アニーリングマシンにおける実数表現の違いがもたらす顕著な計算性能の変化とその考察について

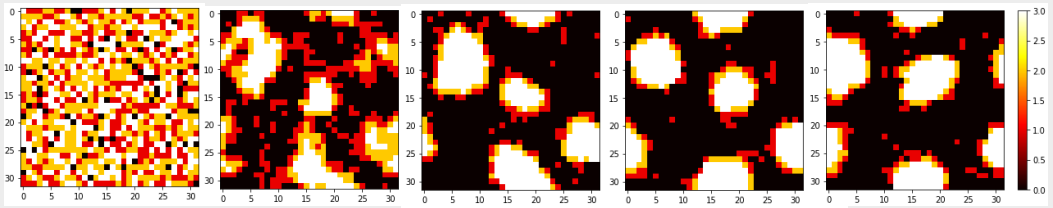
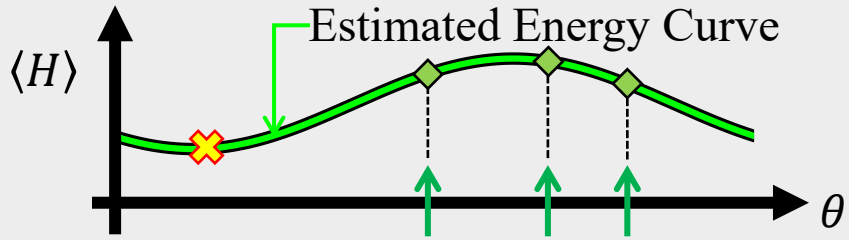
2024/05/10

Katsuhiro Endo (AIST)

量子コンピュータの分類

- 古典コンピュータ ➡古典ビットを利用して計算
- 量子コンピュータ ➡量子ビットを利用して計算

- 量子コンピュータは，アニーリング方式とゲート方式に大別される．

違い ＼ 方式	量子アニーラ（アニーリング方式）	ゲート式量子コンピュータ（ゲート方式）
計算対象	組み合わせ最適化計算に特化	任意の量子計算が可能（汎用）
ノイズの影響	ノイズに強い	ノイズに弱い（計算結果が壊れる）
計算規模（現在）	数千～数万 量子ビット	数百 量子ビット（ノイズ有）
マシン例	D-wave advantage など	IBM Quantum System One など
遠藤の過去の研究（例）	 ミクロ相分離シミュレーション [Endo et al. 2022]	 変分量子回路の最適測定法 [Endo et al. 2023]

アニーリングマシンとQUBO (2次制約なし二値最適化問題)

●アニーリングマシンは，QUBOハミルトニアンが最小となる解を探索できる．

（二値変数の組）

●ハミルトニアンの基底状態を求めることによる方法

解きたい問題

↓ ハミルトニアンの基底状態が解くべき問題の解となるように設計．

ハミルトニアン

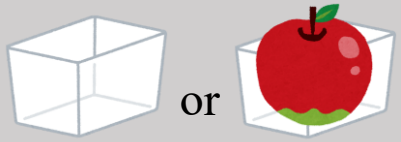
量子コンピュータ

基底状態（解）

●QUBOハミルトニアン

$$H = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} q_i q_j$$

q_i ： 2値変数 0 or 1



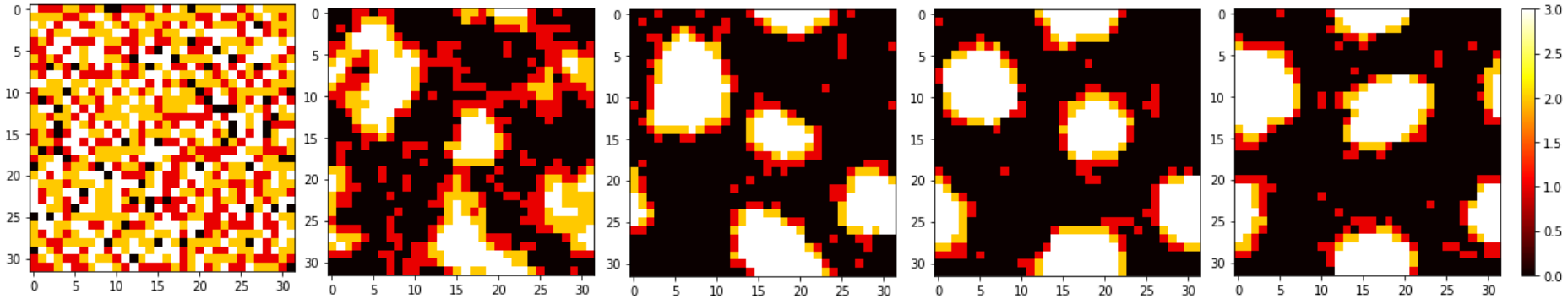
C : QUBOマトリクス (NxN実数係数行列)
ハミルトニアンは2値変数の二次形式．
係数を解きたい問題に合わせて設計

（補足）
- 量子アニーラは，このQUBOハミルトニアンの基底状態（エネルギー最小解）を求めるのに特化．
- ゲート式量子コンピュータでは，QAOAなど他の量子アルゴリズムによって実現可能．

実数を含んだ最適化問題

工学的問題を解決するCAEでは，最適化変数に実数を含むことが多い

- フェーズフィールドシミュレーション （フェーズフィールド変数が実数）



アニーリングマシンによるミクロ相分離フェーズフィールドシミュレーション [Endo et al. 2022]

- トポロジー最適化（材料分布が実数）

（都合により削除）

アニーリングマシンによるトポロジー最適化 [Ye et al. 2023]

2値変数と実数など混合最適化問題もある．

QUBOと実数表現

- 解くべき問題をQUBOに変換する必要がある.
- QUBOは二値変数（0/1）を変数として扱う
- 実数値を決定変数として直接扱うことはできない.

➡ 実数値を二値変数で間接的に表現する必要がある.

$$H = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} q_i q_j$$

QUBOハミルトニアン（再掲）

q_i ： 2値変数 0 or 1

● 実数表現：複数の二値変数で実数値を表現

$$x = t + s\sum_{r=1}^{R} F_r q_r$$

s, t : スケール・シフト
(表現する値の
範囲を定める)

F_r ： 重み係数

例）二進数表現 ：2進数の各桁を表す

$$x = \sum_{k=1}^{R} 2^{-r} q_r$$

$$x = 0.1011_{(2)} = \frac{11}{16}$$

$$F_r = \frac{1}{2}\ \frac{1}{4}\ \frac{1}{8}\ \frac{1}{16}$$

本研究の目的

- 実数を決定変数とする最適化問題を，異なる実数表現でQUBOに変換した場合，違いが表れるか調べる．　（異なる実数表現：異なる方法で生成したF_k）

●今回比較するために提案した実数表現

1. 一様乱数和表現

$$F_r \sim \mathcal{U}(0,1)$$

2. 正規乱数和表現

$$F_r \sim \mathcal{N}^+(0,1^2)$$

3. 定数和表現

$$F_r = 1$$

4. 二進数表現

$$F_r = 2^{-r}$$

(補足)
・乱数は各実数ごとに独立に生成する
・各実数が0-1の範囲を表すようにt,sを決定する．
・$\mathcal{N}^+$は正規分布の正の部分

$$x = t + s\sum_{r=1}^{R} F_r q_r$$

実数表現

実験設定　(線形方程式を解く)

Problem to solve

$$Ax = b$$

A, b : coefficients
x : variables (real number)

Unconstrained problem

$$\underset{x}{\operatorname{argmin}} \ \|Ax - b\|^2$$

QUBO

$$\underset{\{q_i\}}{\operatorname{argmin}} \ \Sigma_{i,j} \ C_{i,j} q_i q_j$$

Ising Machine

Result (binary variables)
$\{q_i\}$

Decision variables (binary variables) $\{q_i\}$

q_i ： 2値変数 0 or 1

Real number representation using binary variables

$$x_i = t_i + s_i \ \Sigma_i \ F_{i,r} q_{Ri+r}$$

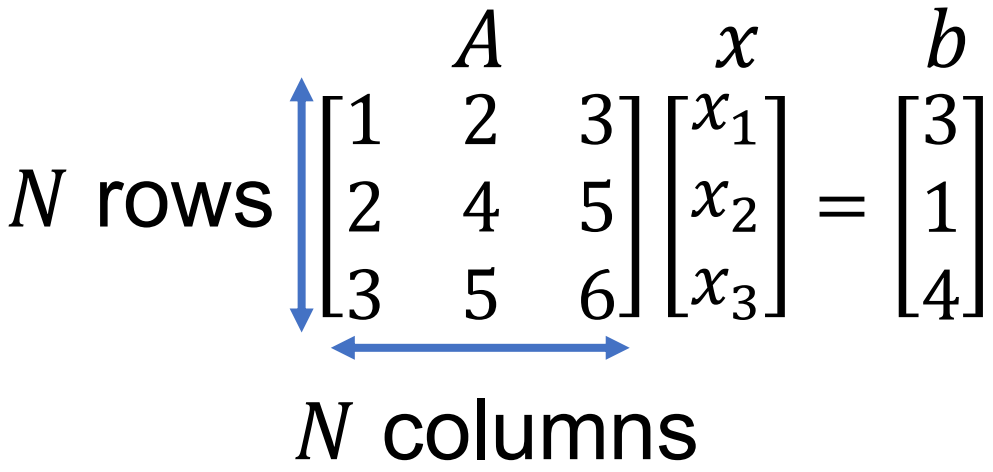
$t_i, s_i, F_{i,r}$ depends on the type of Real number representation

実数表現を変えると解の精度は変わる？

Solution
x

実験条件：線形方程式の生成

- $Ax = b$ を解く



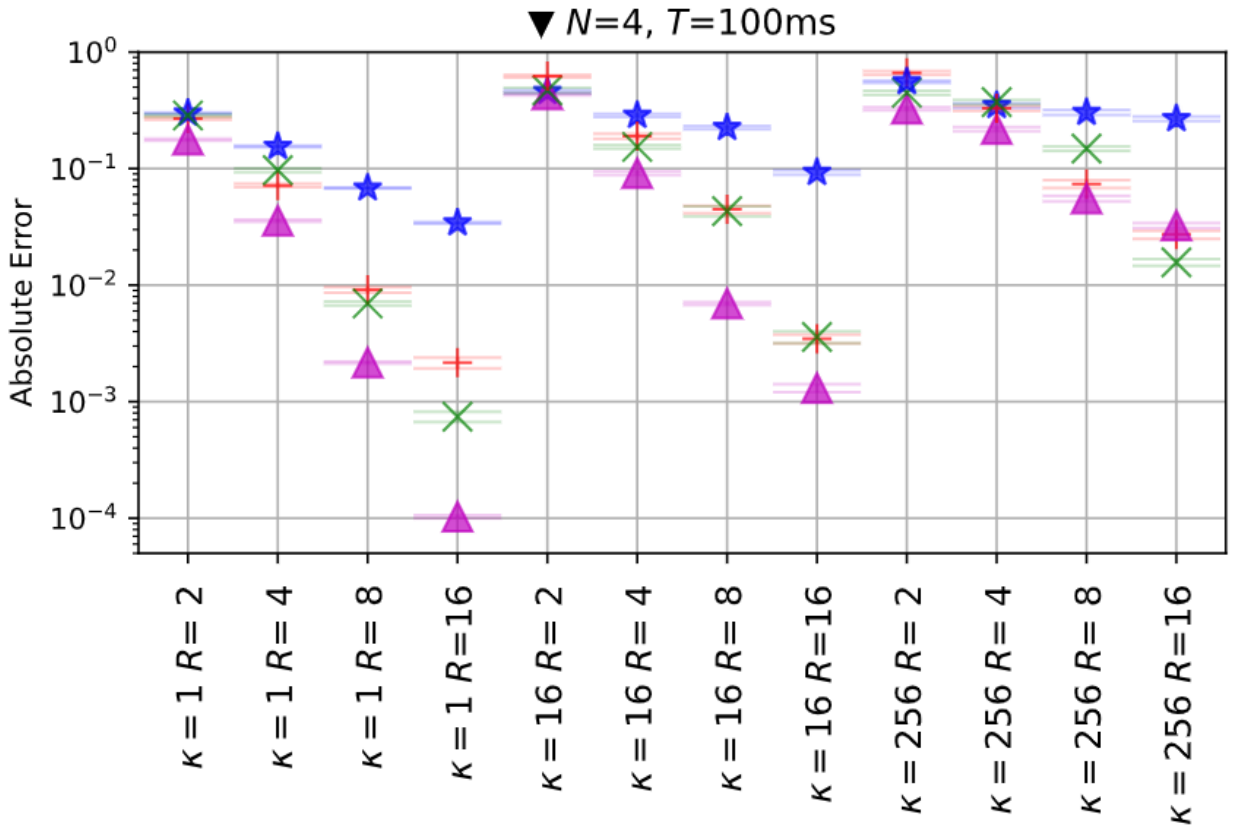
$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix}$$

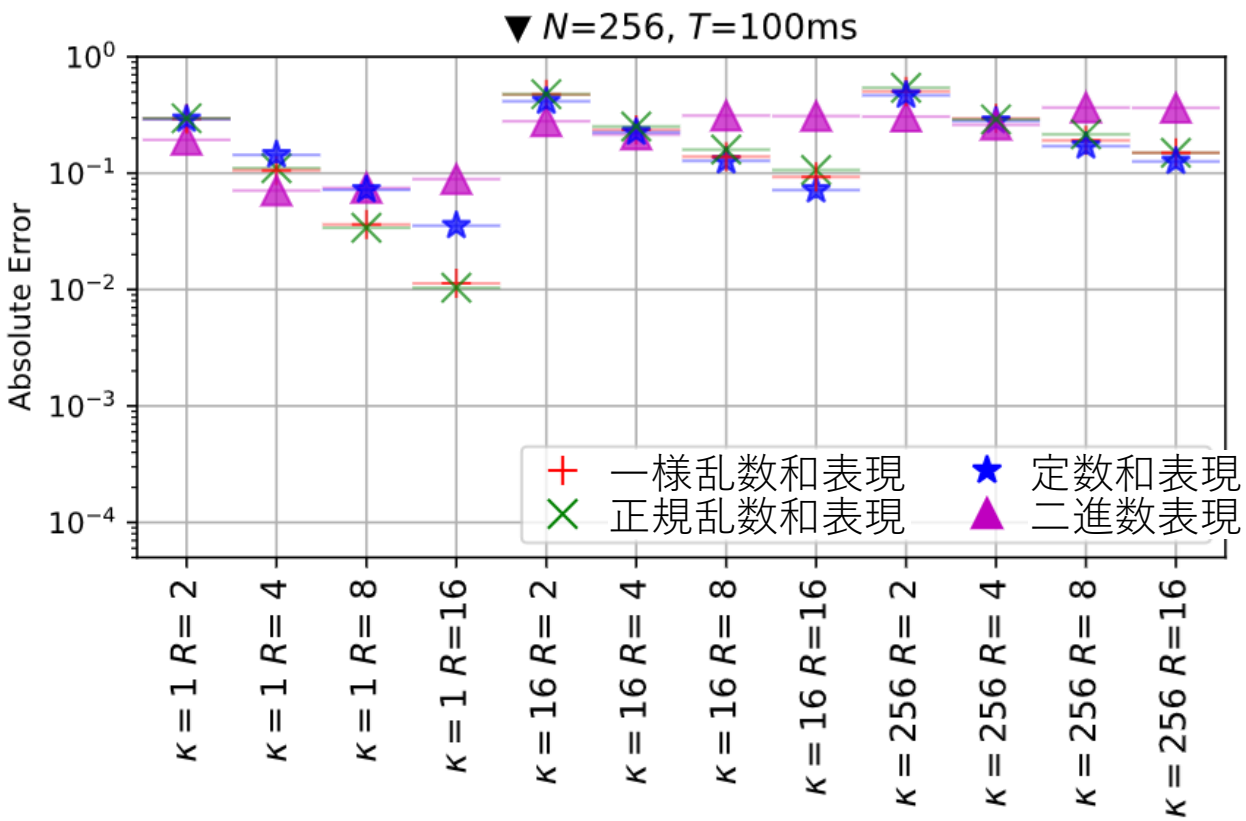
・線形方程式の難易度による比較を検証するため，以下の難易度指標を導入

1. 実数変数の数 (=行列の行/列の数) : N.
2. 行列Aの条件数 : κ, Aの最大固有値と最小固有値の比 ($\kappa = \lambda_{MAX}/\lambda_{MIN}$)

➡ どちらもおおきくなるほど難しくなるといえる．

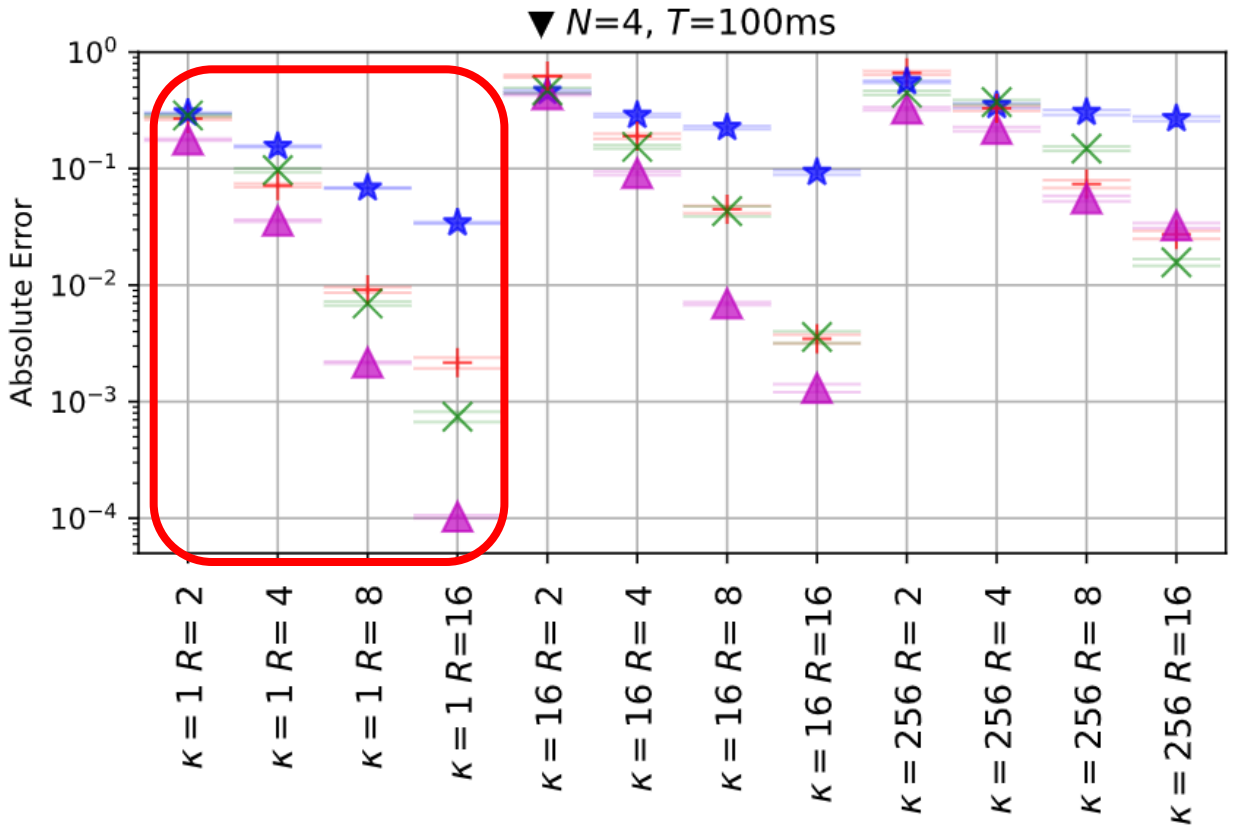
実験結果（Fixstars Amplify Annealing Engineを使用）

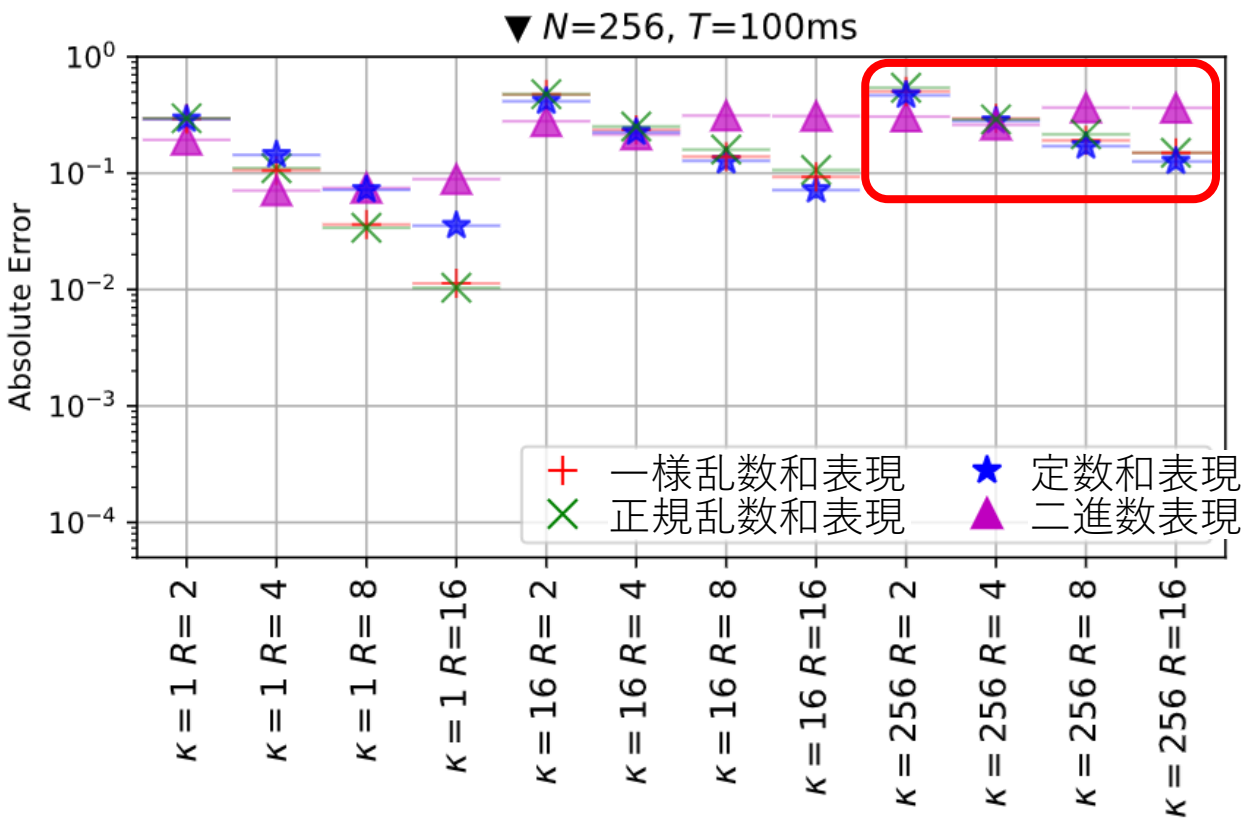




- 左はN=4の場合（各列に異なる条件数κと実数あたりのビット数Rの結果）
- 右はN=256の場合．　各条件は16回試行の平均誤差を表示．
- 実験としてはN=4,8,16,32,64,128,256, κ=1,16,256, R=2,4,8,16のすべての組み合わせで実行した（図は一部を表示している．）

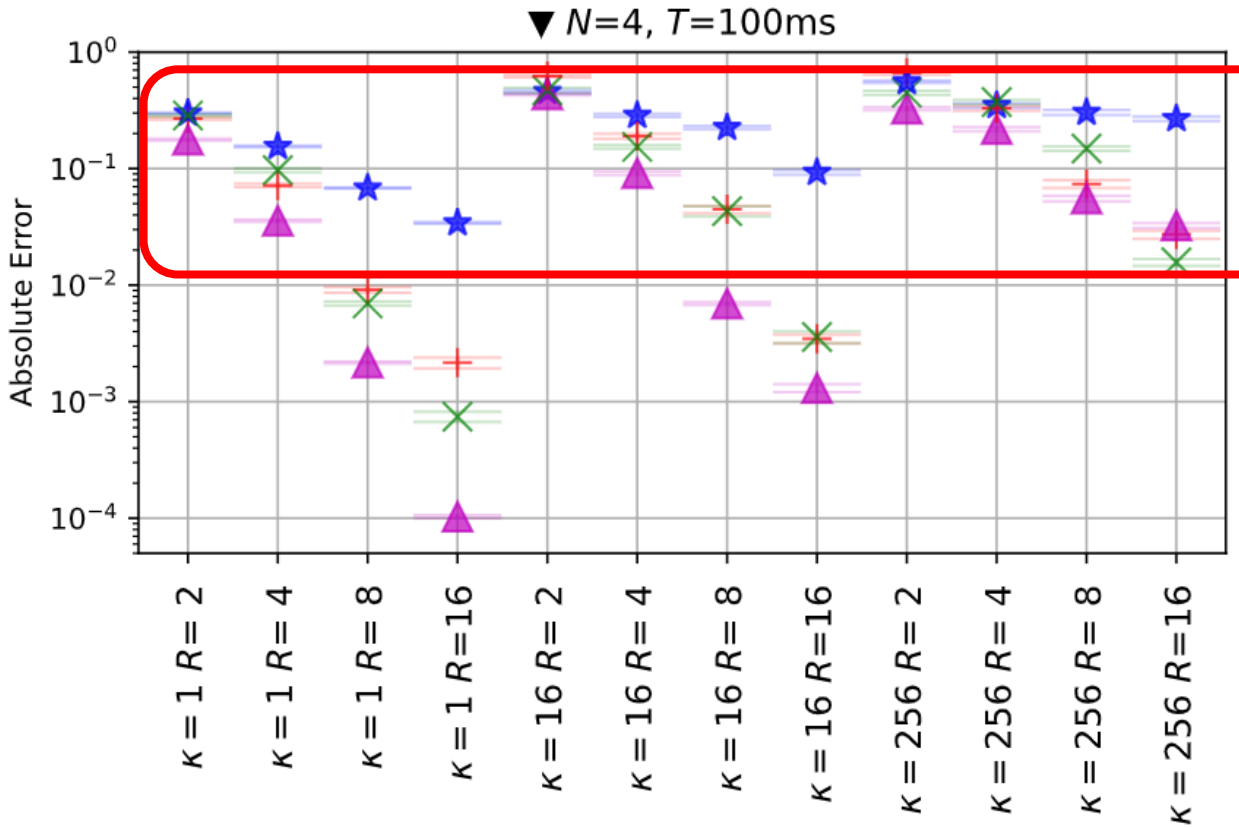
実験結果（Fixstars Amplify Annealing Engineを使用）

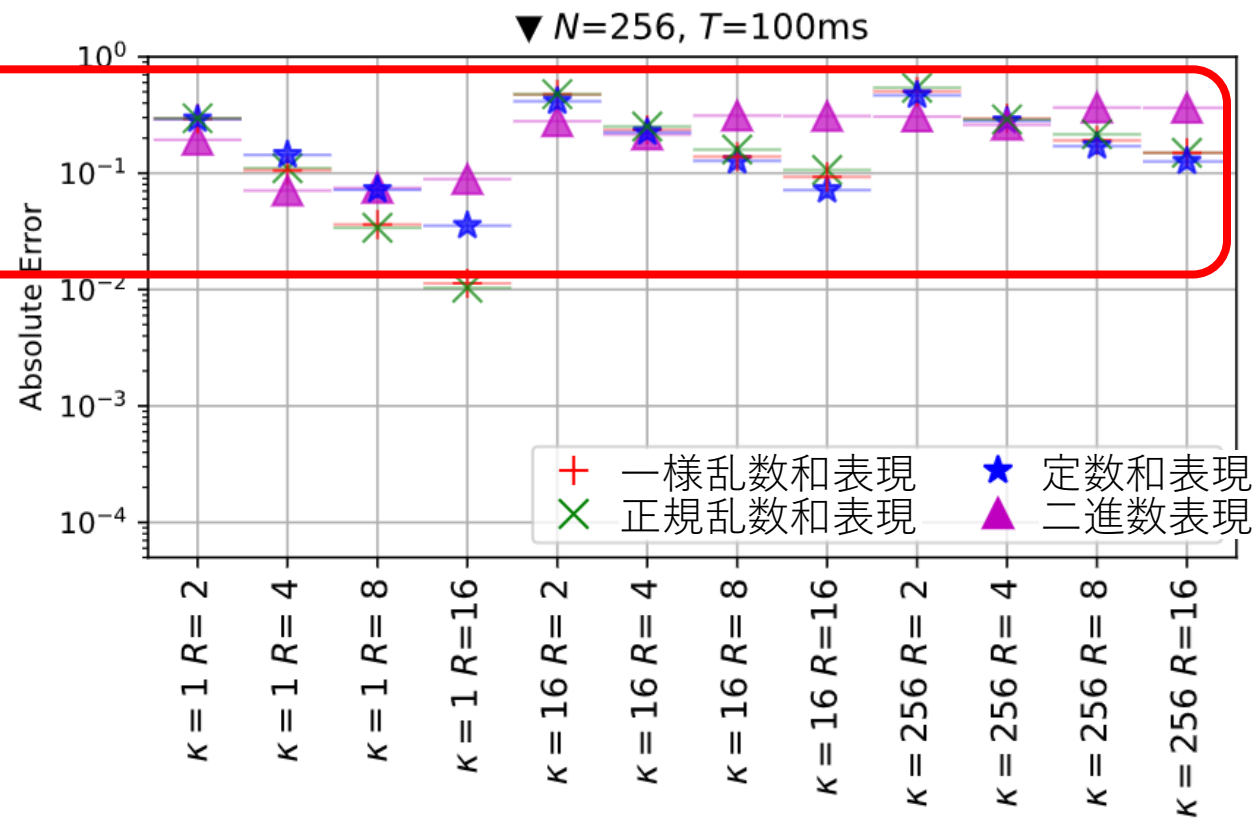




- Nとκが小さいとき，二進数表現 はビット数(R)を多くするほど精度が改善．他の表現と比べても最良
- 一方そうでないとき，二進数表現 は他の手法に負ける．
  それどころか，ビット数(R)を多くすると逆に性能が悪化するケースもみられる．

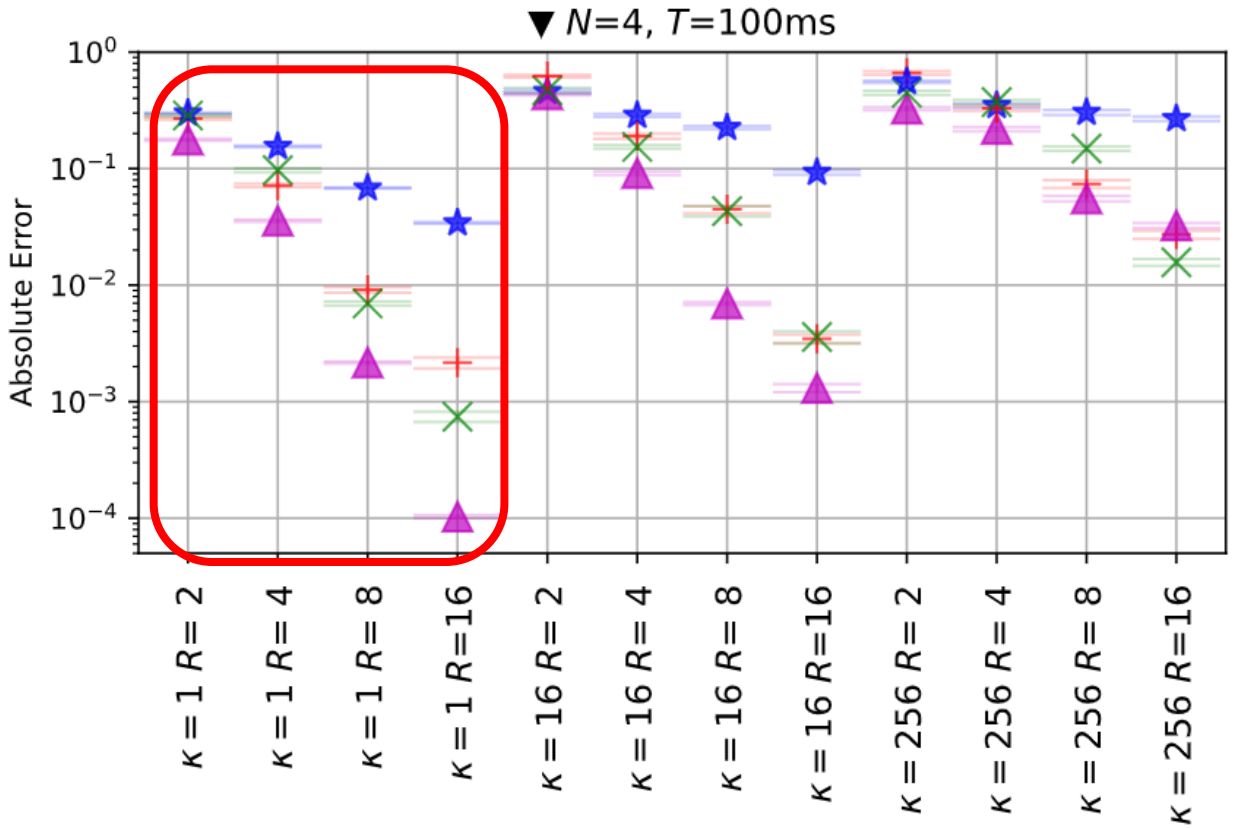
実験結果（Fixstars Amplify Annealing Engineを使用）

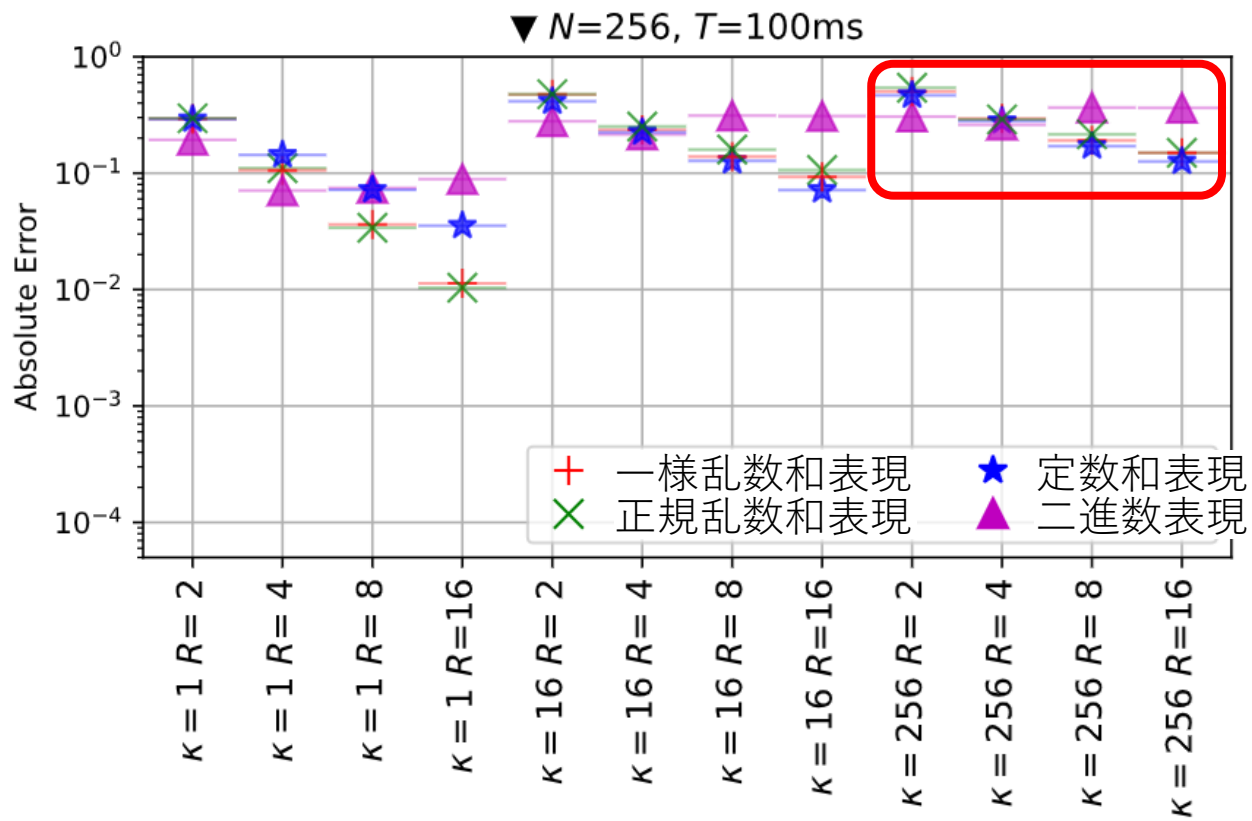




- 定数和表現 の場合，どのような条件においても，ビット数(R)を増やすと精度が改善する．ただし，平均的な性能は悪い．
（これは表現できる数値の解像度が低いため）

実験結果（Fixstars Amplify Annealing Engineを使用）



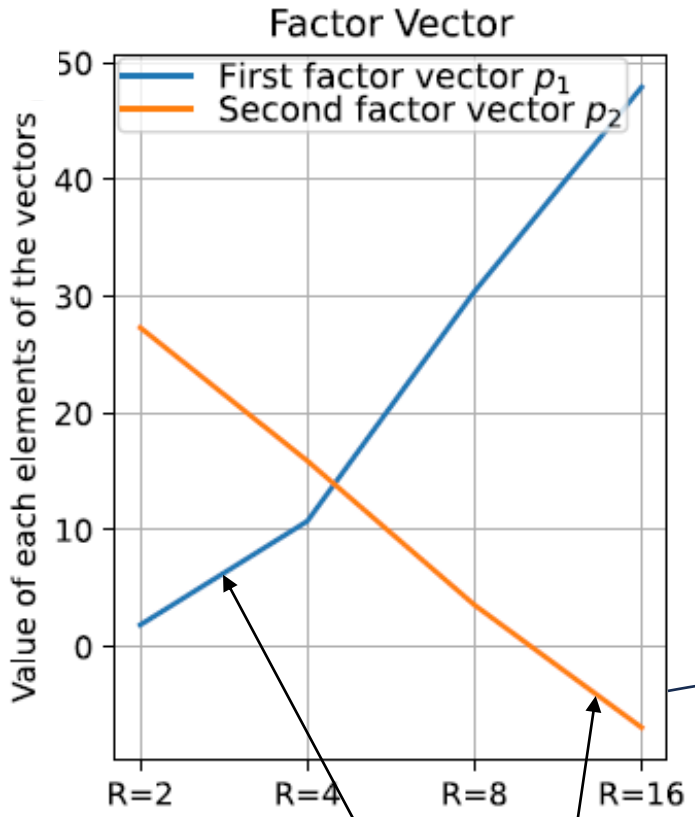


- 2つの乱数和表現は，定数和表現 と 二進数表現の良いとこどりにである．
  Nとκが小さいときは二進数表現に近く，大きいときは定数和表現に近い．

➡（少なくともこの4つの比較では）乱数和表現が優れた性能を持っている．

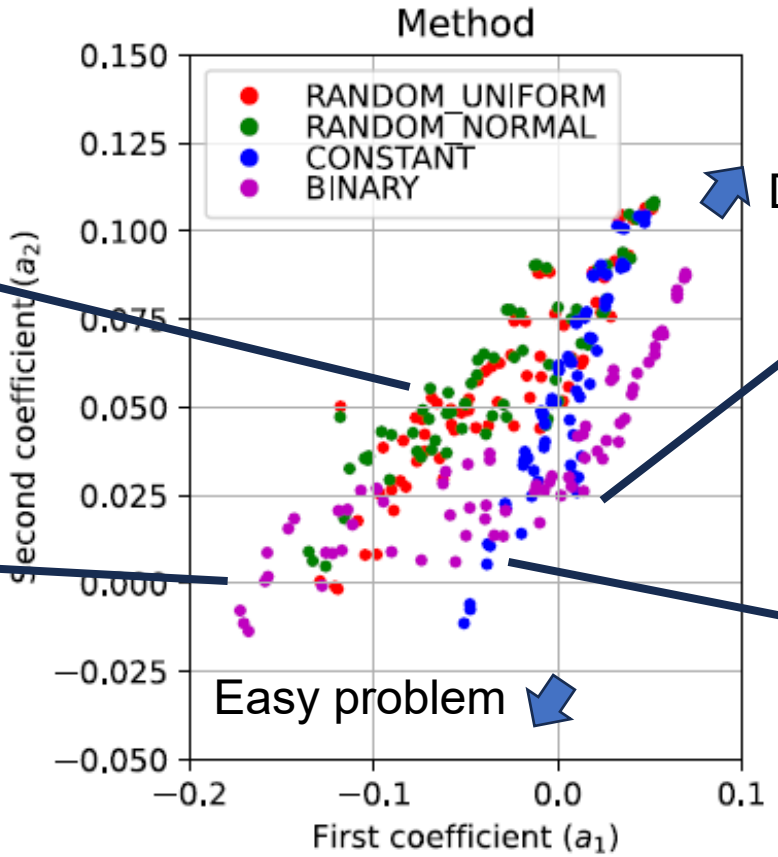
傾向の統計分析　（誤差の主成分分析）

- 同条件でR=2,4,8,16の誤差を1つのベクトルとしてまとめ，主成分分析．



$$\log E^i = \bar{y}1 + a_1^i \boldsymbol{p}_1 + a_2^i \boldsymbol{p}_2 + \boldsymbol{\epsilon}^i$$

98.6% contribution



乱数和表現
いいとこどり

二進数表現
易しいときに
R増⇒精度高

二進数表現
難しいとき
急激に悪化

定数和表現
線形に精度悪

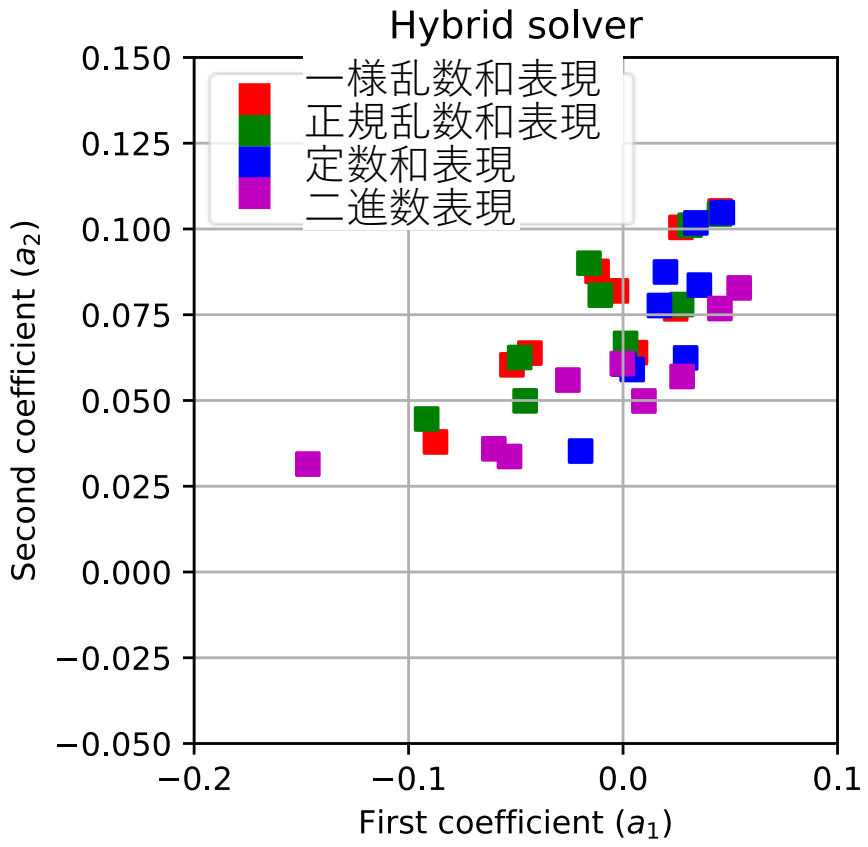
$\boldsymbol{p}_1$：誤差がビット数に伴い増加する成分(悪い状況)．

$\boldsymbol{p}_2$：誤差がビット数に伴い減少する成分(良い状況)．

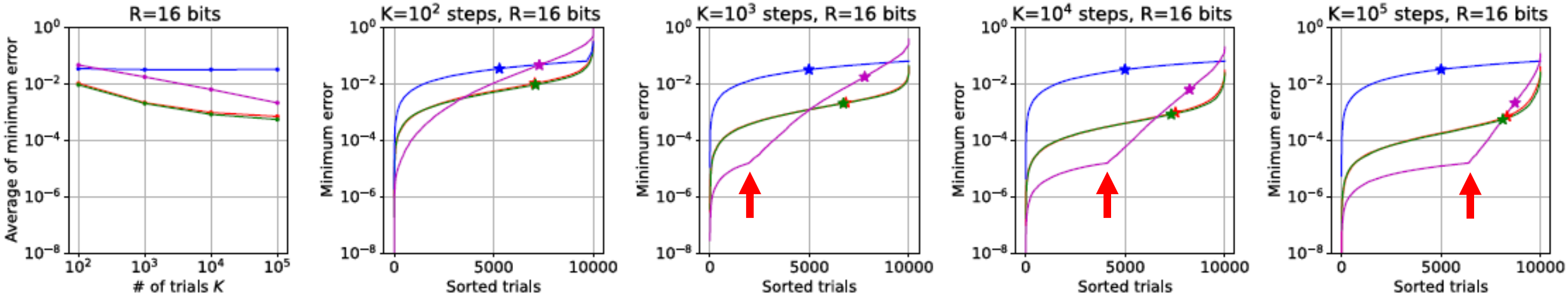
D-wave Hybrid Solver

- D-wave Hybrid Solverでも同様の傾向

➡ GPU Annealer 特有の問題ではない



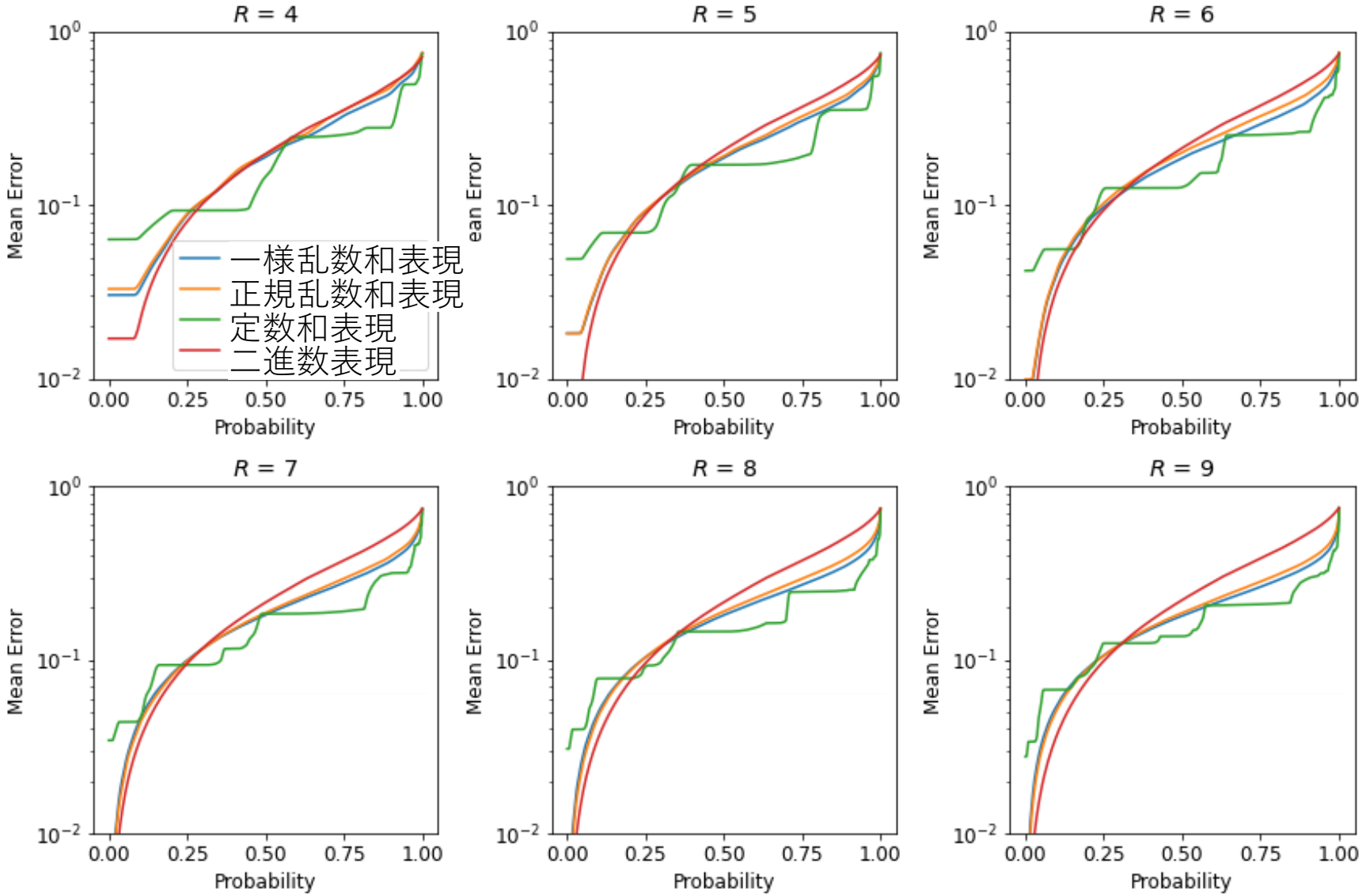
なぜ二進数表現は悪いのか？トイモデルによる解析.



- トイモデルとして，1つの実数値最適化モデルを考えた．
- トイモデル：各ビットは一定の確率で独立に反転を試み、最適値に近づけばそれを受け入れる。
- この図は、このような動きを指定回数K回繰り返したときの誤差を示している。
（図：各条件下で独立に10000回試行し，その結果を精度順に並べたもの）
- 図より：二進数表現は高い精度となる場合と低い精度となる場合の明確な違いが見受けられる（赤矢印）．この境目では，同時にビットを反転させないと誤差が小さくならないような状況が発生していることがわかった．

本物の量子アニーリングマシンでも同じことが起きる

- 1つの実数値最適化モデルを，理想的な量子アニーリングマシンのシミュレーションにより実行した（量子断熱過程）．
- 図より，本物の量子コンピュータでも，実数表現手法の違いにより精度の違いが出ることがわかった．



結論

- 実数を決定変数として持つ線形方程式を，異なる4つの実数表現でQUBOに変換した場合，違いが表れるか調べた．
- その結果，次のことがわかった．
- 二進数表現は問題の難易度が高いときにはビット数を増やしても精度が改善しないため，使用すべきでない．
- 定数和表現はビット数を増やすと順調に誤差が改善する．
  しかしながら平均的な精度は悪い．
- 2つの乱数和表現は，問題が易しいときには二進数表現と同等の性能を，難しいときには定数和表現に近い性能を持つ．
  このため（本実験の範囲では）乱数和表現の使用が推奨される．
- これらの結果の原因を解析し，二進数表現においては同時ビット反転の必要性が精度を悪化させていると考えられる．