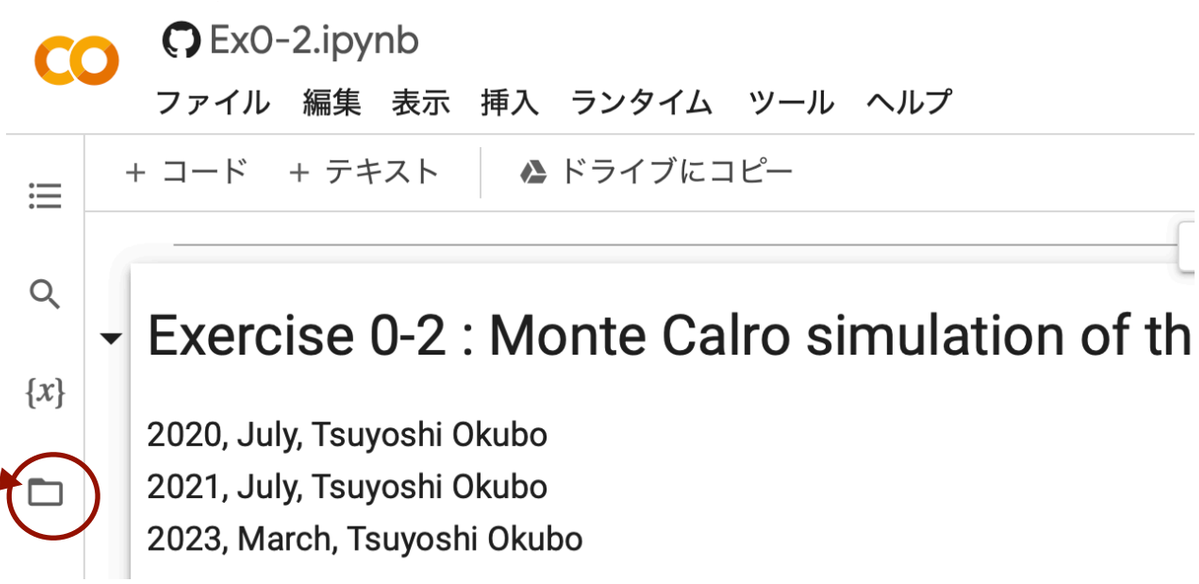
計算するTの生成。この例では、
T_min = 2.0, T_max = 2.7で
温度間隔 T_step = 0.02の計算

原理的にはL毎に異なる温度セットで計算できる
（今の書き方では、全てのLで共通の温度セット）

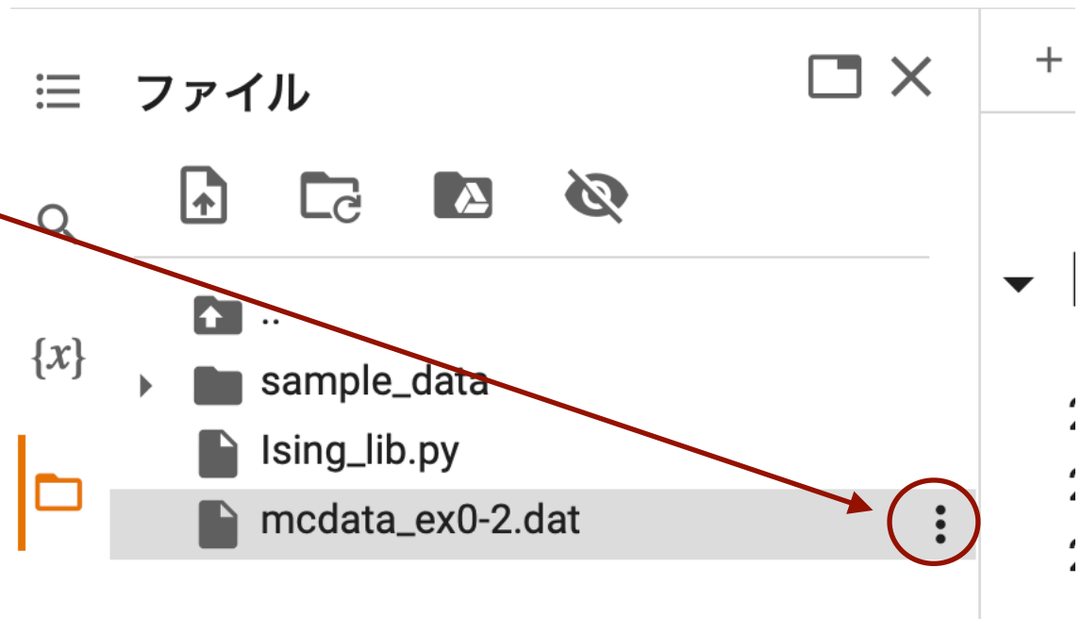
# 実行結果のダウンロード

Google colabのセッションが終了するとdata_fileなどは消えます。
必要であれば、適宜ダウンロード

1. これをクリック




2. これをクリックして
ダウンロードを選ぶ




＊後でテンソル繰り込み群との比較に
使います

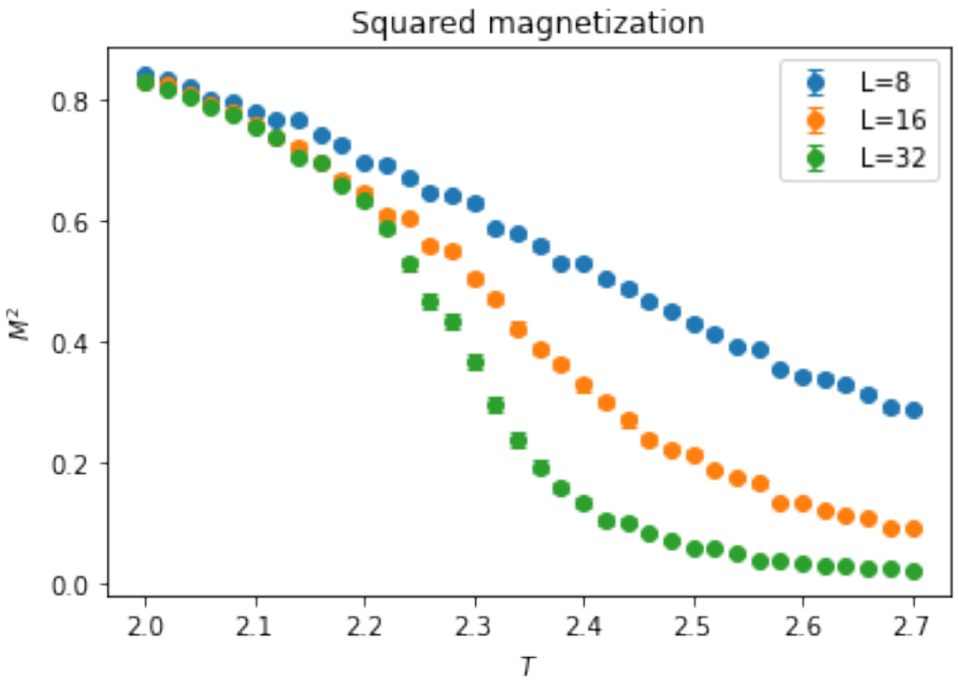
# Ex0-2.ipynbのoutput

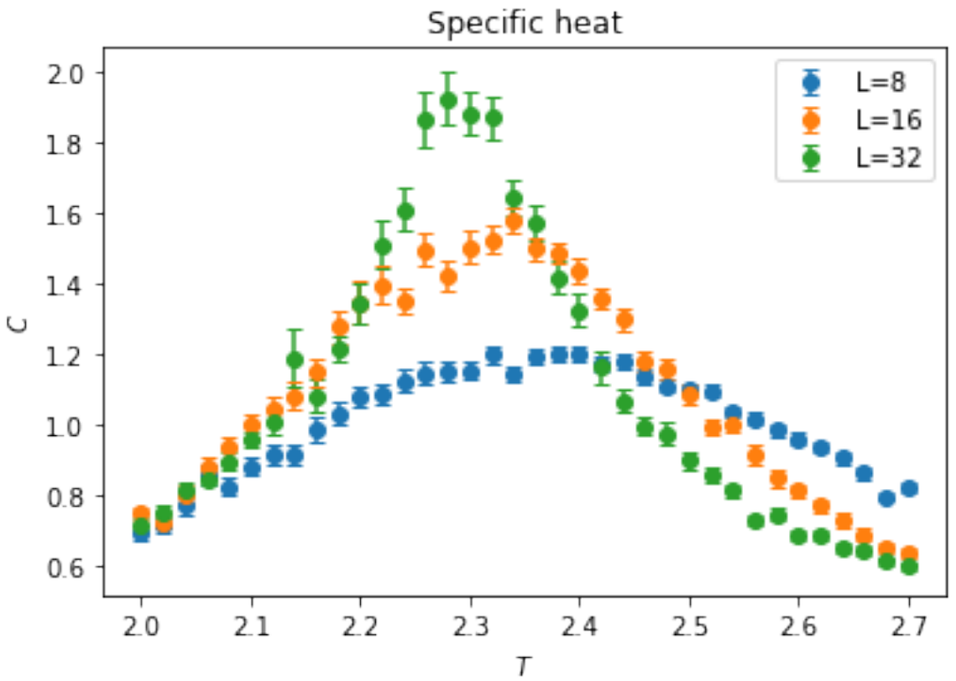
物理量の温度依存性：

- エネルギー
- 比熱
- 磁化の２乗
- 磁化率

見どころ：

- 物理量は誤差を含んでいる
  - エラーバーの大きさは、
    温度、サイズ、物理量に依存
- 比熱や磁化率は$T_c$近傍にピーク
  - $L$ の増大で真の$T_c$に近づく



# 実習0-2

- モンテカルロステップ（observation）を増減させてエラーバーの変化を見よう
  - モンテカルロステップを増やすとエラーバーはちゃんと減る？
    - （初期）observation = 10000 → 40000 に変更してみると？
- リストに$L=64$ を追加して振る舞いを見よう
  - 計算時間はどうなるか？（途中のセルに時間が出力されてる）

```
CPU times: user 1min 19s, sys: 119 ms, total: 1min 19s
Wall time: 1min 19s
```

  - observationを固定したままだと$L=64$のエラーバーどうなる？

＊この結果を保存しておくこと推奨

# 実習1：

# テンソルネットワーク繰り込み群

# テンソルネットワーク繰り込み群

- 分配関数のテンソルネットワーク表現を粗視化していくことで、近似的に分配関数を計算する
    - 粗視化⟷実空間繰り込み群
- アルゴリズムは「特異値分解」と「テンソルの縮約」を繰り返すだけの単純なものであり、例えば、pythonの数値計算ライブラリNumPyを用いれば、非常に簡単に実装できる
- 種々の格子模型に適用可能
    - 分配関数を表すテンソルさえ準備すれば、アルゴリズム（プログラム）は種々の模型に適用可能
    - 物性分野だけでなく、素粒子・原子核分野でも近年研究が進んでいる

# TRGの実習内容

- 正方格子イジング模型のシミュレーション

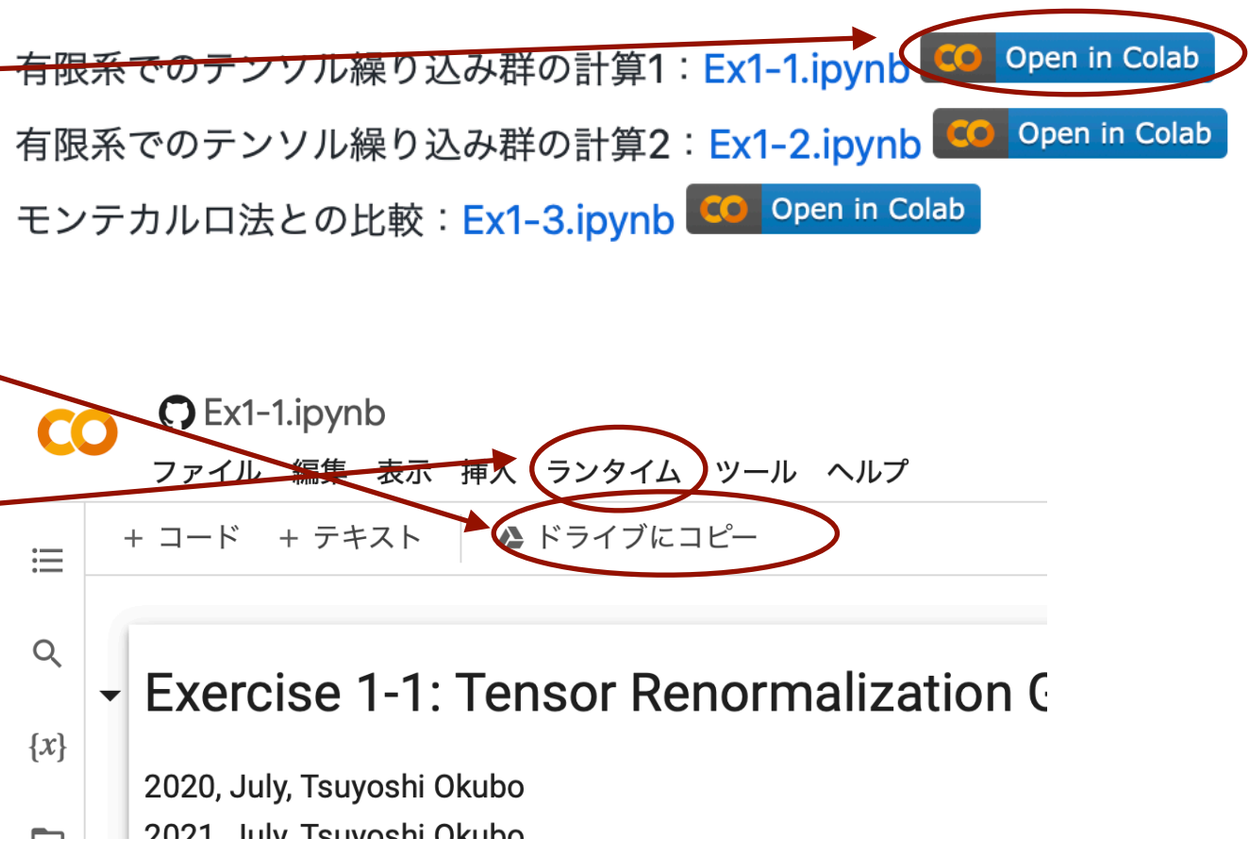
  1. 自由エネルギーの計算と厳密解との比較

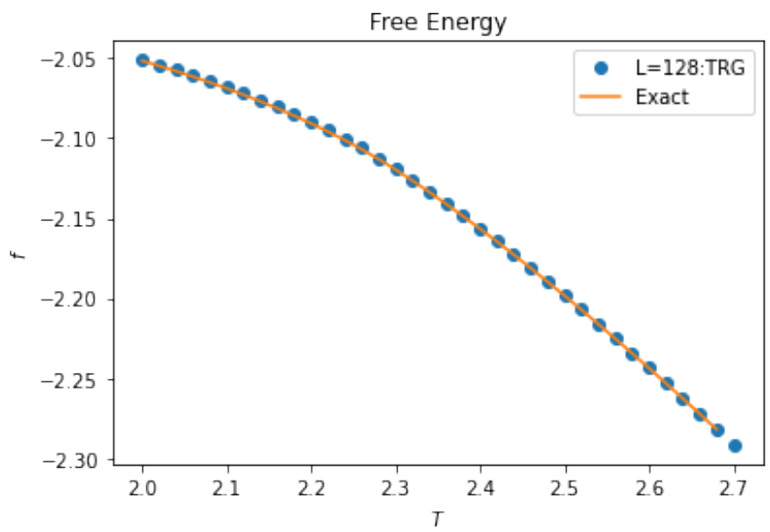
  2. 差分による、比熱、エネルギーの計算

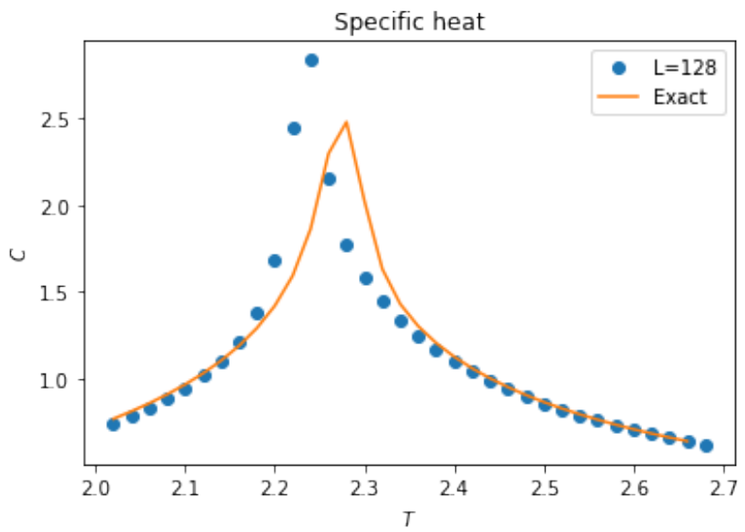
  3. 実習０で行ったモンテカルロシミュレーションとの比較

# Ex1-1：TRGの実行と厳密解との比較

1. Google colabで*Ex1-1.ipynb*を開く
   - これをクリック
2. 「ドライブにコピー」
3. ランタイム/全てのセルを実行
   - 数十秒で終わるはず
   - 以下の図などが出力されていればOK



# Ex1-1.ipynbのinput

## TRGで正方格子イジング模型の物理量を計算します

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

```
Tc = 2.0/np.log(1.0+np.sqrt(2.0)) ## The cri

### input parameters ###
n = 7 ## L = 2^n
T_min = 2.0
T_max = 2.7
T_step = 0.02

T_list=np.arange(T_min,T_max,T_step)

D = 4 ## bond dimension
### input parameters ###
```

厳密なTc（変更しない）

*L*=$2^n$: システムサイズ。nで指定

*T_min*, *T_max*, *T_step*: 温度

*D*：低ランク近似のパラメタ（ボンド次元）

# Ex1-1.ipynbのoutput

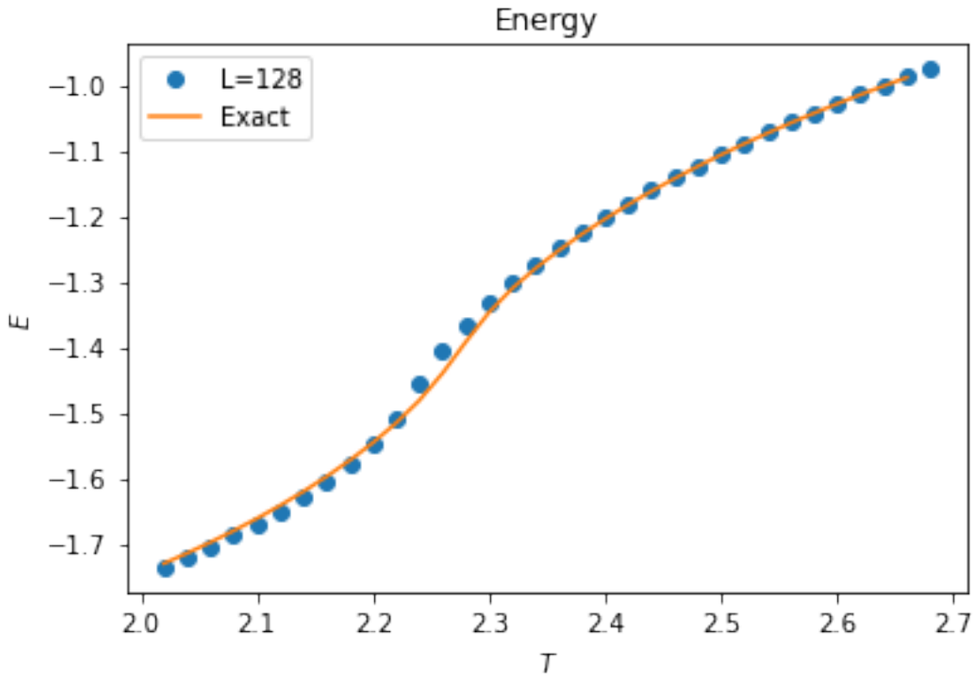
**TRGによる自由エネルギー**

```
T, free_energy_density = 2.0 -2.0511359452135833
T, free_energy_density = 2.02 -2.0542110601942833
T, free_energy_density = 2.04 -2.0574326609027627
T, free_energy_density = 2.06 -2.0608076301385387
T, free_energy_density = 2.08 -2.0643436128975106
T, free_energy_density = 2.1 -2.068049238287201
```

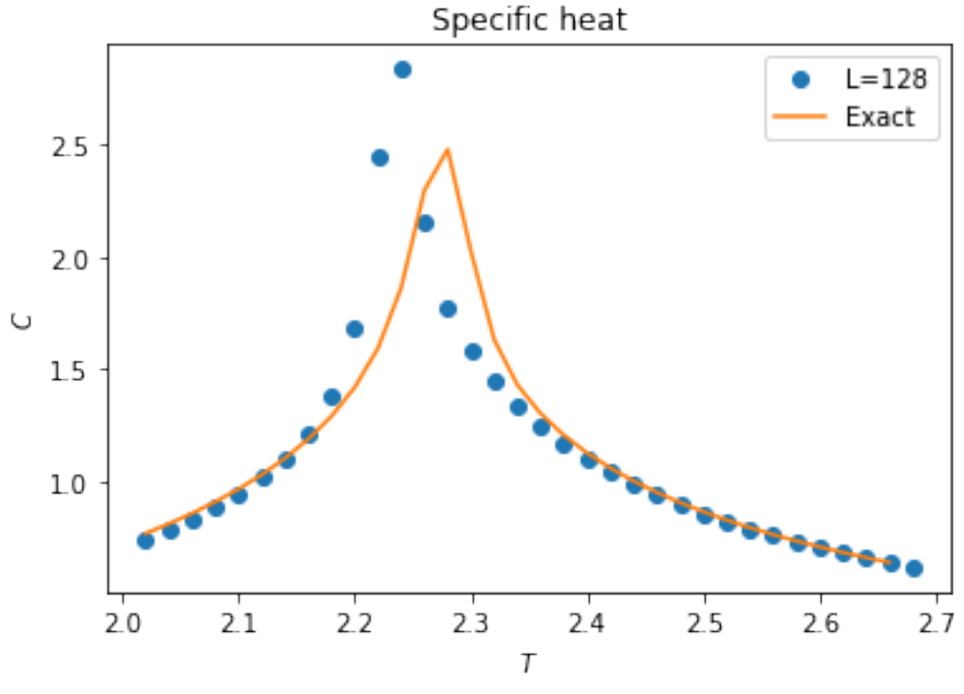
*厳密解は藤堂先生のexact ライブラリを使って計算しました。

https://github.com/todo-group/exact

差分近似で求めたエネルギー




差分近似で求めた比熱

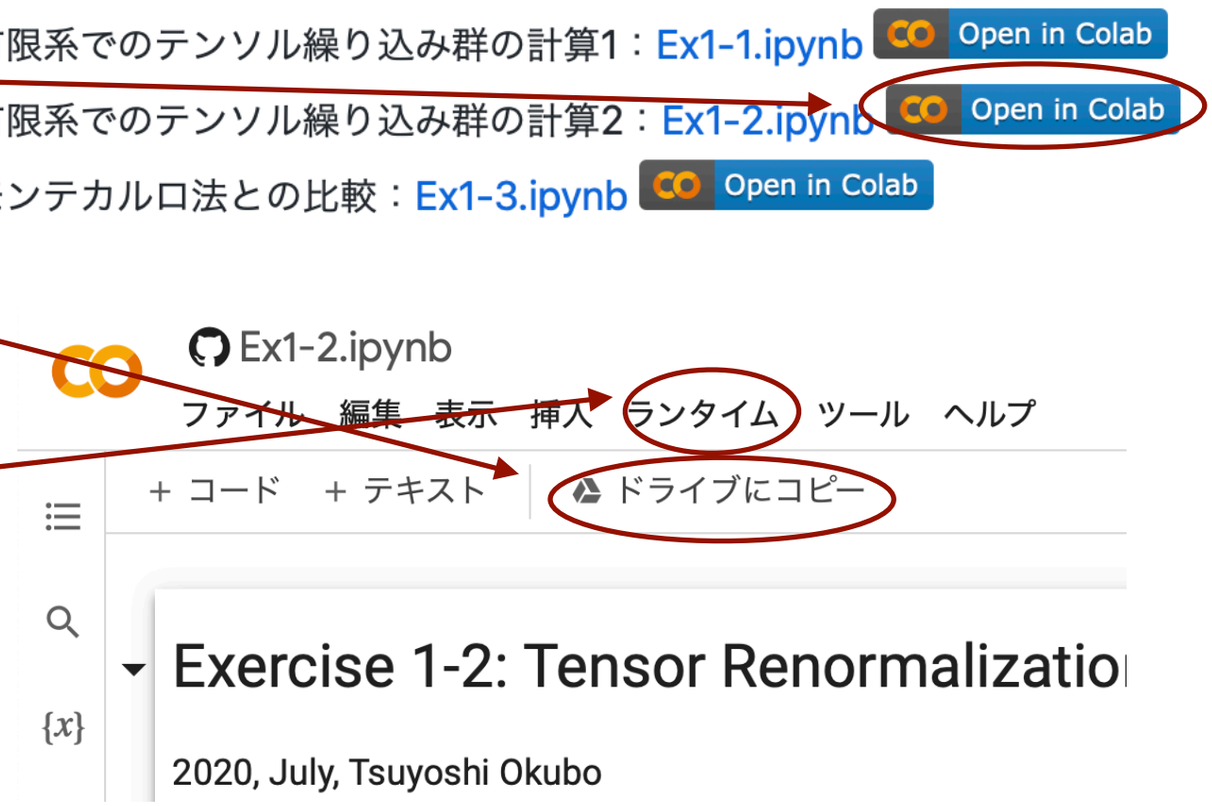


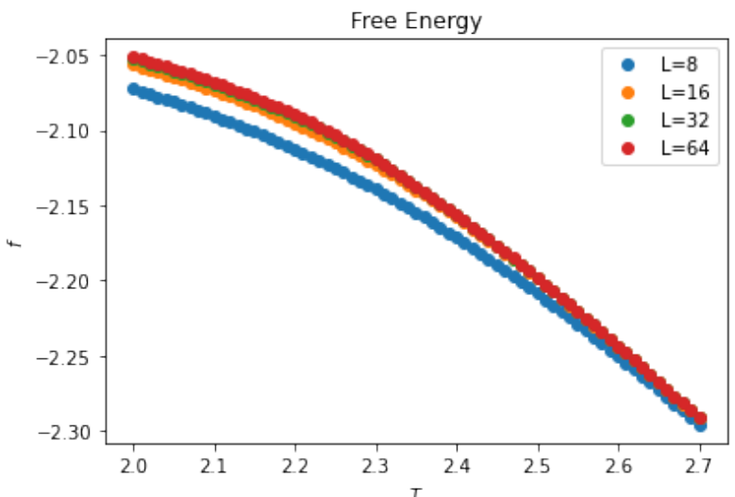
# 実習1-1

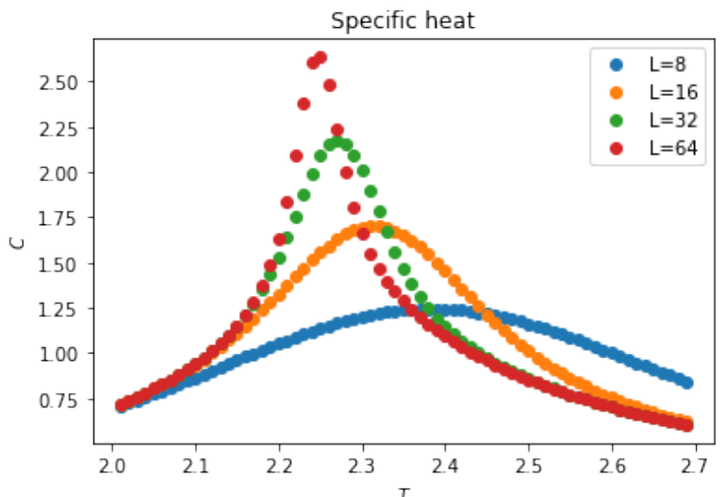
- $L\ (=2^n)$ を変えてみて、計算時間の変化を確認する
    - $n$が1増えると$L$は2倍になるが、計算時間はどう変わるか？
- $D$を変えて、計算時間の変化を確認する
    - （補足情報）TRGにおけるSVDとテンソルの縮約の計算コストは$O(D^6)$
- $D$を変えて、物理量の精度を確認する
    - 自由エネルギーは数字を比べると分かりやすい
    - 物理量はグラフで比べるとよい
        - 物理量は差分近似で求めている。この近似精度はT_stepを小さくすることで上げることができる
        - 厳密解との差は温度によってどう変わっているか？
    - $D$を固定したまま$L$を変えるとどうなるか？

# Ex1-2：複数サイズでのTRGの実行

1. Google colabで*Ex1-2.ipynb*を開く
   - これをクリック
2. 「ドライブにコピー」
3. ランタイム/全てのセルを実行
   - 数十秒で終わるはず
   - 以下の図などが出力されていればOK




*次の実習で使うので、実行結果
（trgdata_ex1-2_D4.dat）
をダウンロードしておいてください

# Ex1-2.ipynbのinput

## TRGで正方格子イジング模型の物理量を計算します

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

```
### input parameters ###
n_max = 6
n_min = 3

T_min = 2.0
T_max = 2.7
T_step = 0.01

D = 4 ## bond dimension

data_file = "trgdata_ex2-2_D"+repr(D)+".dat" #
### input parameters ###
```

*L=2^n*: システムサイズ。
n_min ~ n_maxまで計算

*T_min, T_max, T_step*: 温度

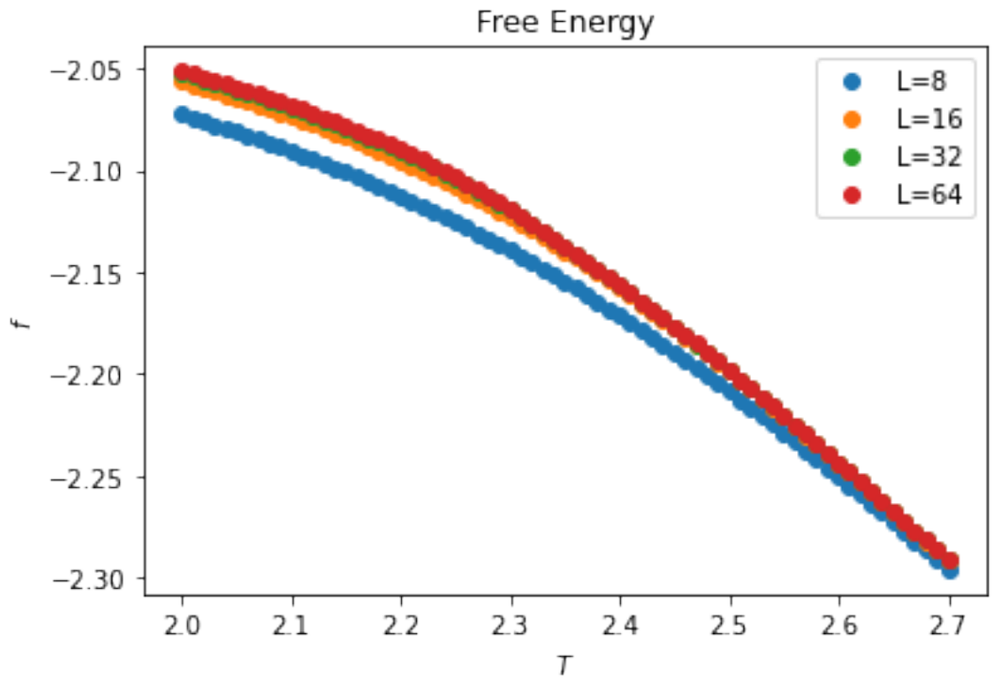
*D*：低ランク近似のパラメタ（ボンド次元）

data_file：計算結果を記録するファイル名。
ここでは、Dに応じて違う名前になるようにしている

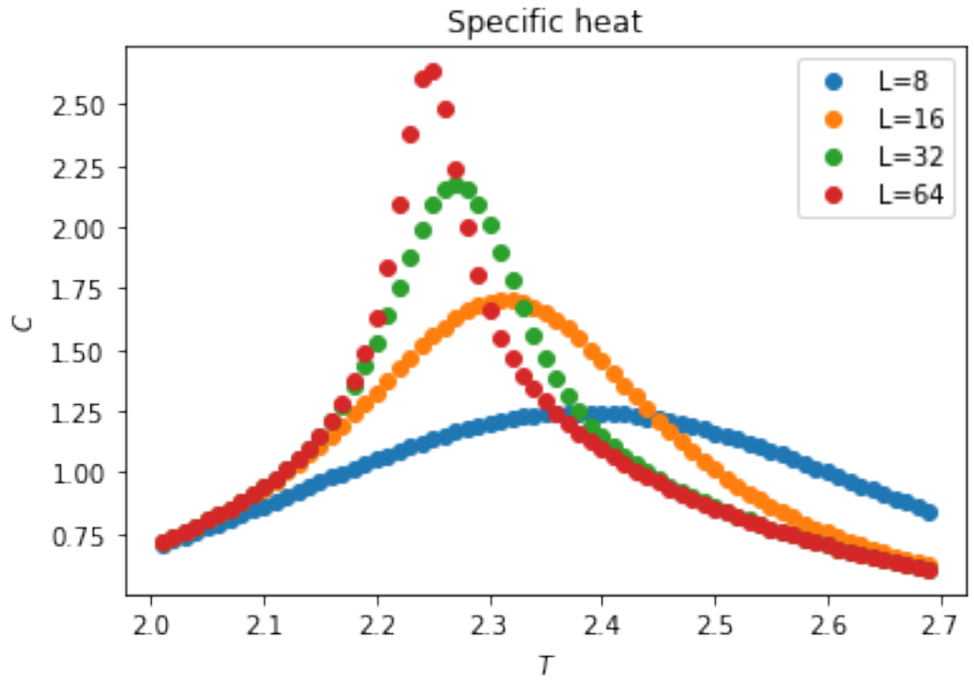
*google colabのセッションが終了するとdata_fileなどは消えます。
必要であれば、適宜ダウンロードしてください

# Ex1-2.ipynbのoutput

差分近似で求めたエネルギー




差分近似で求めた比熱




見どころ：

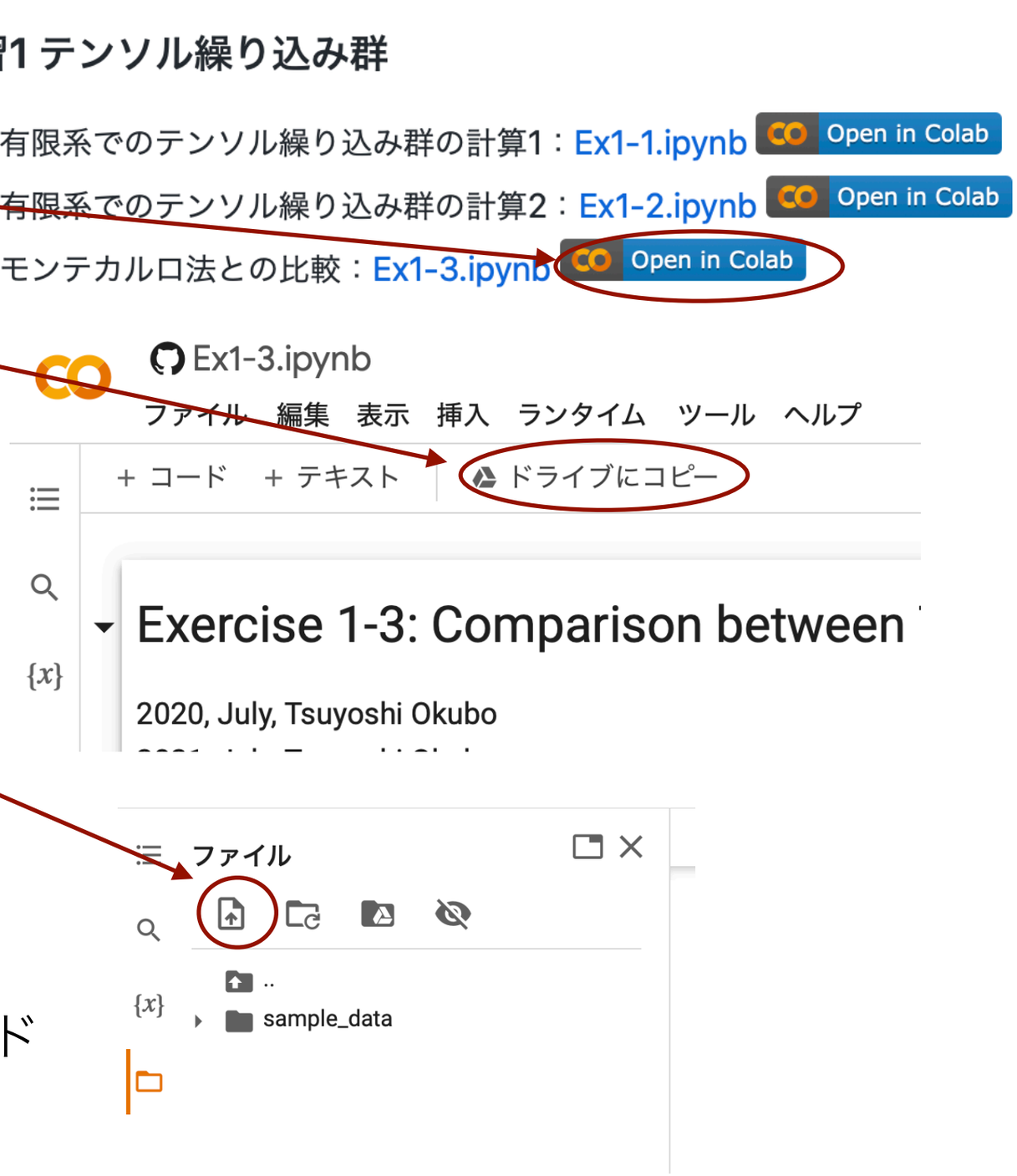
- 物理量には統計誤差はなく、データは滑らか
  - Dによっては、自由エネルギーの不連続性に起因する異常が出る場合もある

# 実習1-2（1-1でやっていればやらなくても良い）

- $D$やT_stepを変えて、物理量の変化を確認する
- 大きなLの計算も行い、物理量のL依存性を確認する
- 大きなLで比熱は発散するだろうか？

# Ex1-3：モンテカルロ法とTRGの比較

1. Google colabで*Ex1-2.ipynb*を開く
   - これをクリック
2. 「ドライブにコピー」
3. ファイルのアップロード
   これをクリックして
   保存しておいた
   *mc_ex0-2.dat*と*trg_ex1-2_D4.dat*を
   アップロード
   （実はuploadしなくても、githubから
   事前に計算した結果を自動でダウンロード
   してくれると思います。）

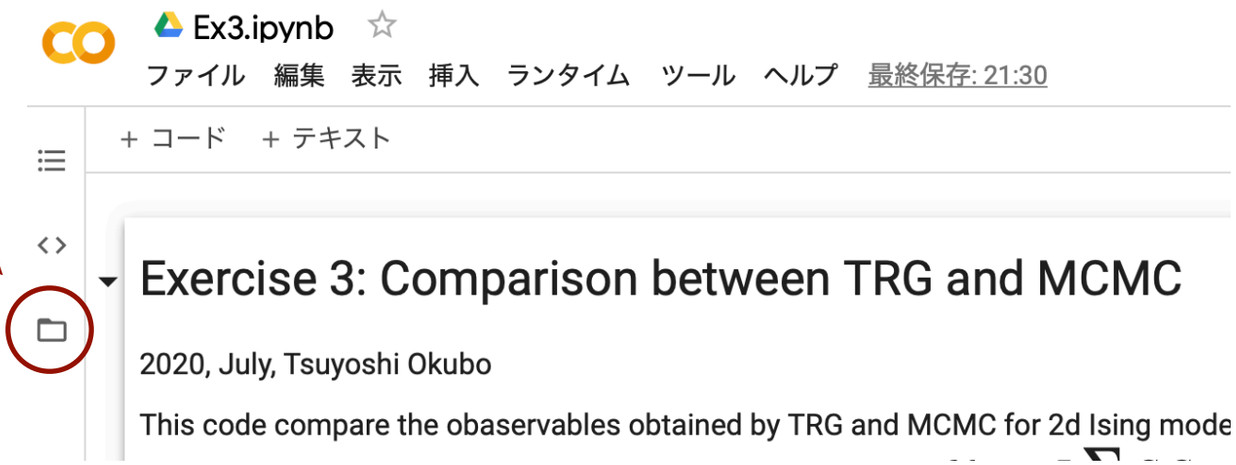


# Ex1-3：モンテカルロ法とTRGの比較

1. Google colabで*Ex1-3.ipynb*を開く

2. これをクリック

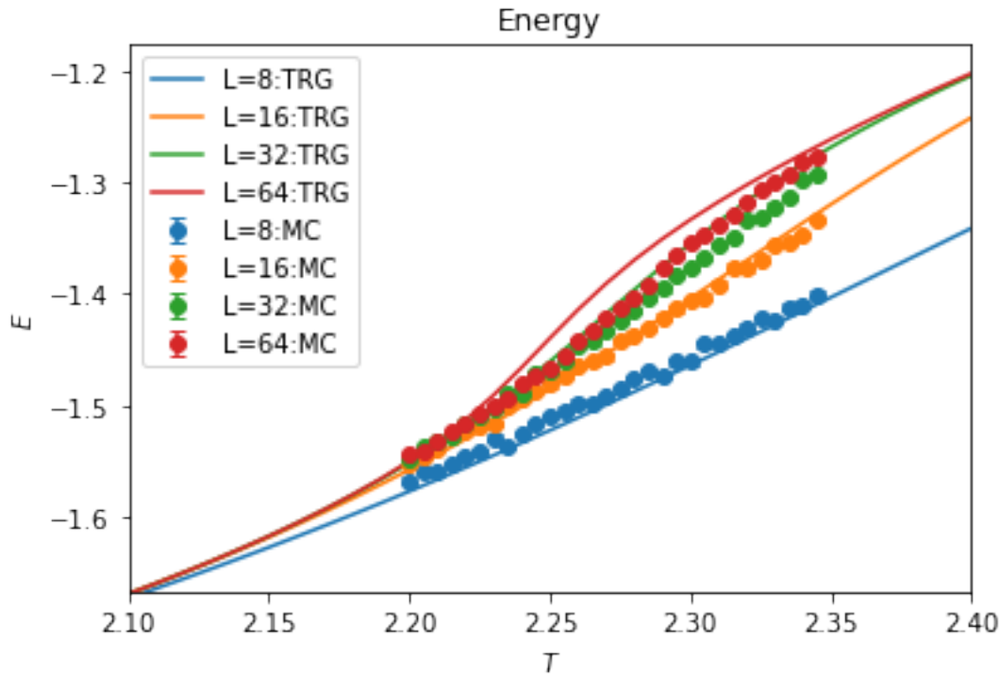
（接続するまで少し待つ）

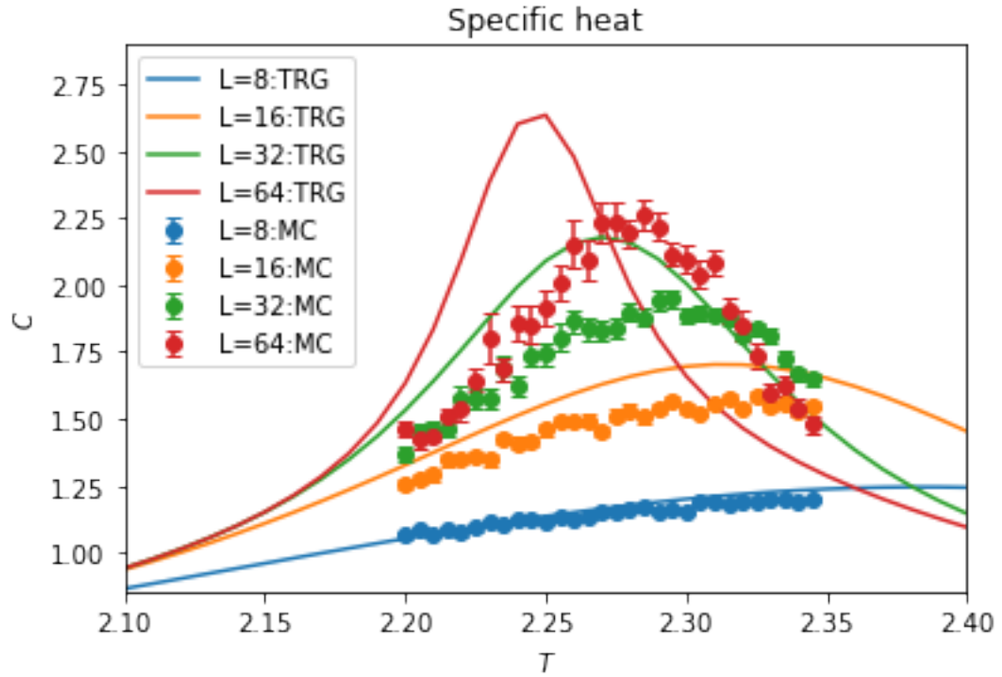



3. これをクリックして
保存しておいた
*mc_ex0-2.dat*と*trg_ex2-2_D4.dat*を
アップロード
（保存し忘れてた場合は、再度計算するか、
配布ファイルの”outputs/"の中のものを使う）

# Ex1-3：モンテカルロ法とTRGの比較

4. 上部メニューから"ランタイム/すべてのセルを実行"を選ぶ
    - すぐに終わるはず
    - 下の方までスクロールして、以下の図などが出力されていればOK



# Ex1-3.ipynbのinput

## モンテカルロ法とTRG結果を読みこんでプロットします

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

```
## plot temperature range
T_min = 2.1
T_max = 2.4
```

*T_min*, *T_max*: プロットする温度範囲

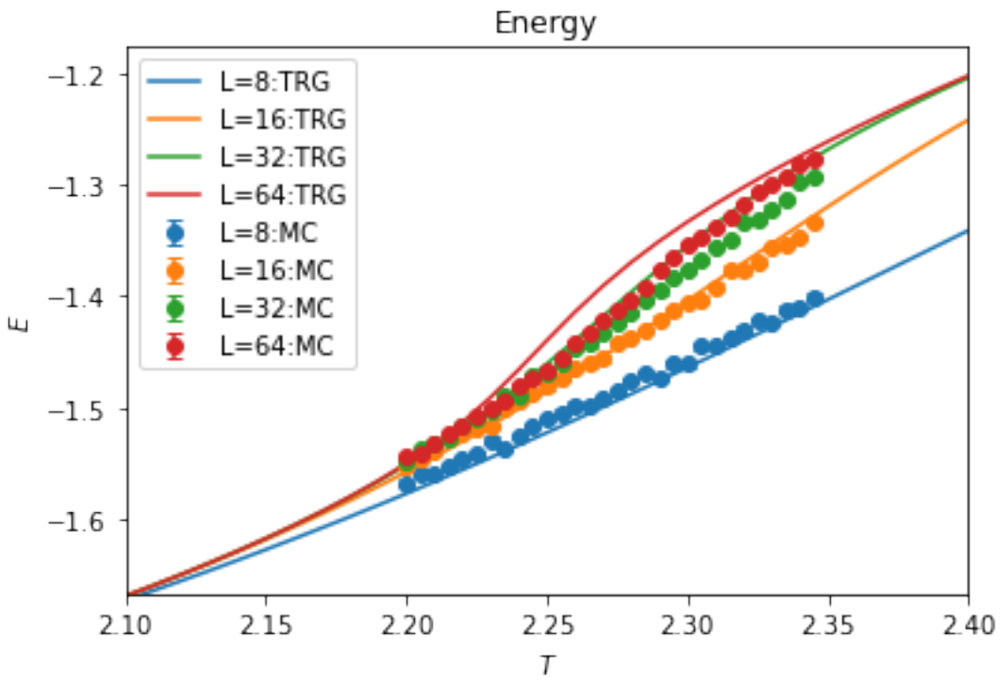
```
## data file
mc_data = "mcdata_ex1-3.dat" ## MC simulation data
trg_data = "trgdata_ex2-2_D4.dat" ## MC simulation data
```

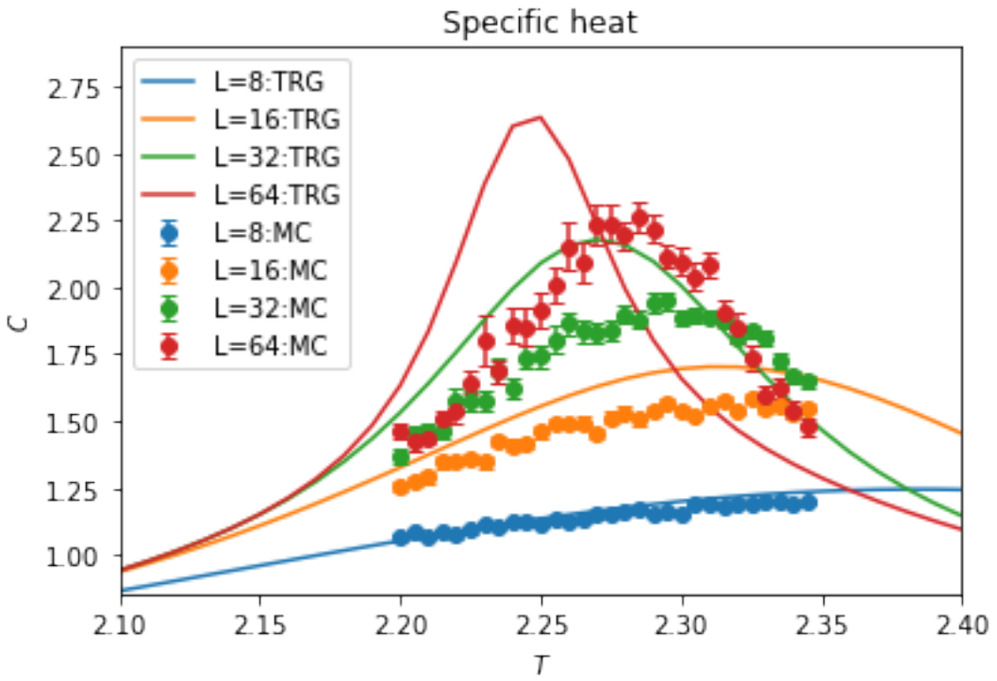
結果が記録されているファイル名

（注）
モンテカルロ法と*TRG*で、同じ*L*が
計算されていることを仮定しています。
*L*のセットがずれていると、プロットした際に、
・凡例が正しくなくなる
・異なる*L*が同じ色になる
などの不具合が生じます。
温度範囲、刻みは異なっていて問題ありません。

# Ex1-3.ipynbのoutput

モンテカルロ法とTRGによって得られたエネルギー・比熱を同時プロット




見どころ：

- モンテカルロ法とTRGの結果、$L$, $D$によってはズレが大きい場合がある

# 実習3

- 以前のEx1-2.ipynbや
  Ex2-2.ipynbでパラメタを変えて実行した結果をプロットしてみよう

  - 計算結果をアップロードして、読み込むファイル名を適切に変更して実行すればよい。

- モンテカルロ法とTRGのズレは、計算のパラメタによってどう変わるだろうか？

  - TRGには系統誤差が、モンテカルロ法には統計誤差があることに注意

# 実習2：
# 行列積状態法

# 行列積状態法

- ベクトル（テンソル）を行列の積で分解して近似する表現方法
  - 行列積状態（MPS）、tensor train decomposition
- 分配関数のテンソルネットワーク表現の縮約において、転送行列の固有ベクトルを計算する際に利用可能
  - 今回は特に、iMPSを用いて、無限系の計算を行う
- 対応するMPSの計算は、逐次的に転送行列をかけていけば良い
  - 転送行列をかける計算は、iTEBDで効率的に近似できる。

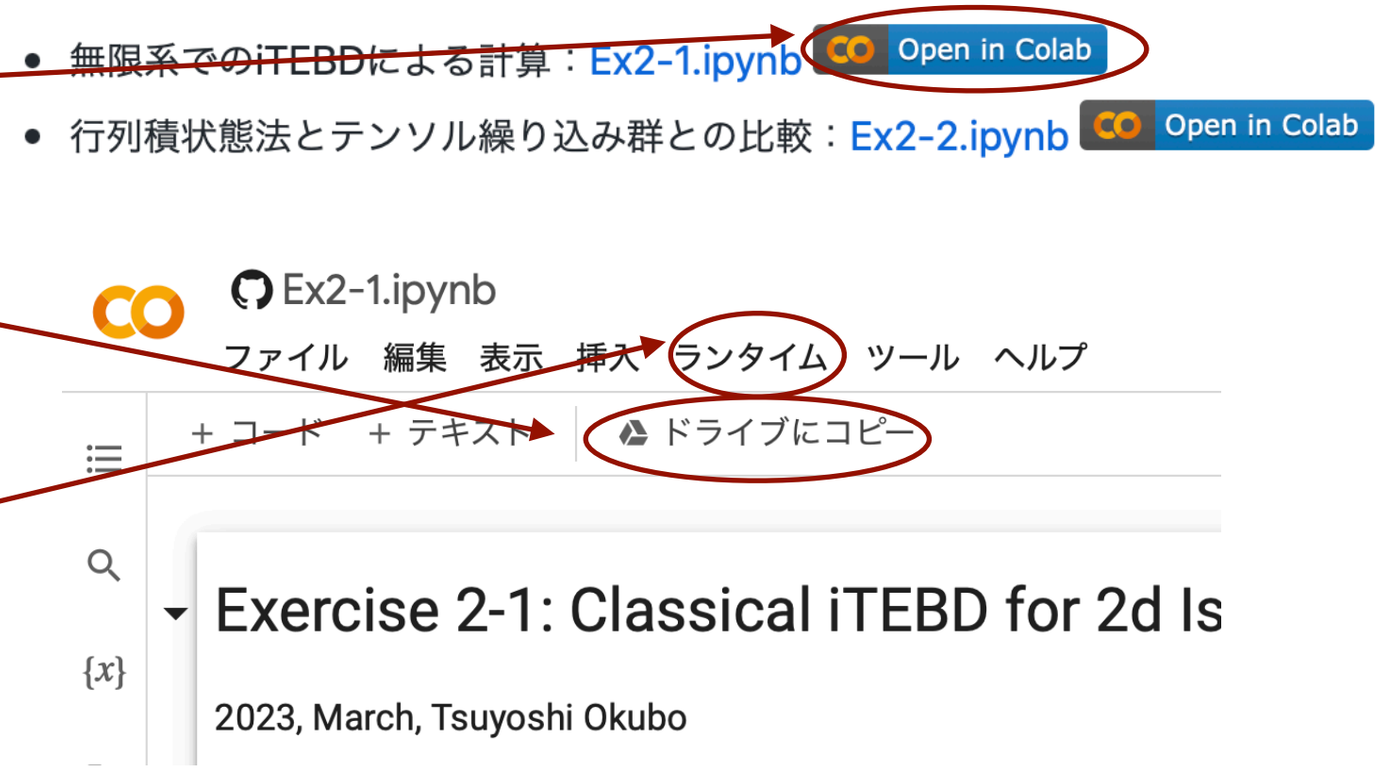
# 行列積状態の実習内容

- 無限系の正方格子イジング模型のシミュレーション

    1. 自由エネルギー（と磁化）の計算と厳密解との比較
    2. テンソル繰り込み群との比較

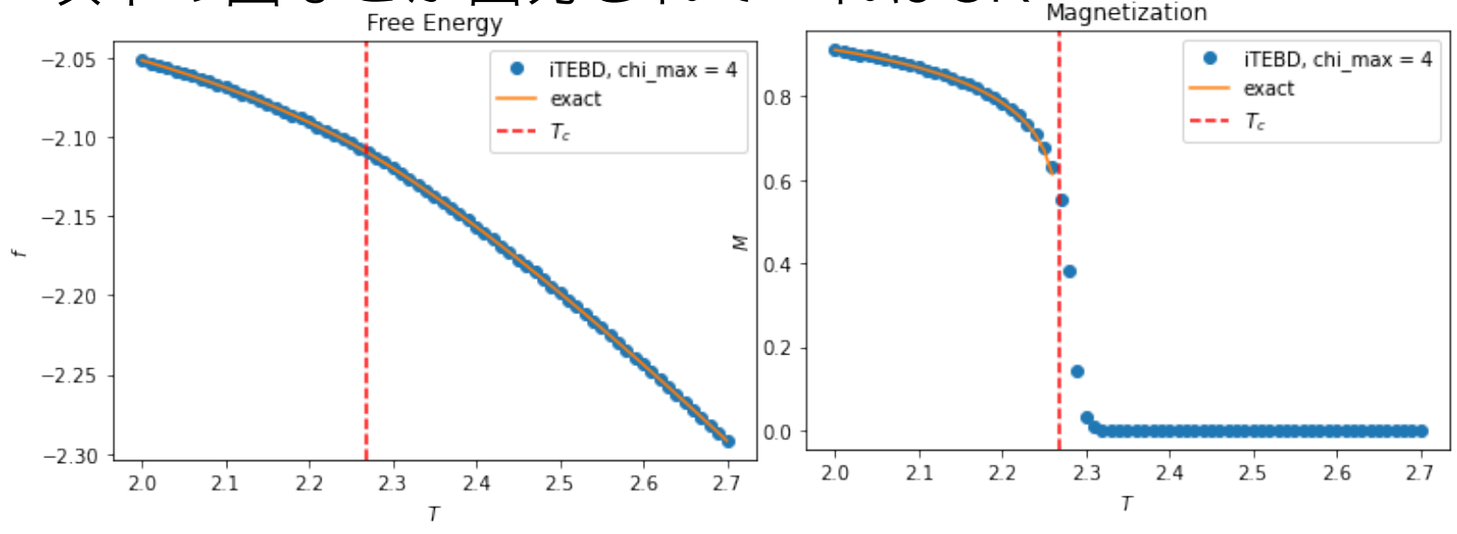
# Ex2-1：行列積状態計算の実行と厳密解との比較

1. Google colabで*Ex2-1.ipynb*を開く
   - これをクリック




2. 「ドライブにコピー」

3. ランタイム/全てのセルを実行
   - 2分くらいで終わるはず
   - 以下の図などが出力されていればOK



# Ex2-1.ipynbのinput

## iTEBDで無限系の正方格子イジング模型の物理量を計算します

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

```
Tc = 2.0/np.log(1.0+np.sqrt(2.0)) ## The critical temp

T_min = 2.0
T_max = 2.7
T_step = 0.01

chi_max = 4 ## bond dimension for iTEBD
max_itr = 200 ## maxmun number of iteration for iTEBD
seed = 13 ## seed for random number generator used for

data_file = "iTEBDdata_ex2-1_chi"+repr(chi_max)+".dat"
### input parameters ###

T_list=np.arange(T_min,T_max,T_step)
```

厳密なTc（変更しない）

*T_min, T_max, T_step*: 温度

*chi_max*：低ランク近似のパラメタ（ボンド次元）

*max_itr*：転送行列を最大何回かけるか

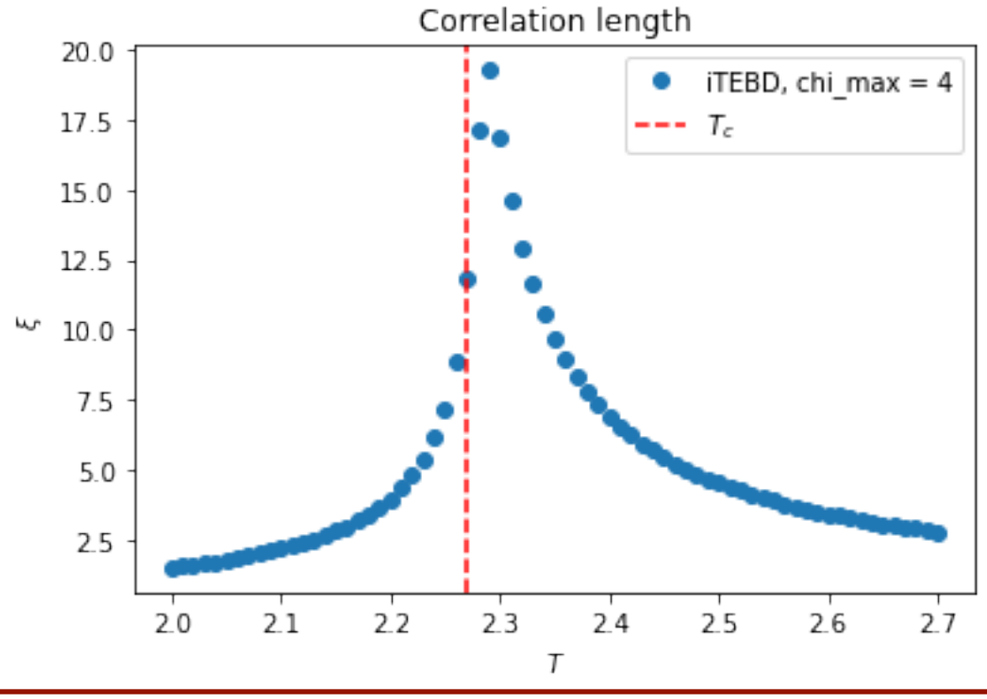
*seed*：*iMPS*の初期テンソルを作る乱数のタネ

# Ex2-1.ipynbのoutput

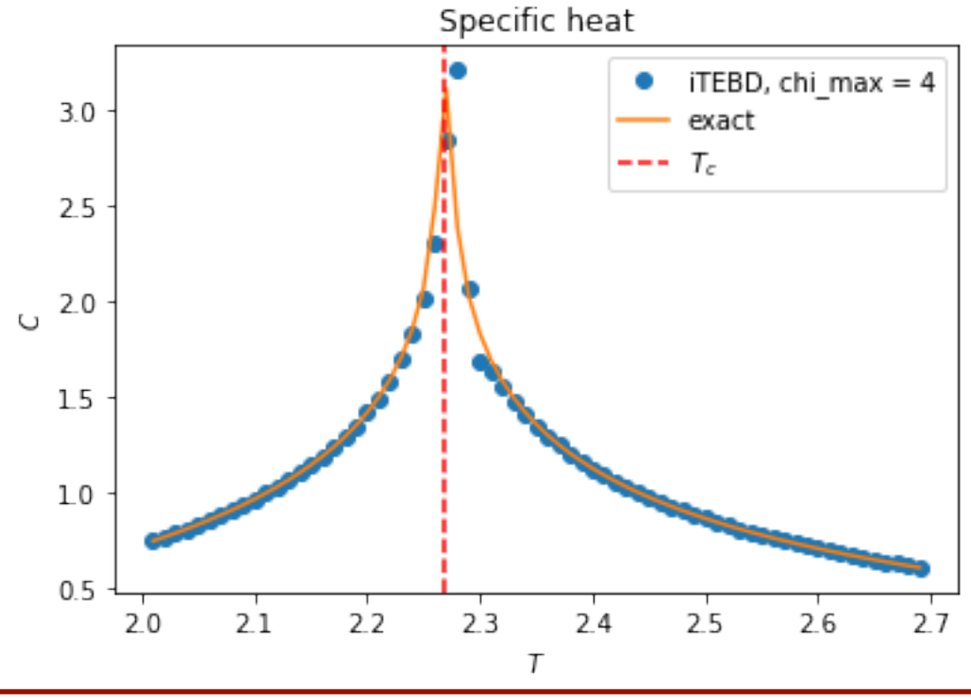
**iTEBDによる自由エネルギー、磁化、相関長**

```
## calculation has finised at 45 steps
T, free_energy_density, exact value = 2.0 -2.0515856173269604 -2.051585625389835
T, magnetization, exact value = 2.0 -0.9113203665311507 0.911319377877496
## calculate correlation length
T, correlation lenght 2.0 1.53226208078651
## calculation has finised at 47 steps
T, free_energy_density, exact value = 2.01 -2.053133972654274 -2.053133982646481
T, magnetization, exact value = 2.01 -0.9079491801051529 0.9079479211094439
```

相関長




差分近似で求めた比熱

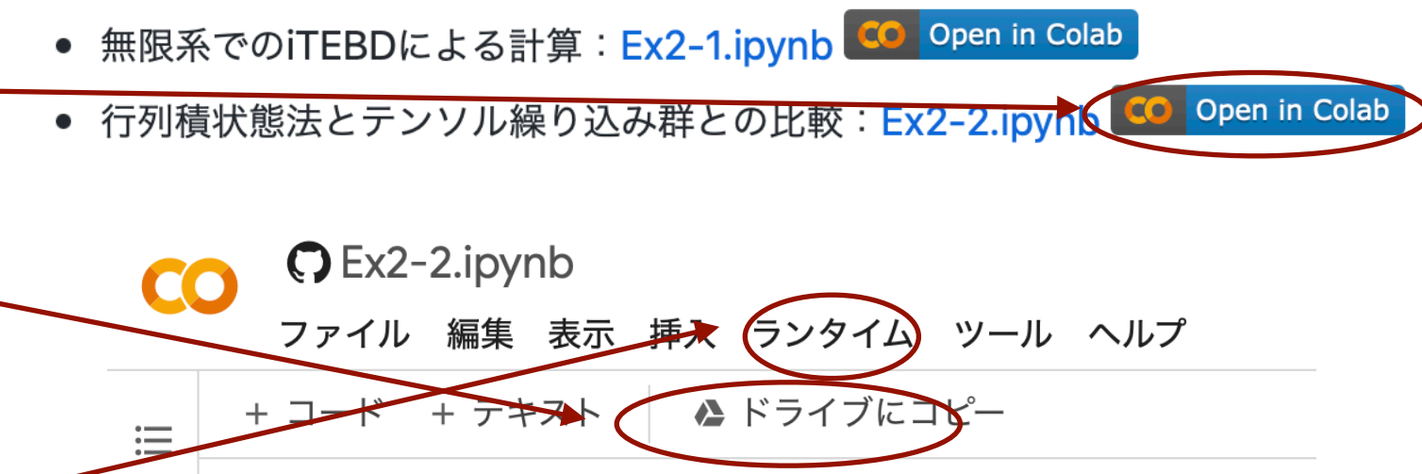


# 実習2-1

- *chi_max*を変えて、計算時間の変化を確認する
  - （補足情報）iTEBDにおけるSVDとテンソルの縮約の計算コストは$O(\chi^3)$
  - （補足情報）計算が収束するまでの回数は温度に依存する
- *chi_max*を変えて、物理量の精度を確認する
  - 磁化の立ち上がりの位置と厳密なTcとの関係
  - Tc近傍の相関長の大きさ
  - ...

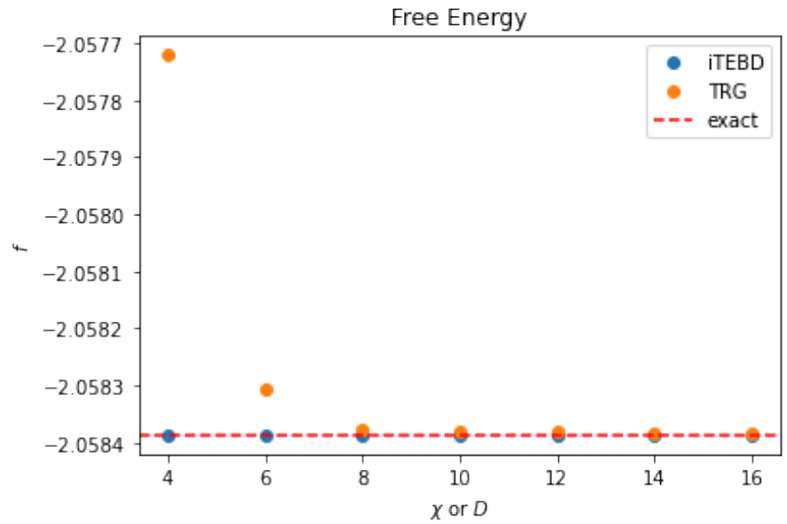
# Ex2-2：行列積状態とTRGの比較

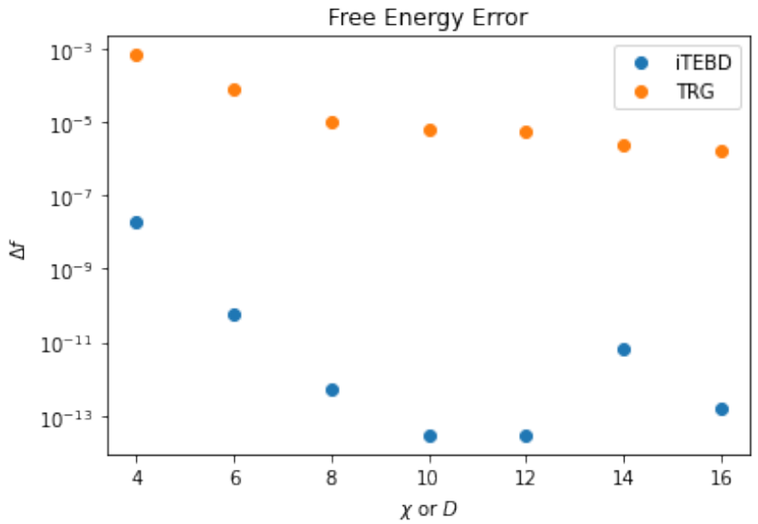
1. Google colabで*Ex2-1.ipynb*を開く
   - これをクリック




2. 「ドライブにコピー」


3. ランタイム/全てのセルを実行
   - 1分くらいで終わるはず
   - 以下の図などが出力されていればOK



# Ex2-2.ipynbのinput

## iTEBDで無限系の正方格子イジング模型の物理量を計算します

2番目のセルでインプットパラメタを変更できます（変更したら"すべてのセルを実行"）

```
Tc = 2.0/np.log(1.0+np.sqrt(2.0)) ## The critical t

T = 0.9 * Tc ## Temperature
```

```
### for iTEBD ###
chi_max_list = [4, 6, 8, 10, 12, 14, 16] ## bond di
max_itr = 200 ## maxmun number of iteration for iTE
seed = 13 ## seed for random number generator used

### for TRG ###
n = 20 ## L = 2^n. Note that n must be sufficiently
D_list = [4, 6, 8, 10, 12, 14, 16] ## bond dimensio

L = 2**n
TRG_step = 2*n -1

## output files
data_file_iTEBD = "iTEBDdata_ex2-2.dat" ## Simulati
data_file = "trgdata_ex2-2_n"+repr(n)+".dat" ## Sim
```

厳密なTc（変更しない）

*T*：温度

*chi_max_list*：iTEBDのボンド次元リスト

*max_itr*：転送行列を最大何回かけるか

*seed*：*iMPS*の初期テンソルを作る乱数のタネ

$L=2^n$: システムサイズ。nで指定

*無限系と比較するので大きくする

*D_list*：TRGのボンド次元リスト

# Ex2-2.ipynbのoutput

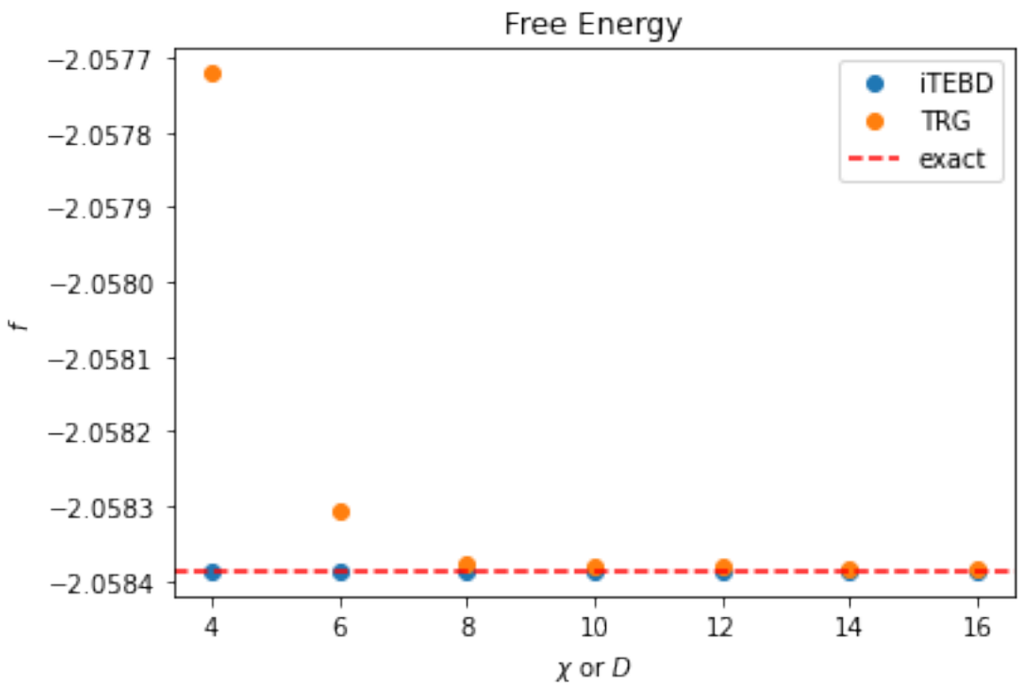
**iTEBDによる自由エネルギー、磁化、相関長**

```
## calculation has finised at 53 steps
T, free_energy_density, exact value = 2.04226678279172 -2.058386743887583 -2.0583867640251055
T, magnetization, exact value = 2.04226678279172 -0.895940335470425 0.8959375370799653
## calculate correlation length
T, correlation lenght 2.04226678279172 1.7597004266052616
```

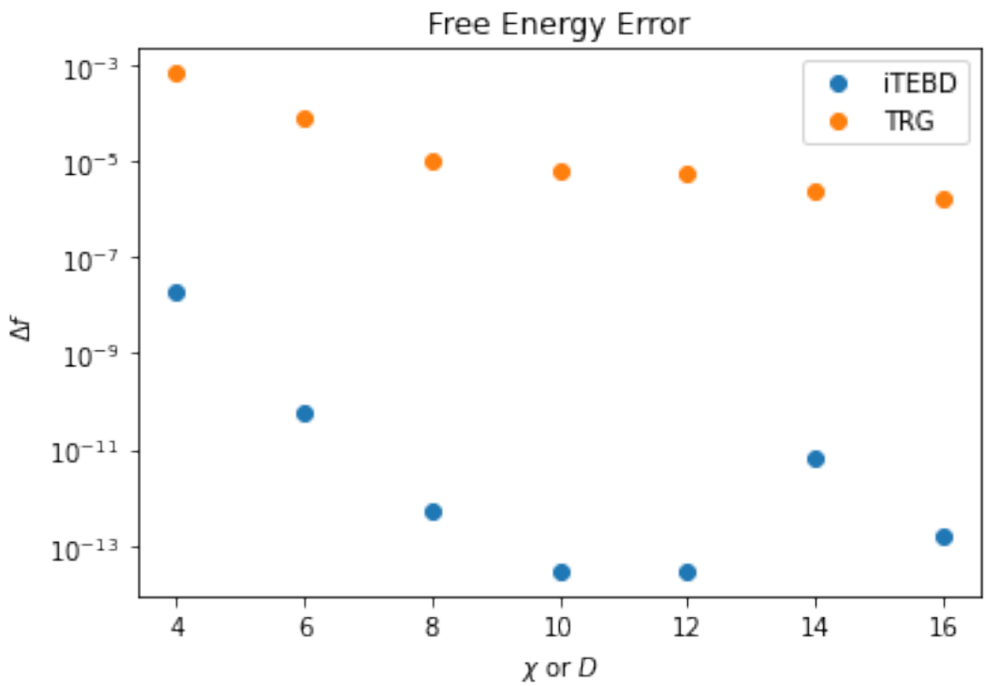
**TRGによる自由エネルギー**

```
T, free_energy_density = 2.04226678279172 -2.057720934733894
T, free_energy_density = 2.04226678279172 -2.0583075284209564
T, free_energy_density = 2.04226678279172 -2.0583767083674065
```

自由エネルギーの比較




自由エネルギーの誤差の比較



# 実習2-1

- $T$を変えて、自由エネルギーの精度を確認する
  - Tc に近づくとどうなるか？
  - 必要に応じで、*chi_max_list, D_list*を変更する
  - 計算時間と精度の関係を考えると、自由エネルギーの計算には、iTEBDとTRGどちらが効率が良いだろうか？

# まとめ

- テンソルネットワーク（TN）は計算科学のいろいろな場面に現れる
  - 問題自体がTN形式で表現される。近似としてTN形式が現れる。
- TNの基本的な計算は、縮約（行列積）、低ランク近似（SVD）、および、固有値問題
  - 応用例：テンソル繰り込み群、固有値問題（行列積状態）
- 今回扱わなかった話題
  - 量子多体問題への応用
  - 量子回路シミュレーション、量子計算
  - 機械学習、データ科学、…

# 参考文献


- テンソルネットワーク法解説記事
  - 数理科学 2022年2月号「特集：テンソルネットワーク法の進展」、サイエンス社
  - 数理科学 2022年11月号の一部「量子多体系とテンソルネットワーク」大久保毅、サイエンス社
  - 「テンソルネットワーク形式の進展と応用」西野友年，大久保毅，日本物理学会誌2017年10月号（https://www.jstage.jst.go.jp/article/butsuri/72/10/72_702/_article/-char/ja/）
  - 「テンソルネットワークによる情報圧縮とフラストレート磁性体への応用」大久保毅、物性研究 Vol. 7, No. 2 （物性若手夏の学校の講義テキスト）（http://mercury.yukawa.kyoto-u.ac.jp/~bussei.kenkyu/archives/category/2018/vol07-2）
- テンソルネットワーク法テキスト
  - 「テンソルネットワークの基礎と応用　統計物理・量子情報・機械学習」西野友年、サイエンス社 SGCライブラリ168 (2021).
- テンソルネットワーク法による数値計算の（お勧め）Review
  - R. Orús, “A practical introduction to tensor networks: Matrix product states and projected entangled pair states”, Annal. Phys. **349**, 117 (2014).