格子スピン模型の計算科学（実習）

理学系研究科　物理学専攻　大久保　毅

e-mail: t-okubo@phys.s.u-tokyo.ac.jp

今日の流れ

- 実習の準備
  - 端末へのログイン、ターミナルの準備
  - 実習に使うコンテンツのダウンロードなど
- 実習１：ALPSを使ったモンテカルロシミュレーション
  （ALPSチュートリアルのmc-09-snapshotの一部）
  - 正方格子イジング模型の物理量計算
  - スピンスナップショット可視化（Paraview）
- 実習２：テンソルネットワーク繰り込み
    - 正方格子イジング模型の自由エネルギー計算
    - モンテカルロ法との比較
- レポート課題の説明
- もっと詳しく学びたい方へ

実習の準備１：環境の準備

- 端末を立ち上げて、Mac環境でログイン
    - 実習でParaviewがMac環境にしか入っていないため
- このスライドのダウンロード
    - **ITC-LMS** (https://itc-lms.ecc.u-tokyo.ac.jp/portal/login)
    - 上記のサイトの「授業スライド」から、「計算科学概論0618.pdf」というファイルをダウンロード
- ターミナルを立ち上げる
- pythonの環境をpython2.7にする

  pyenv shell anaconda-4.0.0

  （この設定は、ターミナルを閉じると消えます。ターミナルを立ち上げる毎に再度設定してください）

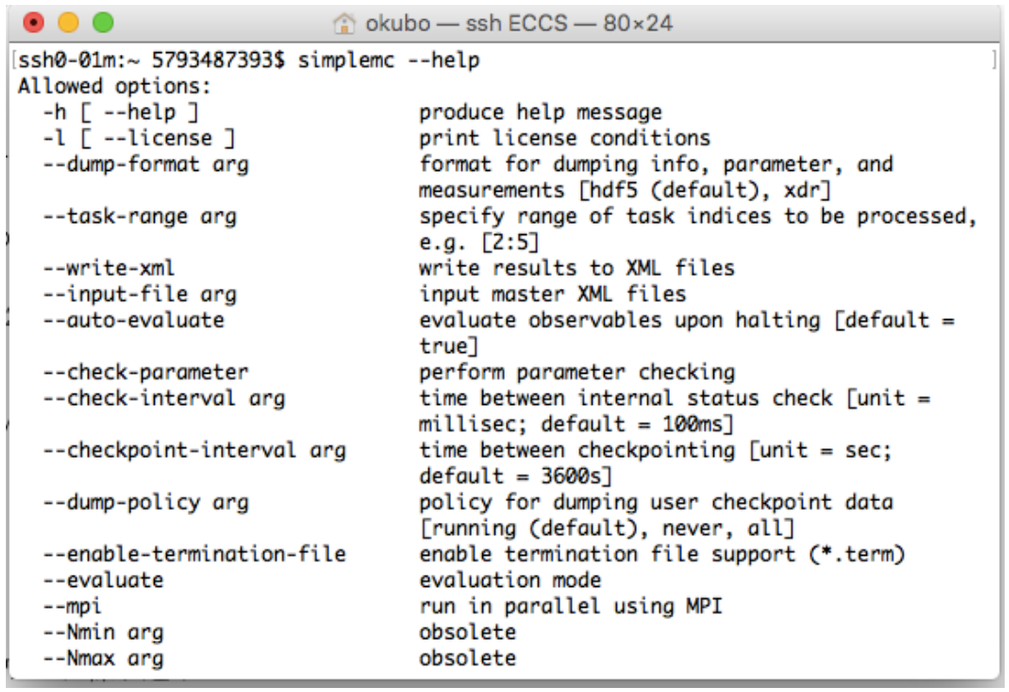
実習の準備２：実習アプリの準備

- **ALPSの準備（以下のファイルはECCSのiMac専用です）**
  - **ITC-LMS** https://itc-lms.ecc.u-tokyo.ac.jp/portal/login または
  - **Dropbox** https://dl.dropboxusercontent.com/s/av4o8ljkbfj0ok3/alps-20160816.zip
    から "alps-20160816.zip"をダウンロードして解凍（おそらく、自動で解凍される）
    （解凍されない場合はダブルクリック）
  - できたフォルダをhomeに移動

    cd
    mv Downloads/alps-20160816 .
  - 設定ファイルの実行

    . alps-20160816/bin/alpsvars.sh
    - "alps-201608..." の前の". " を忘れない
    - ターミナルを開くたびに再設定が必要
  - 実行のチェック

    simplemc --help



- **チュートリアルファイルの準備**
  - **ITC-LMS** https://itc-lms.ecc.u-tokyo.ac.jp/portal/login
    から「Tutorial20180618.zip」をダウンロードして解凍
  - できたフォルダ“Tutorial20180618”をhomeに移動

    cd
    mv Downloads/Tutorial20180618 .

実習１：

ALPSによるモンテカルロシミュレーション

実習１：ALPSによるモンテカルロシミュレーション

ALPS（Applications and Libraries for Physical Simulation)

- 種々の格子模型シミュレーションのためのライブラリ・アプリケーション群
- 格子スピン模型、ハバード模型（高温超伝導等の模型）、近藤格子模型（重い電子系の模型）などに対応
- 様々な種類の解法が準備されている
  - 古典・量子モンテカルロ法、厳密対角化法、密度行列繰り込み群法（DMRG）、動的平均場理論（DMFT）など
  - 解きたい問題、興味のある現象に合わせて最適な解法を選べる
  - 最先端の研究に使える性能と信頼性

ALPS Wiki http://alps.comp-phys.org/mediawiki/index.php/Main_Page

ALPSを使った研究論文

(2015年以降のものを適当に抜粋)

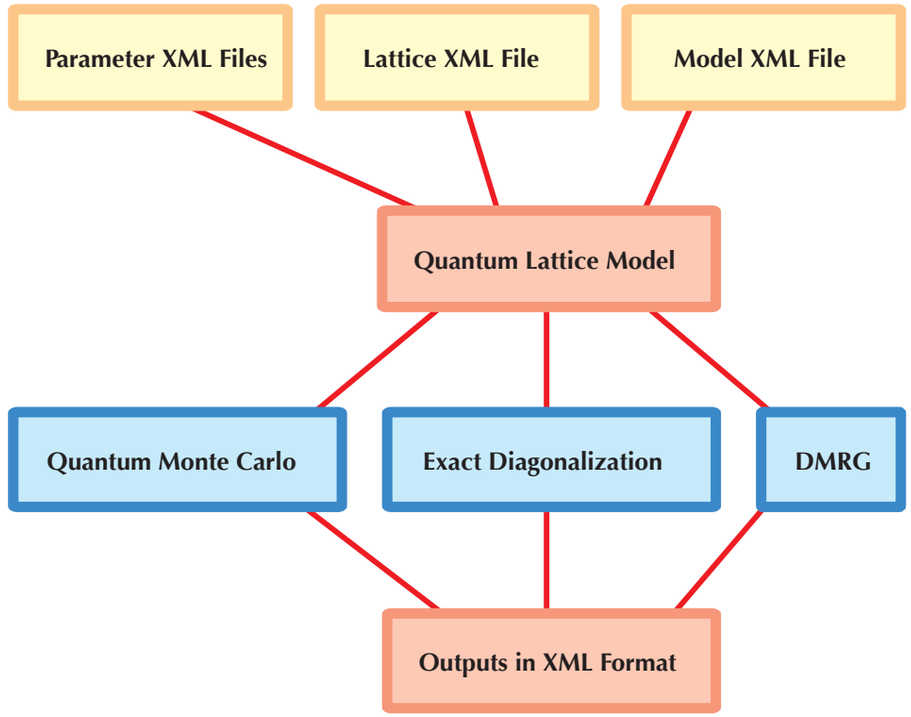
- Phase transition of ultracold atoms immersed in a BEC vortex lattice
- Entanglement entropy and topological order in resonating valence-bond quantum spin liquids
- First-order topological phase transition of the Haldane-Hubbard model
- DMFT Study for Valence Fluctuations in the Extended Periodic Anderson Model
- Static and dynamical spin correlations of the S =1/2 random-bond antiferromagnetic Heisenberg model on the triangular and kagome lattices
- Transport properties for a quantum dot coupled to normal leads with a pseudogap
- Magnetic structure and Dzyaloshinskii-Moriya interaction in the S =1/2 helical-honeycomb antiferromagnet α -$Cu_2V_2O_7$
- Mott transition in the triangular lattice Hubbard model: A dynamical cluster approximation study
- SU (N) Heisenberg model with multicolumn representations
- Superconductivity in the two-band Hubbard model
- Local Electron Correlations in a Two-Dimensional Hubbard Model on the Penrose Lattice

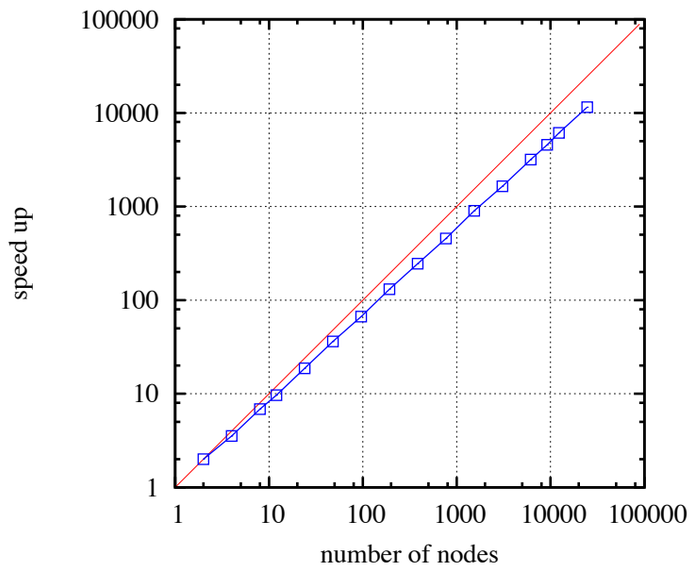
詳細： http://alps.comp-phys.org/mediawiki/index.php/PapersTalks

ALPS の機能

＊MateriApps のハンズオン資料から借用

- 入出力支援
  - 格子構造, 模型は XML を用いて柔軟に指定
  - 全てのソルバーに共通した入出力形式
  - Python インターフェースを用意
    - Python から直接実行、グラフを作成
- 並列化
  - パラメータ並列のための並列化スケジューラ
  - 量子モンテカルロソルバ (looper)
    - 京で20,000ノードまで良好なスケーリング
- 競合するアプリケーション：「なし」？





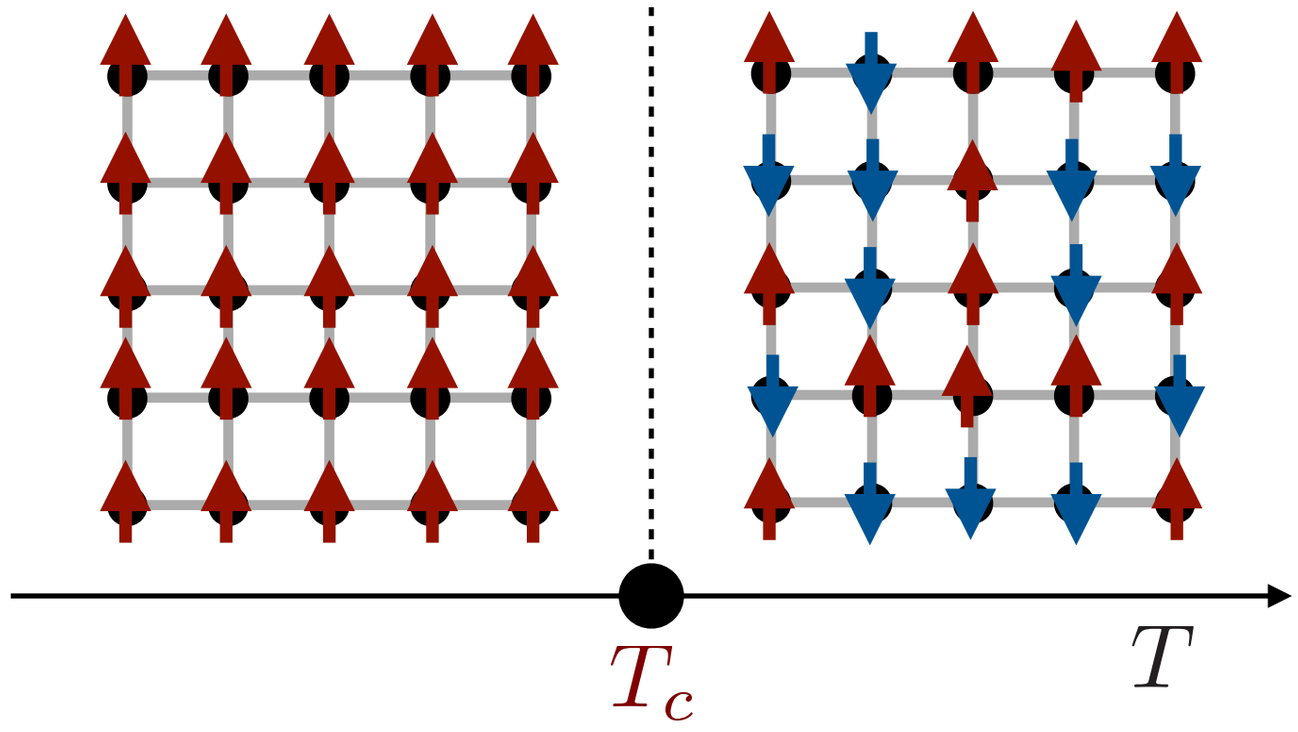
ALPSの実習（１）：平衡シミュレーション

- 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、平衡状態の物理量を見る

ALPSの実習（２）：可視化

- 古典格子スピン模型のスナップショット（スピン状態）の可視化
  - 古典モンテカルロ法（メトロポリス法）ではスピン状態$\{S_i\}$をサンプリング

1. イジング模型のスナップショット

   - 強磁性状態、常磁性状態、臨界点でのスピン状態を実際に見る

2. （今日はやらない）正方格子古典XY模型に現れる渦

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \left(S_i^x S_j^x + S_i^y S_j^y \right) \qquad (S_i^x)^2 + (S_i^y)^2 = 1$$

   - 有限温度では自発的対称性の破れはない
   - しかし、$T = T_c$でスピン渦の解離・凝集による相転移（Kosterlitz-Thouless転移）
     - 低温相：渦が対で存在し、スピン相関がべき的（臨界点と同じ）
     - 高温相：渦は解離し、自由に“動ける”渦が存在
   - この渦の様子を実際に見てみる

正方格子イジング模型のシミュレーション

- **simplemc**: メトロポリス法による基本的な古典モンテカルロ法シミュレーター
- チュートリアルのディレクトリに移動
  - *cd*
  - *cd Tutorial20180618/Tutorial_MC/simplemc*
- パラメータファイルの変換（ALPS用にXMLで書かれたインプットファイルを生成）
  - *parameter2xml parm9a*
- シミュレーションの実行（終了までおよそ5分弱）
  - *simplemc parm9a.in.xml*
- 結果のプロット（比熱、エネルギー、磁化の２乗の期待値がプロットされる）
  - *python plot9a.py*
  - （３枚のグラフ全てを閉じると、pythonスクリプトが終了する）

Tips for error in matplotlib

- サンプルコード"plot9a.py"の実行でエラーが出た場合
  - 日本語の文字コードが原因の可能性があります。以下のコマンドを入力して、文字コードをUTF-8に設定してください。
    - *export LC_ALL=ja_JP.UTF-8*
    - デフォルトでUTF-8になっているはずですが、環境変数*LC_ALL*が正しく設定されていない場合があるようです。
    - この修正はシェルを立ち上げる度にやる必要があります。

パラメタファイルの説明：parm9a

```
LATTICE="square lattice"
J=1
ALGORITHM="ising"
SWEEPS=65536
L=8
{ T=5.0 }
{ T=4.5 }
{ T=4.0 }
{ T=3.5 }
{ T=3.0 }
{ T=2.9 }
{ T=2.8 }
{ T=2.7 }
...
```

格子を指定：正方格子

相互作用を指定：J > 0 強磁性相互作用

モデルを指定：イジングモデル

モンテカルロステップ数を指定

1モンテカルロスッテプ＝全てのスピンが１度反転の試行

＊THERMALIZATION（熱平衡化）

期待値への初期状態依存性を小さくするために、

THRMALIZATION ステップの計算を「空回し」

明示的に指定しなければ、SWEEPSの1/8

格子の大きさを指定：L × L

計算する温度のリスト

- 中括弧{}で囲ったパラメタ：１回の計算のパラメタ
  - 中括弧の数だけ、シミュレーションが実行される
- 中括弧で囲ってないパラメタ：
  それ以降の全ての計算に共通のパラメタ

プロットファイルの説明：plot9a.py

Pythonスクリプト

計算データからの結果の抽出：**pyalps.loadMeasurements**

X軸Y軸の設定とサイズ毎の整理：**pyalps.collectXY**

結果のプロット：**matplotlib**

パラメタファイルの名前
必要に応じて変更する

```
data = pyalps.loadMeasurements(pyalps.getResultFiles(prefix='parm9a'),
    ['Specific Heat', 'Magnetization Density^2', 'Energy Density'])
for item in pyalps.flatten(data):
    item.props['L'] = int(item.props['L'])

magnetization2 = pyalps.collectXY(data, x='T', y='Magnetization Density^2', foreach=['L'])
magnetization2.sort(key=lambda item: item.props['L'])

pyplot.figure()
alpsplot.plot(magnetization2)
pyplot.xlabel('Temperture $T$')
pyplot.ylabel('Magnetization Density Squared $m^2$')
pyplot.legend(loc='best')
```

プロットする物理量を
追加・変更する場合は
ここに追加して、
他を参考に以下の行も
コピペ・修正する

正方格子イジング模型：計算条件を変える

1. パラメタファイルをコピー

   cp parm9a parm9a_2

2. parm9a_2を書き換えて、パラメタを変える

   ・ （例）SWEEPを増やす、温度を増やす・変える

3. 実行する（前と同様。ただし、ファイル名に注意）

4. プロットする

   ・ plot9a.pyの読み込むファイル名の部分を変更する必要あり

(optional)モンテカルロ法の並列化

- モンテカルロ法の並列化
  - アルゴリズムの並列化（担当する領域を分割する等）
  - 独立なサンプリングの単純並列
    - パラメタを変えたもの or 同じパラメタで乱数が違う

ALPS：

- 古典モンテカルロ法のアプリ、simplemcとspinmcは、後者にのみ対応
- 量子モンテカルロ法のアプリ、loop は前者にも対応している

Reedbush-Uでの実行

残念なお知らせ：

Reedbush-Uに入っているalpsのバージョンが古いため、

"simplemc"がなく、"spinmc"しか使えない

両者の主な違いは

- 入力ファイルのパラメタ
- 物理量の評価方法
- 物理量の名前
- spinmcはクラスターアルゴリズムが使える

今回は、別にコンパイルしたものを使います。

Reedbushに最初からインストールされているalpsを使う場合は、

spinmcの使い方について後の方のスライドを参考にしてください

simplemcによるシミュレーション@Reedbush-U

- ReedbushにsshでログインしてCD、/lustre に移動
  - *cdw*
- コンパイル済みのalpsをコピーして解凍、設定ファルを実行
  - *cp /lustre/gt03/share/alps-20160816.zip .*
  - *unzip alps-20160816.zip*
  - *. alps-20160816/bin/alpsvars.sh*
- チュートリアルのファイルをコピーして解凍
  - *cp /lustre/gt03/share/Tutorial20180618.zip .*
  - *unzip Tutorial20180618.zip*
- *simplemc*のディレクトリに移動
  - cd *Tutorial20180618/Tutorial_MC/simplemc*
- パラメタファイルの変換
  - *parameter2xml parm9a_sc*
- バッチファイルをスパコンにsubmit（中で、計算の実行とevaluateまで行う）
  - qsub sub_parm9a_sc.sh
- 結果のプロット（比熱、エネルギー、磁化の２乗の期待値がプロットされる）
  - *python plot9a_sc.py*
  - *ECCSに結果のファイルコピーして、そちらでプロットでも良い*
- *parm9a_sc2*という入力ファイルも準備しています。

入力ファイルの違い：parm9a_sc, parm9a_sc2

parm9a_sc

```
LATTICE="square lattice"
J=1
ALGORITHM="ising"
SWEEPS=655360

L=8
{ T=5.0 }
```

モンテカルロステップ数を10倍

単純には10倍の時間かかるが...

parm9a_sc2

```
LATTICE="square lattice"
J=1
ALGORITHM="ising"
SWEEPS=655360
NUM_CLONE=2

L=8
{ T=5.0 }
```

NUM_CLONE：

同パラメタでの独立した計算を
NUM_CLONE回やる。結果は
自動的に統合して解析される

＊「独立」なので容易に並列化できる

ジョブスクリプト：sub_parm9a_sc.sh

```
#!/bin/sh
#PBS -q u-lecture3
#PBS -l select=1:mpiprocs=36:ompthreads=1
#PBS -W group_list=gt03
#PBS -l walltime=00:10:00
#PBS -N simplemc_parm9a_sc

cd ${PBS_O_WORKDIR}
. /etc/profile.d/modules.sh

echo "Current directory is [$(pwd)]."
#set alps environment
echo "[$(date)] set alps."
. ${HOME}/alps-20160816/bin/alpsvars.sh

echo "[$(date)] start main simulations."
mpirun simplemc --mpi parm9a_sc.in.xml

echo "[$(date)] end simulation."
```

- 講義中はu-lecture3
- 講義後はu-lecture

1ノード、 36MPIでの並列

＊NUM_CLONE=2のときは、select=2として、２ノードになっています。

設定ファイルの読み込み

- mpirun で実行
- --mpiのオプション

並列計算の注目点：

計算時間はどうか？

36並列しているので、10倍のモンテカルロステップ

でも、ECCS端末より遅くないはず。

MCステップ数増大の影響？（NUM_CLONEも含めて）

統計誤差が小さくなることが期待されます。

グラフで見ると、誤差棒が短くなっているはず。

spinmcによるシミュレーション@ECCS

- **spinmc**: メトロポリス法に加えてクラスターアルゴリズムも使える古典モンテカルロ法シミュレーター
- チュートリアルのディレクトリに移動
  - *cd*
  - *cd Tutorial20180618/Tutorial_MC/spinmc*
- パラメータファイルの変換（*ALPS*用に*XML*で書かれたインプットファイルを生成）
  - *parameter2xml parm9a*
- シミュレーションの実行
  - *spinmc --Tmin 5 parm9a.in.xml*
    “--Tmin n” はシミュレーションが終わったかをチェックする最小の時間間隔を設定する。
    I今の例では、初期値 n=60 （*60*秒）は長すぎるので --Tmin 5 or --Tmin 1をお勧めする
- 物理量を評価する （**simplemc**と違って、幾つかの物理量についは明示的に物理量評価を実行する必要がある）
  - *spinmc_evaluate parm9a.task*.out.xml*
- 結果のプロット（比熱、エネルギー、磁化の２乗の期待値がプロットされる）
  - *python plot9a.py*
  - （３枚のグラフ全てを閉じると、pythonスクリプトが終了する）

Explanation of parameter file: parm9a

```
LATTICE="square lattice"
J=1
MODEL="Ising"
UPDATE="local"
THERMALIZATION=8192
SWEEPS=65536
L=8
{ T=5.0 }
{ T=4.5 }
{ T=4.0 }
{ T=3.5 }
{ T=3.0 }
{ T=2.9 }
...
```

格子を指定：正方格子

相互作用を指定：J > 0 強磁性相互作用

モデルを指定：イジングモデル

アルゴリズム： “local” (メトロポリス法) or “cluster”

モンテカルロステップ数を指定

1モンテカルロスッテプ＝全てのスピンが１度反転の試行

THERMALIZATION（熱平衡化）

期待値への初期状態依存性を小さくするために、

THRMALIZATION ステップの計算を「空回し」

spinmcの場合は明示的に書かなくてはならない

格子の大きさを指定：L × L

計算する温度のリスト

- 中括弧{}で囲ったパラメタ：１回の計算のパラメタ
  - 中括弧の数だけ、シミュレーションが実行される
- 中括弧で囲ってないパラメタ：
  それ以降の全ての計算に共通のパラメタ

プロットファイルの説明：plot9a.py

Pythonスクリプト

計算データからの結果の抽出：**pyalps.loadMeasurements**

X軸Y軸の設定とサイズ毎の整理：**pyalps.collectXY**

結果のプロット：**matplotlib**

パラメタファイルの名前
必要に応じて変更する

```
data = pyalps.loadMeasurements(pyalps.getResultFiles(prefix='parm9a'),
    ['Specific Heat', 'Magnetization^2', 'Energy Density'])
for item in pyalps.flatten(data):
    item.props['L'] = int(item.props['L'])

magnetization2 = pyalps.collectXY(data, x='T', y='Magnetization^2', foreach=['L'])
magnetization2.sort(key=lambda item: item.props['L'])

pyplot.figure()
alpsplot.plot(magnetization2)
pyplot.xlabel('Temperture $T$')
pyplot.ylabel('Magnetization Density Squared $m^2$')
pyplot.legend(loc='best')
```

プロットする物理量を
追加・変更する場合は
ここに追加して、
他を参考に以下の行も
コピペ・修正する

spinmcによるシミュレーション@Reedbush-U

- Reedbushにsshでログインして、/lustre に移動
  - *cdw*
- チュートリアルのファイルをコピーして解凍
  - *cp /lustre/gt03/share/Tutorial20180618.zip .*
  - *unzip Tutorial20180618.zip*
- *spinmc*のディレクトリに移動
  - cd *Tutorial20180618/Tutorial_MC/spinmc*
- アルプスのモジュールをload
  - *module load alps/2.1.1-r6176*
- パラメタファイルの変換
  - *parameter2xml parm9a_sc*
- バッチファイルをスパコンにsubmit（中で、計算の実行とevaluateまで行う）
  - *qsub sub_parm9a_sc.sh*
- 結果のプロット（比熱、エネルギー、磁化の２乗の期待値がプロットされる）
  - *alpspython plot9a_sc.py*
  - *"python"*ではなく、*"alpspython"*になっていることに注意
  - *ECCS*に結果のファイルコピーして、そちらでプロットでも良い

入力ファイルの違い：parm9a_sc

parm9a_sc

```
LATTICE="square lattice"
J=1
MODEL="Ising"
UPDATE="local"

THERMALIZATION=81920
SWEEPS=655360

L=8
{ T=5.0 }
{ T=4.5 }
{ T=4.0 }
```

モンテカルロステップ数を10倍
（併せて、Thermalzationも10倍）

→ 単純には10倍の時間かかるが...

＊NUM_CLONEはspinmcでは使えません

ジョブスクリプト：sub_parm9a_sc.sh

```
#!/bin/sh
#PBS -q u-lecture3
#PBS -l select=1:mpiprocs=36:ompthreads=1
#PBS -W group_list=gt03
#PBS -l walltime=00:10:00
#PBS -N spinmc_parm9a_sc

cd ${PBS_O_WORKDIR}
. /etc/profile.d/modules.sh

echo "Current directory is [$(pwd)]."

#load alps module
echo "[$(date)] load modules."
module load alps/2.1.1-r6176

echo "[$(date)] start main simulations."
mpirun spinmc --mpi --Tmin 5 parm9a_sc.in.xml

echo "[$(date)] start evaluations."
spinmc_evaluate parm9a_sc.task*.out.xml

echo "[$(date)] end simulation."
```

- 講義中はu-lecture3
- 講義後はu-lecture

1ノード、 36MPIでの並列

ALPSモジュールのload

- mpirun で実行
- --mpiのオプション

evaluateは並列化できない

基本課題１：

- 正方格子イジング模型のシミュレーション
  - 比熱は臨界点T_cで発散することが期待されるが、有限サイズ（L）では、
    発散せずピーク位置 $T_p(L)$ もT_cとは異なる。
    この$T_p(L)$はLが十分に大きい時、

$$T_p(L) = T_c + cL^{-1/\nu}$$

    と振る舞うことが知られている（有限サイズスケーリング）。
    ここで、νは相関長の臨界指数であり、正の実数である。
    これを念頭に、parm9aを適当に修正して以下の課題に取り組め
    （parm9aのままだと誤差が大きいので、少なくともSWEEPを大きくした方が良いです。）

1. エネルギー、比熱、磁化（の２乗）のグラフを示せ
2. シミュレーション結果から比熱のピーク位置をおよそ読み取れ
3. 比熱のピーク位置と厳密なT_cの関係から未知係数cの符号を推定せよ
   - 答えの正しさではなく、推定の論理を評価します。
4. 物理量（比熱）の値・誤差と、温度・Lとの関係について考察しつつ、
   精度よくT_cを決める手法を提案せよ

発展課題１：

- ３次元立方格子イジング模型のシミュレーションをする
  - パラメタファイルを適切に書き換えて３次元立方格子イジング模型のシミュレーションを行い、
    以下の点を議論せよ（＊LATTICE="simple cubic lattice"と変更することで立方格子になります。）

1. 転移温度T_Cはおよそどれくらいと推定できるか
2. 正方格子イジング模型の場合と比べて、（同じLで）計算時間はどう変化したか
    1. （注）正方格子の場合と同じLでは時間がかかりすぎると思います
3. その他なんでも

Tips

- グラフの上でマウスカーソルを動かすと、右下にカーソル位置の座標が出ます
  - （右クリックが必要かもしれません）
- グラフではなく数値を見たい場合には、データをグラフではなく、テキストで出力するサンプルスクリプトtextout9a.pyを参考に。

  python textout9a.py

  で画面に数値が出力される。画面ではなくファイルに書き出したい場合は、

  python textout9a.py > filename.txt

- （注）モンテカルロ法では、SWEEP数が小さすぎる場合、そもそも熱平衡分布を再現できない。
  パラメタ変える場合には、十分に大きなSWEEP数になっているかの検証が必要

実習１－２：
ALPSによるスピン状態の可視化

スナップショット(スピン状態)の可視化

＊MateriApps のハンズオン資料から借用

- パラメタファイル (parm9b) ⇒
  - SNAPSHOT_INTERVAL が追加
  - SNAPSHOT_INTERVALごとにスナップショットが出力される(*.snap)
- シミュレーションの実行
  - *parameter2xml parm9b*
  - *simplemc parm9b.in.xml*
  - *ls -l parm9b.*.snap*
- スナップショットの変換 (*.snap ⇒ *.vtk)
  - *snap2vtk parm9b.*.snap*
  - *ls -l parm9b.*.vtk*
  - VTK ファイルには格子点の座標とスピン状態(±1)が収められる

```
LATTICE="square lattice"
J=1
ALGORITHM="ising"
SWEEPS=16384
THERMALIZATION=0
SNAPSHOT_INTERVAL=16384
L = 128
{ T = 3.0 }
{ T = 2.3 }
{ T = 2.0 }
```

ParaViewによるスナップショットの可視化

＊MateriApps のハンズオン資料から借用

- *paraview*　**（ECCSでは、"アプリケーション"にparaview.appがある）**
- file ⇒ open ⇒ parm9b.task1.clone1.16384.vtk を選択 ⇒ OK ⇒ Apply
  - filters ⇒ common ⇒ glyph (あるいは地球儀のような形のアイコンをクリック)
  - Glyph Type = Box, X Length = 0.08, Y Length = 0.08, Maximum Number of Points = 20000 ⇒ Apply
- OpenGL window の右上の水平分割アイコンをクリック
- file ⇒ open ⇒ parm9b.task2.clone1.16384.vtk を選択 ⇒ OK ⇒ Apply
  - filters ⇒ common ⇒ glyph (あるいは地球儀のような形のアイコンをクリック)
  - Glyph を同様に設定
- OpenGL window の右上の水平分割アイコンをクリック
- file ⇒ open ⇒ parm9b.task3.clone1.16384.vtk を選択 ⇒ OK ⇒ Apply
  - filters ⇒ common ⇒ glyph (あるいは地球儀のような形のアイコンをクリック)
  - Glyph を同様に設定

ParaView (続き)

＊MateriApps のハンズオン資料から借用

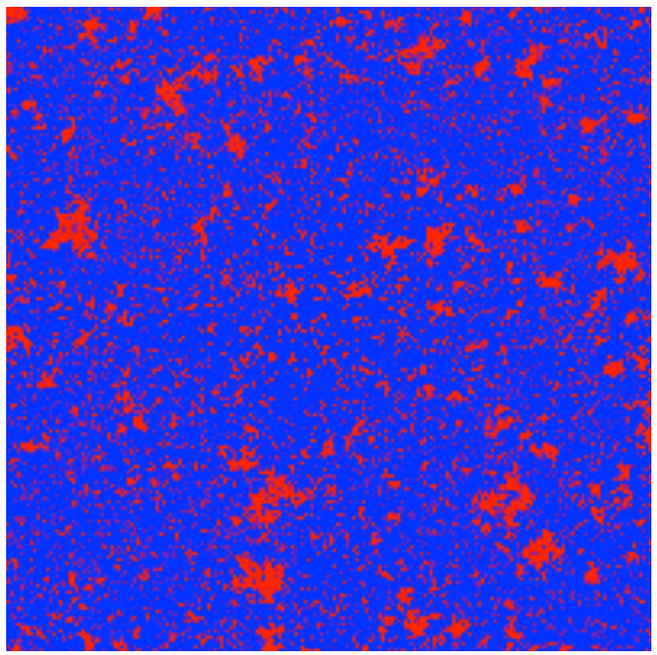
- カメラのリンク
  - ウインドウのサイズが同じになるように調整
  - 真ん中のウインドウをクリック ⇒ Tools ⇒ Add Camera Link ⇒ 左のウインドウをクリック
    - (あるいは 真ん中のウインドウ右クリック ⇒ Link Camera... を選択 ⇒ 左のウインドウをクリック)
  - 同様に右のウインドウも左のウィンドウとリンク

より大規模な系でのスナップショット

＊MateriApps のハンズオン資料から借用

$$F = E - TS$$

T=0.995T_c

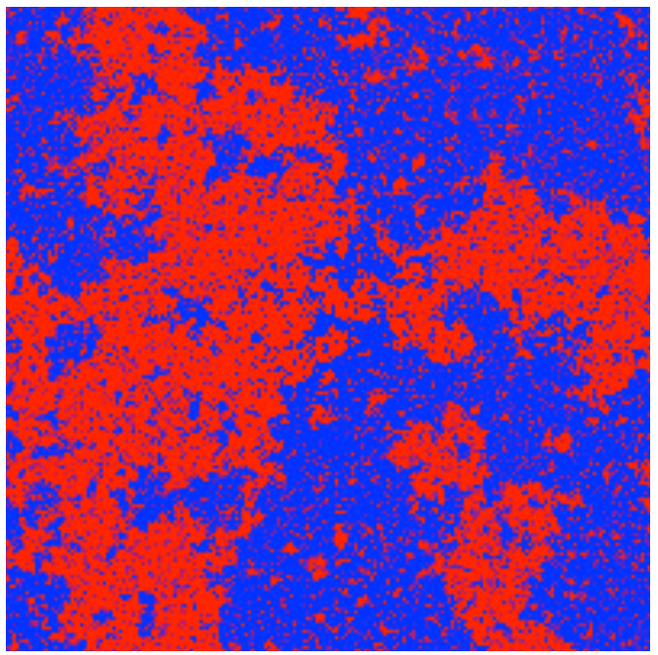


エネルギー利得

秩序状態

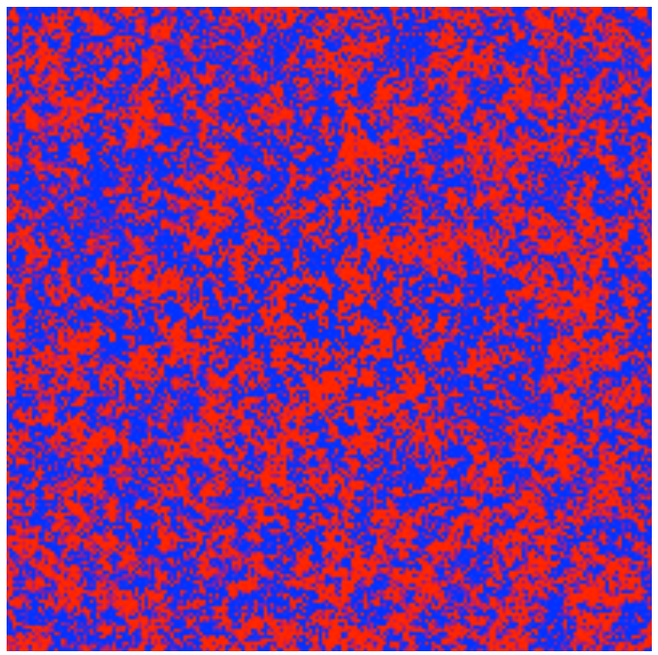
ordered state

T=T_c



臨界点 critical point

T=1.05T_c



エントロピー利得

無秩序状態

disordered state

二次元XY模型のスピン渦の可視化 （おまけ）

＊MateriApps のハンズオン資料から借用

- パラメータファイルの変換・シミュレーションの実行・スナップショットの変換
  - *parameter2xml parm9c*
  - *simplemc parm9c.in.xml*
  - *snap2vtk parm9c.*.snap*
- ParaView による可視化
  - *paraview*
  - file ⇒ open ⇒

    parm9c.task1.clone1.16384.vtk を選択 ⇒ OK
  - properties タブ ⇒ Apply
  - Glyph を追加
  - Glyph Type = Arrow, Tip Radius = 0.2, Shaft Radius = 0.06, Translate = -0.1 0 0, Scale = 0.2 0.2 0.2 -> Apply

```
LATTICE="square lattice"
J=1
ALGORITHM="xy"
SWEEPS=16384
THERMALIZATION=0
SNAPSHOT_INTERVAL=16384
L = 64
{ T = 0.01 }
```

おまけ：

- サイズや温度を変えて、渦の様子をみてみよう
  - 正方格子XY模型のKosterlitz-Thouless転移温度は $T_c/J \simeq 0.89$
  - この温度を境に、何か変化が見えるか？

実習２：
テンソルネットワーク繰り込み群

実習２：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１で行ったモンテカルロシミュレーションとの比較

TRGの実習（１）：自由エネルギーの計算

- TensorNetwork_TRG.py：
  テンソルネットワーク繰り込みを用いた（正方格子イジング模型の）自由エネルギー計算アプリ
- チュートリアルのディレクトリに移動
  - *cd*
  - *cd Tutorial20180618/Tutorial_TRG*
- *L=8, T=2.0*の自由エネルギー計算のテスト
  - *python TensorNetwork_TRG.py -n 3 -T 2.0*
    - *n: L=2n*でサイズを指定　*n=3→L=8*
    - **出力:T, free_energy_density= 2.0 -2.07271839193**
    - ファイル：freee_energy_L8_D4.dat ができる（D: SVDの打ち切りの大きさ、初期設定は４）
- スクリプトによる複数計算の実行
  - *python TRG_tutorial1-1.py*
  - free_energy_L2_D4.dat などができる
- 結果のプロット（TRGで計算した自由エネルギーと厳密解がプロットされる）
  - *python Plot_TRG_1-1.py*
  - 厳密解の計算は藤堂先生作成のコードを利用しました：*https://github.com/todo-group/exact*

TensorNetwork_TRGの使い方

ヘルプメッセージの出力

python TensorNetwork_TRG -h

```
usage: TensorNetwork_TRG.py [-h] [-D D] [-n n] [-T T] [--step] [--
energy]
                            [-Tmin Tmin] [-Tmax Tmax] [-Tstep
Tstep]

Tensor Network Renormalization for Square lattice Ising model

optional arguments:
  -h, --help    show this help message and exit
  -D D          set bond dimension D for TRG
  -n n          set size n representing L=2^n
  -T T          set Temperature
  --step        Perform multi temperature calculation
  --energy      Calculate energy density by using impurity tensor
  -Tmin Tmin    set minimum temperature for step calculation
  -Tmax Tmax    set maximum temperature for step calculation
  -Tstep Tstep  set temperature increments for step calculation
```

パラメタ

D：SVDで近似する際のランク

n：$L=2^n$の形のシステムサイズ

T：温度

flag:

-- step:複数の温度での計算

-Tmin：最低温度

-Tmax：最高温度

-Tstep：温度きざみ

-- energy:エネルギーも計算する

（どう計算してるでしょうか？）

＊Pythonスクリプトから関数としても実行可能

TensorNetwok_TRG.Calculate_TRG(D, n, T, Tmin, Tmax, Tstep, Energy_flag, Step_flag)

プロットファイルの説明：Plot_TRG1-1.py

```
# coding:utf-8
import matplotlib.pyplot as plt
import TensorNetwork_TRG as TN

## read data
T_L2,f_L2 = TN.read_free_energy(“free_energy_L2_D4.dat")
T_L4,f_L4 = TN.read_free_energy(“free_energy_L4_D4.dat”)
…

# read exact data
T_L2e,f_L2e = TN.read_free_energy(“exact/outputs/free_energy_exact_L2.dat")
T_L4e,f_L4e = TN.read_free_energy(“exact/outputs/free_energy_exact_L4.dat")
…

## plot data
fig1,ax1= plt.subplots()
ax1.set_xlabel("T")
ax1.set_ylabel("f")
ax1.set_title("Free energy density of square lattice Ising model")
ax1.plot(T_L2e, f_L2e, "r",label = "L=2: Exact")
ax1.plot(T_L4e,f_L4e, "g",label = "L=4: Exact")
…
ax1.plot(T_L2, f_L2, "ro",label = "L=2")
ax1.plot(T_L4,f_L4, "go",label = "L=4")
…

ax1.legend(loc="lower left")
plt.show()
```

計算データ

厳密解データ

の読み込み

TensorNetwork_TRG.read_free_energy(file_name)

Matplotlibでグラフを描画

基本課題２：

- TRGで計算した自由エネルギーと厳密解を比較する
  - TRGでは近似的に分配関数（自由エネルギー）を計算するため、厳密な自由エネルギーとはずれが生じる。
    1. 温度をいくつか選び、TRGの計算値と厳密解とのズレの大きさとシステムサイズとの関係を調べよ
       （計算値はグラフから読み取るか、出力ファイルから数字を直接見る）
    2. （温度、サイズ）の組みをいくつか選んで低ランク近似の打ち切り次元（D）の大きさを変えた複数のシミュレーションを実行し、厳密解とのズレの変化を調べよ
- Tips
  - Plot_TRG_1-1.pyで使った厳密解の値は、exact/outputs/ に置いてある
  - 厳密解の値は、exact/free_energy_finite を実行すれば計算できる

    usage: *exact/free_energy_finite L Tmin Tmax Tstep*
    例： L=64 T=1.0 ~ 2.0まで 0.1刻み

    exact/free_energy_finite 64 1.0 2.0 0.1

TRGの実習（２）：差分を用いた物理量の計算

- Make_EC.py：自由エネルギーの出力結果から差分近似でエネルギーと比熱を計算するスクリプト
- 実習（１）の出力ファイルは別のフォルダに移動
  - 例えば
    - mkdir tutorial1-1
    - mv *.dat tutorial1-1
- *L=2, T=1.5~3.0*の自由エネルギーをΔT=0.01の刻みで計算
  - *python TensorNetwork_TRG.py -n 1 -Tmin 1.5 -Tmax 3.0 -Tstep 0.01 --step*
    - ファイル：free_energy_L2_D4.dat ができる
- Make_EC.pyによるエネルギーと比熱の計算
  - *python Make_EC.py free_energy_L2_D4.dat*
  - energy_from_free_energy_L2_D4.dat, specific_heat_from_free_energy_L2_D4.datができる
- 結果のプロット
  - *python Plot_TRG_2-1.py*

TRGの実習（２）：つづき

- スクリプトによる複数サイズの計算
  - L=2,4,8,16,32の計算
  - *python TRG_tutorial2-2.py*
- 結果のPlot（エネルギー、比熱の温度依存性のグラフ）
  - *python Plot_TRG_2-2.py*
  - スクリプト内でMake_ECの関数Calculate_EC(T,f)を呼んでエネルギーと比熱を計算しているため、ファイルの書き出しはない

TRGの実習（３）：モンテカルロ法との比較

- TRGで計算したエネルギー・比熱をモンテカルロ法の結果と比較する
  - *python Plot_TRG_3-1.py*
  - エネルギー、比熱の温度グラフが表示される
  - （注）ディレクトリ構造を変えた場合には、プロットファイルを修正する
  - モンテカルロ法の計算結果を消した場合には再計算が必要
    - *cd ../Tutorial_MC/simplemc*
    - *parameters2xml parm9a*
    - *simplemc parm9a.in.xml*

基本課題３：

- モンテカルロ法とTRGで計算したエネルギー、比熱を比較する
  - モンテカルロ法は（SWEEP数が十分に大きければ）、統計誤差の範囲内で厳密である一方、TRG法は特異値分解を用いた低ランク近似と、エネルギー・比熱の差分近似からくる二つのバイアス（真の値からのズレ）が存在する。この点を念頭において、

    1. プロットしたグラフを見て、モンテカルロ法とTRGの結果を比較し、それぞれの手法の精度、それらのサイズ・温度依存性などについて議論せよ
    2. モンテカルロ法、TRGの双方について、計算精度を上げる（モンテカルロ法であれば、統計誤差を小さくする、TRGであれば、真の値からのズレを小さくする）パラメタの変更例、計算の変更例を提案せよ。また、それらのパラメタで実際に計算を行った場合の計算時間の増減について議論せよ（単に、減る・増えるだけではなく、およそ何倍になる等、少し定量的な議論を期待します）。

- Tips
  - もし厳密なエネルギー・比熱の値を精度よく知りたければ、exact/free_energy_finite を、“細かい温度刻み”で実行してファイルに書き出し、

    （例） *exact/free_energy_finite 64 1.0 2.0 0.001 > free_energy_exact_L64.dat*

    差分近似でエネルギー・比熱を計算すれば得られる。

    python Make_EC.py free_energy_exact_L64.dat

発展課題２：

- 大きなシステムサイズの計算をする
  - モンテカルロ法とTRGでは、他のパラメタを固定してシステムサイズLを変えた際の計算時間のL依存性が異なる。

    1. L=48, 64, …とサイズを増やした計算を実際に行って、計算時間のサイズ依存性を調べよ
    2. 誤差を一定にするように、サイズ以外のパラメタも合わせて変えた場合、計算時間がどうなるか調べる or 考察（推測）せよ
       （モンテカルロ法では例えば“比熱の誤差”が同程度に、TRGでは、“自由エネルギーの厳密解からのズレ”が同程度になるように、サイズに合わせて、パラメタ（SWEEP数、Dなど）を変える）
    3. 以上の結果を踏まえて、大きいシステムサイズの計算ではモンテカルロ法とTRGのどちらを使う方が良いか議論せよ。

レポート課題の説明

- レポート内容
  - 今日のの実習で提示した基本課題１、２、３と発展課題１、２をレポートにまとめて提出してください。
    - 基本課題１、２、３には全て「取り組む」こと
      （課題の結果を出せなかった場合でも、一言だけでも「〇〇が難しくてできなかった」など書いていれば「取り組んだ」とみなします。）
      - 発展課題はoptionalです。やらなくても良いですが、もしやっていれば、加点します。
- レポートの提出方法
  - ITC-LMSからレポートを提出してください
  - ITC-LMSが使えない場合、メールでの提出も受け付けます。
    t-okubo@phys.s.u-tokyo.ac.jp
- レポートの提出締め切り
  - 7/31

もっと詳しく学びたい人へ

- ALPS wiki
  - http://alps.comp-phys.org/mediawiki/index.php
  - 日本語対応。チュートリアルもあります。
- ALPS解説記事
  - 「“実験技術”としての量子多体系シミュレーションソフトウェアALPS」藤堂眞治, 日本物理学会誌, **70**, 275 (2015年4月).
- モンテカルロ法
  - “A Guide to Monte Carlo Simulations in Statistical Physics” D. P Landau and K. Binder, 4th edittion, Cambridge University Press (2015)
- テンソルネットワーク法（テンソルネットワーク繰り込み群）原論文
  - “Tensor Renormalization Group Approach to Two-Dimensional Classical Lattice Models”, M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007)
- テンソルネットワーク法解説記事
  - 「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, 日本物理学会誌2017年10月号
    （別刷りが少しありますので欲しい方は連絡ください）

MateriApps ― 物質科学シミュレーションのポータルサイト

＊MateriApps のハンズオン資料から借用

・公開ソフトウェア(アプリケーション)を核としたコミュニティー形成をめざして

2013年5月公開

- 155の物質科学アプリケーションやツールを紹介(2015年9月現在)
- 「やりたいこと」からアプリケーションを検索
  - 検索タグ：「特徴」「対象」「手法・アルゴリズム」
- 開発者の声を利用者に届ける
  - アプリ紹介、開発者ページ、アプリの魅力・将来性・応用性
- フォーラム(掲示板)を利用した意見交換
- 講習会情報・web講習会・更新情報
- 月間 8000 ページビューにまで成長

MateriApps 掲載アプリケーション

＊MateriApps のハンズオン資料から借用

- 155の物質科学アプリケーションやツールを紹介 (2015年9月現在)

密度汎関数法

AkaiKKR☆

OpenMX☆

xTAPP☆

ABINIT☆

... (37)

量子化学

FMO☆

SMASH☆

GAMESS☆

DC☆

... (19)

分子動力学

MODYLAS☆

Gromacs☆

ERmod☆

MDACP

... (19)

格子模型

ALPS☆

DSQSS

BLOCK

DMRG++

... (22)

連続体シミュレーション

ANSYS Multiphysics

Octa ... (8)

データ解析

CLUPAN☆

phonopy☆ (26)

可視化

fu☆

TAPIOCA☆ (28)

☆ MateriApps LIVE! 収録 (一部予定) アプリ

MateriApps 活動の目的

＊MateriApps のハンズオン資料から借用

- 開発者側の問題点
  - 有益なプログラムはもっと使われるべきだが、多くのソフトは研究室内にとどまって終わる
  - 公開・情報発信には手間がかかる
  - アプリ開発を成果として主張しにくい(指標がない)
- 利用者側の問題点
  - どんなプログラムがあるのかよくわからない
  - インストール・使い方について知りたい
  - 開発者の活動(特に講習会情報)をもっと知りたい

- 両者をつなぐ役割を果たしたい

MateriApps 企画・制作

＊MateriApps のハンズオン資料から借用

- 運営:
  - 計算物質科学イニシアティブ (CMSI)、東京大学物性研究所 (ISSP)、自然科学研究機構 分子科学研究所 (IMS)、東北大学金属材料研究所 (IMR)
- 計算物質科学イニシアティブ(CMSI) 広報小委員会
- MateriApps 開発チーム
  - 藤堂眞治 (東大理/ISSP)、加藤岳生 (ISSP)、五十嵐亮 (CMSI-ISSP)、笠松秀輔 (ISSP)、川島直輝 (ISSP)、小西優祐 (CMSI-ISSP)、寺田弥生 (CMSI-IMR)、野田真史 (CMSI-IMS)、松尾春彦 (RIST)、吉澤香奈子 (RIST)、吉見一慶 (ISSP)
  - (委託) 佐々木翔一、土田成宏
- 協力:
  - CMSI元素戦略拠点、東京大学物性研究所 計算物質科学研究センター、自然科学研究機構 分子科学研究所 計算分子科学研究拠点、東北大学金属材料研究所 計算材料科学研究拠点、高度情報科学技術研究機構