スクリプト言語 Xcrypt による 格子 QCD シミュレーションの自動化

斎藤 華[†1,†2] 朴 泰祐[†1] 金谷 和至[†2]
塙 敏博[†1] 佐藤 三久[†1]

本研究では，小規模問題に対する格子 QCD シミュレーションのパラメータサーチを自動化する．複数ジョブによりパラメータ空間の探索を行う場合，ジョブ間に依存関係があるため，より多くの人的コストがかかってしまう．ここではスクリプト言語 Xcrypt によって，依存するジョブの実行をコントロールしジョブ実行を自動化することで，バッチスケジュールや複数ジョブの管理による煩雑さを回避し，人的コストの削減を目指す．従来は，次に探索すべきパラメータを前のシミュレーション結果から人手により求めてきたが，本研究では結果データのまとめと推測を自動化し，実験の遂行を効率化する．また，自動化による問題点を確認し，物理シミュレーションが正しく行われるための条件を明確化する．作成した Xcrypt プログラムをスーパーコンピュータ T2K-Tsukuba 上で動作させた．これまで人手によってヒューリスティックに行ってきた小規模 QCD 問題のパラメータサーチ処置を効率的に自動化することができ，物理的に意味のある結果が得られ，同システムの有効性が確認された．

Automatic Parameter Search on Lattice QCD Simulation with Script Language Xcrypt

Hana Saito,[†1,†2] Boku Taisuke,[†1] Kazuyuki Kanaya,[†2]
Toshihiro Hanawa[†1] and Mitsuhisa Sato[†1]

We optimize the parameter search in a sequence of small size simulation jobs in lattice QCD using the script language Xcrypt. Thanks to the job control functions of Xcrypt, we avoid complications in the batch scheduling and arrangement of multiple dependent jobs. To accelerate the parameter search problem on small size QCD simulation, it is required to process the job execution result automatically and dispatch multiple jobs according to their dependency. We implemented such a system on T2K-Tsukuba supercomputer. As a result, the simulation workflow is automated and the correctness of the physics result is proved with fully automated and effective parameter search.

1. はじめに

物質の最小単位は素粒子であり，クォークはその素粒子の一つである．約１兆度以上の超高温におけるクォーク物質の性質を解明することは，宇宙の初期進化を解明する上で重要である．グルーオンによって媒介されるクォーク間の力は QCD(量子色力学) で記述されるが，相互作用が極めて強いために，時空を格子で離散化した格子 QCD に基づく数値シミュレーションが，クォーク物質の性質を研究する唯一の系統的方法になっている．最高速のスーパーコンピュータを使った大型計算が進められており，並列化やアルゴリズム開発による計算の高速化・最適化も盛んに研究されている[1)]．

本研究では，大規模計算ではなく小規模な問題の繰り返しに注目する．ここでは，格子 QCD シミュレーションにおいて，パラメータを変えながら小規模ジョブを繰り返し実行することによってパラメータ空間を探索するタイプの問題に，スクリプト言語 Xcrypt によるジョブの制御を導入する．依存関係のあるジョブの管理には煩雑さが伴う．Xcrypt を用いて，それを回避する．格子 QCD シミュレーションにおけるパラメータサーチの一例全体を最適化することが最終目

†1 筑波大学システム情報工学研究科
Systems and Information Engineering, University of Tsukuba
†2 筑波大学数理物質科学研究科
Graduate school of Pure and Applied Sciences, University of Tsukuba

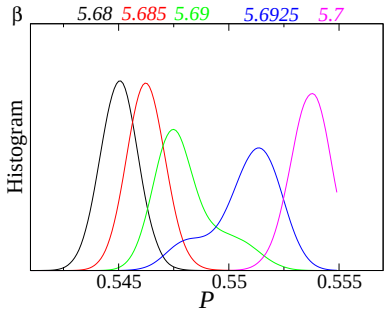

図 1　物理シミュレーションにおいて必要となるヒストグラム

標であり，本研究では 1 つのジョブ結果の処理から，そのジョブを継続して行うか別のパラメータの探索を行うかを決定すると共に，状況に応じて，同時実行可能な複数ジョブの同時投入を行うことにより，シミュレーション全体を自動化・最適化する．

2. 物理シミュレーションの全体像

2.1 有限温度格子 QCD シミュレーションとクォーク物質の相構造

クォーク物質の相転移の解明は重要な研究テーマの一つである[2]．相転移の性質は相転移点の極近傍でしか見られないため，その解明にはパラメータ空間の探索が必要となる[3]．具体的な操作を説明する前に，得たいデータの特徴について述べる．

相転移を調べるためのシミュレーションにおいて，必要となるデータは**図 1** のようなものである．これはプラケット P と呼ばれる物理量（エネルギー密度に相当）の確率分布関数である．一度のジョブでのパラメータ β（β は温度に対応）に対して確率分布関数のピークが一つ得られる．ピークの位置や幅は β によって変化する．ジョブを繰り返し，複数の確率分布関数を求める事で図 1 が得られる．一次相転移点の極近傍ではダブルピーク，それ以外の点ではシングルピークのグラフがそれぞれ得られるので，確率分布関数から 1 次相転移の情報を引き出す上で重要となるのは以下の二つの特徴である：

特徴 1　ダブルピークの兆候が見られる β があること（図 1 の緑や青の β）．

特徴 2　範囲内の全ての P に対してヒストグラムに 1 個以上の 0 でない値があること．より厳密には，1 次相転移点をカバーする広い P の範囲で十分な統計精度があること．

2.2 シミュレーション全体の流れ

次に相転移点を探すシミュレーションの自動化の具

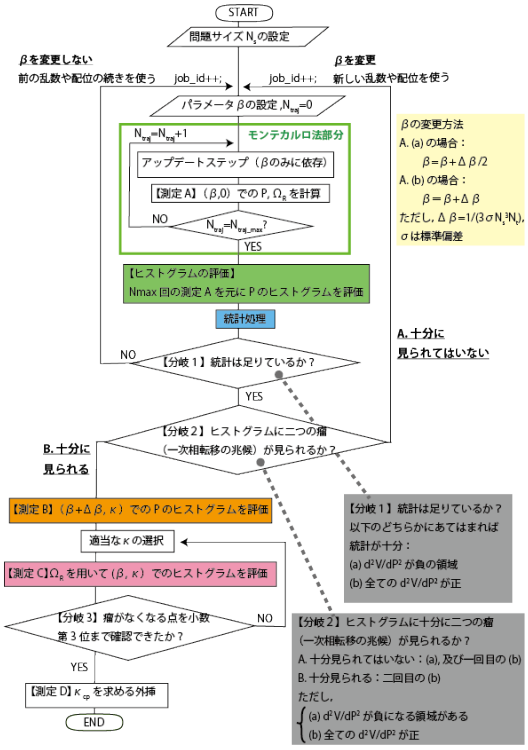

図 2　フローチャート

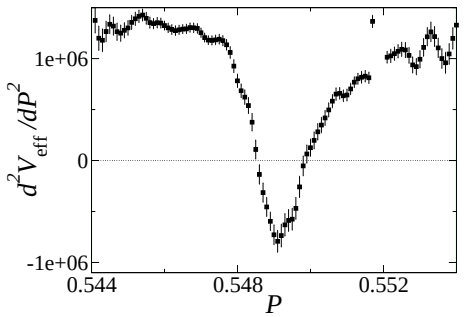

図 3　条件分岐 2 で用いられるデータ

体的な手順について説明する．本シミュレーションの具体的な目的は，予想されている範囲の中で，一次相転移の特徴を求めてパラメータ空間を探索することである．ピークの位置や幅の情報は大まかには予想できるが，正確には予想できない．つまり，上記の特徴を捉えるためにはヒューリスティックな処理を行い，トライ＆エラーを繰り返してパラメータ空間を探索しなければならない．これを実際に行うためには**図 2** のようなフローチャートに従ってシミュレーションを行う必要がある．このシミュレーションは計算の中心であるモンテカルロ法部分（図図 2 において緑の四角で囲

まれた部分）と，統計量のチェックと β のチェックを行いヒストグラムの作成をガイドするガイダンス部分（図図 2 におけるモンテカルロ法以外の部分）に区別される．モンテカルロ法は一般に，乱数を使って期待値を評価するために用いられる．ここでは特に，期待値計算に用いる確率分布を抽出している．また，ガイダンス部分における条件分岐のうちの分岐 1，2 では，各場合に対してパラメータが変更されてモンテカルロ法が繰り返されるような構造となっている．

ガイダンス部分について詳しく説明する．条件分岐 1 は統計量が不足しているため同じ β を用いてシミュレーションを継続して行うか，又はその β に関する統計量が十分に得られたため，異なる値の β を用いて始めからやり直すかを表している．これは統計精度によって決められるもので，研究の進捗に合わせて変更する可能性がある．また，条件分岐 2 は一次相転移点が見つけられたかどうかを判定しており，**図 3** のようなデータを用いて判定する．図 3 はヒストグラムの指数関数を用いて定義されたポテンシャルの曲率であり，相転移の次数はその曲率の符号から理解することができる．曲率の値が負になる領域があればヒストグラムの傾きが増加することを意味する．ヒストグラムの傾きが増加することはダブルピークの兆候があるということでもあるので，この特徴を用いれば，単純な条件で一次相転移点 の有無を判断できる．相転移点を十分に特定できた場合はジョブの実行を終了する．一方，特定できない場合には β の新しい値を設定して次のジョブを投入する．次のジョブで用いる β の値は，ピークの裾が重なるように選ぶ．ピークが正確なガウス分布になっている場合には，この条件を満たす β を正確に予想できる．しかし，相転移点の近傍ではピークはガウス分布にはならない（図 1 の緑や青の線）ので，条件を満たすような β を正確に選ぶことは難しくなる．ここでは，上記の方法で予想される β を基準に選び，それより狭い間隔としてその半分の値を選択することにする．

これらの条件分岐の判定は計算コストが最も多くかかるモンテカルロ法部分の結果に対して行われるが，その結果に対する処理であるためモンテカルロ法処理部分とは切り離して別のプログラムで処理可能である．またモンテカルロ法部分は格子 QCD シミュレーションに共通して用いられるものであり，アルゴリズムの改良や最適化が盛んに行われている．従って，モンテカルロ法処理を含む，図 2 のフローチャートに示す全ての処理を単一のプログラムとしてコーディングしてしまうことは，モンテカルロ法のアルゴリズムのみの改良に適さず，全対処理が複雑になり過ぎてしまうという結果を招き，これらを同一コードに書き合わせてしまうとコードが複雑になってしまう．コード生成や性能チューニングといった面での生産性を高めるために，モンテカルロ法部分とその他のガイダンス部分は別々に扱い，シミュレーション全体を制御するガイダンス部分をスクリプト言語で自動制御できるようにすることが望ましい．

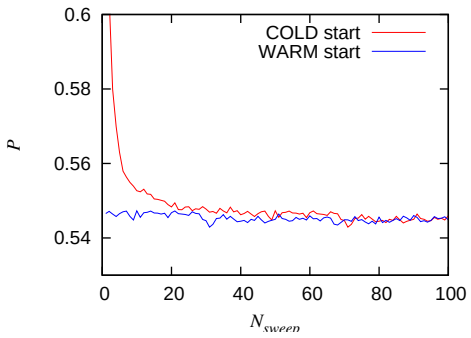

図 4 アップデータ過程の例

2.3 分散処理による性能向上

さらに，全処理の性能向上を考えると，ジョブの同時実行が重要となる．小規模 QCD の繰り返しは，1 つの β に対して複数の初期条件でシミュレーションを行い，その結果を統計的にマージして精度を高めることが可能である．ここで，依存関係のない独立したジョブを複数の計算機リソースを用いて同時に実行することができれば，当然その分パフォーマンスを向上させることができる．モンテカルロ法による数値積分ではこのようなジョブの実行が系統的に必要になる．これは，初期条件に依存しないデータだけを取り出すという操作が必要なためである．具体的な操作を以下で示す．

モンテカルロ法では**図 4** で示されているような配位を用いて積分値を求める．この図 4 では，二つの異なる初期条件 ('COLD start', 及び'WARM start') で始めたシミュレーション過程として, 各配位における観測量 P の値を示している．アップデートの回数を N_{sweep} と表すと，各 N_{sweep} での配位は一つ前の配位から作られるので．$N_{sweep} = 1$ の配位は $N_{sweep} = 0$ の配位の影響を大きく受け，$N_{sweep} = 2$ の配位は $N_{sweep} = 1$ の配位の影響を受けている．N_{sweep} が大きくなると，初期条件の影響が薄れ，ある値の回りを揺らぐ．また，初期条件によってその収束の様子は異なる．図 4 中では'WARM start' は収束性が良いように見えるが，一般にはそのような初期条件が見つかるとは限らない．

これらの平均値が積分値になるので，図 4 からわかるように，アップデートの回数の値が小さい区間だけを選び，期待値を計算すると，誤った答えを出してしまう可能性がある．そのため，異なる初期条件からシミュレーションを行う必要があり, 初期条件に依存している部分を除くという操作を行わなければならない．これらのシミュレーションでは各々のジョブの間には依存関係がなく, ジョブが独立なので同時に実行することができる．実シミュレーションに必要となるジョブの同時実行，さらにはそれらの結果を統合する操作までの自動化を行うことによって作業の効率化を測ることができると考えらえる．結果として，スーパーコンピュータのような多数の計算リソースを持つ環境において，パラメータサーチ問題を加速することが可能となり，処理の自動化と高速化が実現することができることになる．

分散処理することによる性能向上の程度は当然，分散処理できるジョブ数の，ジョブ総数に対する割合で決まる．本研究の場合には．ジョブの総数は以下のように見積もることができる．本シミュレーションでは β の値毎に順番にジョブを投入する．β の数はおおよそ 5 から 10 程度である．その各 β に対して初期条件の異なるシミュレーションを行うために，一つの β に対して二つのジョブを投入する．このようにして得られた結果を確認し，統計が不十分な場合にはジョブを追加で投入する．追加するジョブの数は 2 から 4 である．したがって，シミュレーション全体で投入するジョブの数は 20 から 60 程度となる．一方で，この一連のシミュレーションにおいて分散処理を取り入れられる箇所は初期条件の異なる二系列のジョブ投入部分であり，分散処理を取り入れる事で実行時間をおおよそ半分程度にすることができると期待される．

以上のシミュレーション全体は，１つの格子サイズ（3 次元空間＋ 1 次元時間の計 4 次元のサイズで決まる）について１つ行うことになる．また，本論文では直接触れないが，さらにシミュレーションそのものを特徴づける κ, μ というパラメータがあり，研究全体を見た場合はこのシミュレーションをさらに多数回実行する必要がある．従って，このようなシミュレーション全体のガイダンスン部分を作り，シミュレーションを自動化することは１回のモンテカルロ法処理のタイムステップ数の最適化や人手によるコストとエラーを大幅に減らし，研究全体の加速につながる．本論文では，一組の格子サイズ，κ，μ を対象としたシミュレーションを行うことを前提とする．

3. Xcrypt を用いた格子 QCD シミュレーションの実装

3.1 Xcrypt とその利点

図 2 のフローチャートでは次のシミュレーションパラメータの値を選ぶ時に前のシミュレーションパラメータでの解析結果を反映する必要がある．このトライ＆エラーとなる部分が全体の制御をより難しくしている．また，モンテカルロ法の計算プログラムは一般性があるので，これを独立したプログラムとし，入力パラメータを変えながら実行する形にしたい．一方で，このようなシミュレーションパターンは格子 QCD シミュレーションに限らず，科学技術計算では頻繁に見られるため，アプリケーション側のユーザが容易にプログラムでき，煩雑さを避けられるような言語が求められる．Xcrypt は，そのようなニーズに応えるために提案されている[4)–6)]．

Xcrypt はスーパーコンピュータ上の計算科学分野のシミュレーションを対象とし，同一のプログラムを異なるパラメータで大量実行することを自動化する目的で作られた．その特徴として，バッチ処理の煩雑さからの解放とディレクトリ操作やコーディングの容易さが挙げられる．現在，スーパーコンピュータのジョブ投入はバッチ処理によるものが主流となっている．シミュレーションを繰り返す場合，このバッチ処理を行うスクリプトが大量となり，その管理が煩雑となる．また，スクリプトに可搬性がないためにマシンによって書き換えなければならないという不便さもある．Xcrypt はバッチ処理に関するこれらの問題点を解消できるよう設計されている．また，ジョブを繰り返す場合には，ジョブを実行するディレクトリとデータを保存するディレクトリに注意しなければならない．例えば，ディレクトリを混同してインプットファイルを誤って読み込んでしまう可能性もある．Xcrypt の sandbox 機能を用いていることで，ディレクトリの操作が容易に行え，このようなトラブルを避ける事ができる．さらに，Xcrypt は perl をベースに作られている．このため，アプリケーション側のユーザでも容易にコーディングできるような設計となっている．

本研究における実装では，１つのジョブ結果に対する統計量の検証（十分かどうか）, その時の β に対するヒストグラム作成処理，それに基づく次の β の選定等，ある程度のプログラミングが必要でかつ条件を容易に変えられるようなガイダンス記述の枠組みが必要である．Xcrypt はこのような処理を容易に実現することが可能である．

表 1 Xcrypt が提供するコマンド

コマンド名	内容
xcrypt *myscript*.xcr	実行
xcryptdel *jobID1 jobID2 ...*	実行中のジョブを殺す（finished なジョブは無視）
xcryptdelall	
xcryptcancel *jobID1 jobID2 ...*	実行中のジョブを殺す （finished なジョブも aborted にする）
xcryptcancelall	
xcryptinvalidate *jobID1 jobID2 ...*	実行中のジョブを強制的に finished にする （二度と実行されない）
xcryptstat	実行中のジョブ一覧を表示
xcryptclean	実行中のジョブを全て殺し，履歴情報を忘れる

3.2 Xcrypt によるジョブコントロール

本節では Xcrypt の実行手順について説明する．科学技術計算で見られるトライ＆エラーによるシミュレーションは一般にパラメータの決定，ジョブスクリプト生成，ジョブ投入，終了待ちの繰り返しと見なす事が出来る．これらの操作を行うために，Xcrypt ではジョブ実行インターフェイスが以下の三つの部分から構成されている：

- パラメータの決定：
  @jobs=prepare(%template)
- ジョブオブジェクト生成，ジョブ投入：
  submit(@jobs)
- ジョブ終了待ち：**sync(@jobs)**

Xcrypt ファイルは上記の関数を含み，解析部分は perl によって記述される．このようにしてつくられたファイルを**表 1** 中の実行コマンドによって実行する．また，Xcrypt が提供するその他のコマンドについても表 1 に挙げる．

Xcrypt のジョブオブジェクトは**図 5** にあるのように作成される．’exe0’ は実行ファイルであり，’copied-file1’ 等はジョブ実行ディレクトリにコピーしたい実行ファイルやインプットパラメータが書かれたファイル等である．

```
%template = (
  'id' => "job$ job_id",
  'exe0' => './goupd_sx.sh',
  'copiedfile1' => 'goupd_sx.sh',
  'copiedfile2' => 'gaugeprm.001',
);
```

図 5 ジョブオブジェクトの例

3.3 実　　装

本実装ではパラメータの決定，ジョブスクリプト生成，ジョブ投入，終了待ちを組み合わせた以下の関数を用いる：

@jobs=&prepare_submit_sync (%template);

Xcrypt コードには，この他にジョブ実行に必要な情報である%template を Xcrypt に渡す操作の部分がある．また，perl で記述する部分として，インプットファ

```
use base qw (sandbox core);
for($job_id=0;$job_id<$job_num;$job_id++){
  # βの設定 (インプットファイルの生成)
  open Inputfile, "> gaugeprm.001"
  printf Inputfile "%lf\n", $beta;
  ...
  close Inputfile;

  %template = (
    'id' => "job$ job_id",
    'exe0@'
      => sub {"goupd_sx_b_$VALUE[0].sh" },
    'RANGE0' => [0, 1],
    'copiedfile1@'
      => sub {"goupd_sx_b_$VALUE[0].sh"},
    'copiedfile2' => 'gaugeprm.001',
  );
  @jobs=&prepare_submit_sync (%template);

  # ヒストグラムの評価，統計処理
  # （シミュレーション結果の解析）
  ...

    # 条件分岐 1,2（次のパラメータの値の決定）
  ...

} # end of job_id
```

図 6 本研究で実装した Xcrypt コードの概要

表 2 実験環境（T2K-Tsukuba）

cpu	Quad-Core AMD Opteron
	8356 (4-core x 4 socket)
	2.3GHz
OS	Red Hat Enterprise Linux v.5 WS
メモリ容量	32GB
compiler	GNU Fortran compiler

表 3 シミュレーションパラメータ

問題サイズ	$24^3 \times 4$
N_{sweep_max}	100
β の初期値	5.6800

イルの生成，異なる初期条件から得られた結果を判定し合わせる箇所，次のシミュレーションのインプットパラメータの値の決定を行う箇所もある．本研究で実装した Xcrypt の概要を**図 6** に示す．%templete 部分において重要となるのはジョブの同時実行に関係する部分である．ジョブの同時実行は'RANGE0' によって指定されており, 同時に実行されるジョブ毎に異なるファイルを参照する場合には'@' の表記を用いる．'exe0@' や'copiedfle1@' がそれに該当する．本実装では，同時に実行されるジョブは初期条件が異なる．これらはコマンドライン引数で指定される．図 6 で指定される goupd_sx_b_$VALUE[0].sh はジョブ実行のためのスクリプトであり，予め用意しておく．ただし，$VALUE[0] は'RANGE0 'の値を参照するようになっており，今回には 0, または 1 をとる．この値が異なるとスクリプト中のコマンドライン引数が異なって，初期条件が変更されるようにした．図 6 のテンプレート部分と図 5 を比較してわかるように，Xcrypt では容易にジョブの同時実行が可能である．ただし，本実装では図 2 の条件分岐 1 までの繰り返し部分に対し，複数ジョブの同時実行を実装した．これに加えて, 図 2 中の 1 ジョブに対して初期条件の異なるジョブを 2 ジョブだけ投入するようにした．現時点では，特定の領域の全ての P に対してヒストグラムが 1 個以上の 0 でない値を持つことを確認し，ジョブの同時投入に加え，初期条件に依存した部分を取り除く作業までを容易に自動化できることを確かめることが目的である．今回はテストとして $job\_id < 3$ とした．

4. 評　　価

4.1 評価環境と結果

本実験は筑波大学計算科学研究センターのスーパーコンピュータ T2K-Tsukuba(http://www.open-supercomputer.org/) 上の少数ノード群を用いて行った．Xcyrpt は現在，筑波大学・東京大学・京都大学の全ての T2K システム上で動作可能であるが，本実験は筑波大学に設置されている T2K-Tsukuba 上で行った．実験環境を**表 2** にまとめる．また，シミュレーションパラメータを**表 3** に示す．

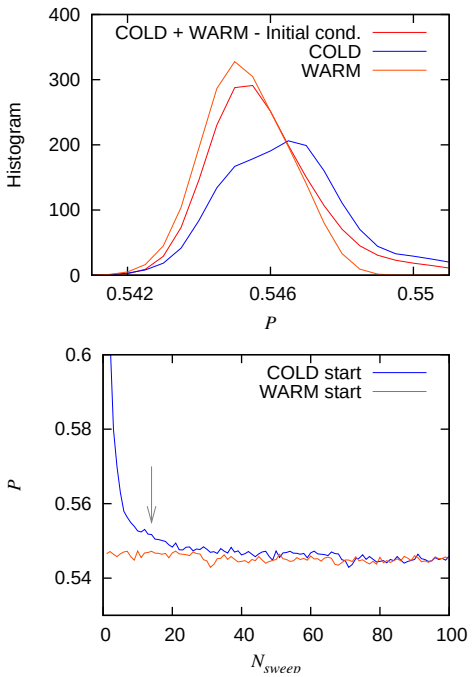

図 7 異なる初期条件でのヒストグラムとそれらのデータを合わせて作られた $\beta = 5.68$ でのヒストグラム（上）とアップデート過程（下）

初期条件に依存したデータを取り除く事ができているかについて確認した．**図 7** では $\beta = 5.68$ に対するヒストグラムを示し，初期条件の異なるグラフを別々にプロットしている．ヒストグラム中の'COLD'，及び'WARM' は初期条件の違いを示す．また，'COLD+WARM-initcond.' はこれらのデータを合わせたものから初期条件に依存するものを除いたデータのヒストグラムである．図 7 下はアップデート過程における P の値を示している．本実装では初期条件の異なるジョブでの各アップデートにおける観測量 P の値の差が 0.02 以下に一度なったところで初期条件の依存性が消えたと見なしてデータを合わせることにした．ここでは図中の矢印以降のデータのみを用いてヒストグラムを作成した．図 7 から，初期条件依存性が残っていることが分かる．

結果として，**図 8** が得られた．Xcrypt の条件分岐により選択された β は 5.684, 5.688 であり，範囲内の全ての P に対してヒストグラムに 1 個以上の（0 でな

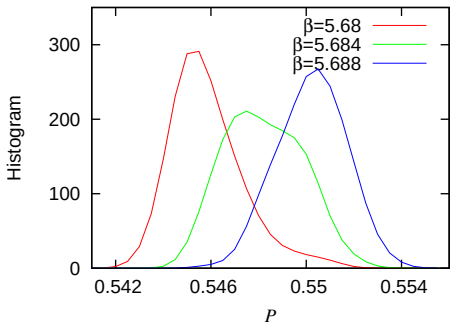

図 8 Xcrypt によって自動で生成されたヒストグラム

い）値を得ることができた．これより，β の統計精度に応じて，次の β の値が正しく設定できていると言える．一次相転移の特徴であるダブルピークの構造はまだ十分には見られていないが，このシミュレーションを繰り返し行うことで，取りこぼす領域なくパラメータ空間全体を探索することができるはずである．したがって，フローチャート図 2 で表されるシステムによりパラメータ空間を自動で探索できることがわかった．

4.2 考　　察

今回の実装では今後に向けて課題が得られた．まず，相転移点を特定できるように修正しなければならない．今回の実装では相転移点を特定できなかった．図 1 ではダブルピークの兆候（図の緑と青）が見られているのに対して，図 8 にはそのような形状ははっきりと見られないことがそれに対応する．これは各 β での統計が十分ではなかったためと考えられる．今回はジョブの同時実行が主な目的であったため，統計が十分にない場合にシミュレーションを繰り返すという操作については未実装となっている．特に，図 8 の $\beta = 5.684$ の結果（緑）では二瘤の兆候が見られており，相転移点での振る舞いが見られている．この点での統計を増やすような条件分岐を設定する事が相転移点を特定することができると考えられる．特に，二瘤の片方が成長する場合が考えられるので，それに対応して，$\beta = 5.68$ と 5.684 の間（緑と赤の間），もしくは $\beta = 5.684$ と 5.688 との間（緑と青の間）のシミュレーションを行い，どちらがより顕著な特徴を持つかによってさらに探索を繰り返す必要があるだろう．次回の実装では N_{sweep_max} をより大きな値に設定すると共に，それまで行ってきたジョブを必要に応じて延長するような条件分岐を正確に取り入れていきたい．

さらに，より多くのジョブの同時投入についても検討が必要である．まず，より多くの異なる初期条件を用いたジョブの実装が考えられる．並列度を高めて，より多くの統計を得ることができ，さらには，初期条件に依存しないデータを得るためにも有効である．しかしながら，並列度を高めると捨てるデータも多くなってしまうので，その点は留意する必要がある．初期条件に依存する配位が十分少ない場合には並列度を高めることも有効と考えられる．また，図 7 から分かるように，初期条件の異なる二つのデータを合わせて得られた'COLD + WARM - Initial cond.' もまだ'COLD' の初期条件に依存したデータの影響を含んでいるように見える．具体的には図 7 の P が大きい領域で COLD + WARM - Initial cond.' が大きな値となっている特徴である．初期条件の依存性が消えると判断できる基準については今後調整する必要があると考えている．

5. おわりに

小規模格子 QCD シミュレーションにおけるパラメータサーチ問題を，スクリプト言語 Xcrypt によって自動化し，同システムが複数ジョブの同時実行と複雑なパラメータサーチ適応可能であることを示した．これを用い，前回のシミュレーション結果から毎回のジョブ投入のパラメータと初期条件をヒューリスティックに考え，人手で投入するという従来手法を自動化し，スーパーコンピュータの複数リソースを用いて高速化するための基本プログラムを実現した．Xcrypt とその基本となっている perl 記述により，処理全対のフローチャートを比較的容易にジョブ制御記述に実装できることがわかり，本シミュレーションの基本部分が実現された．そして，Xcrypt の機能を用いて，ヒューリスティックな操作の一部を自動化することに成功した．

今後の進展として，複数の β に対するジョブの同時実行を検討したい．具体的には**図 9** のような構成である．β は分布が重ならないように選べばよく，各々のジョブには直接的な依存関係はないので並列化を行うことが可能である．もちろん，完全には独立ではないので β を自由に選べる訳ではない．その点を注意して条件分岐を指定しなければならない．

また，自動化によるシミュレーション全体の効率化を実現したいと考えている．自動化により，統計量不足を補うためのジョブの繰り返しを比較的容易に行うことができるため，N_{sweep} の回数を，人手で行う場合よりも小さく設定することが可能となる．その結果として，十分な統計量を得るための N_{sweep} に対する無駄を省くことができ，効率化が図れると考えている．

さらに，シミュレーション全体の更なる自動化とモンテカルロ法部分の並列化を検討している．計算の主

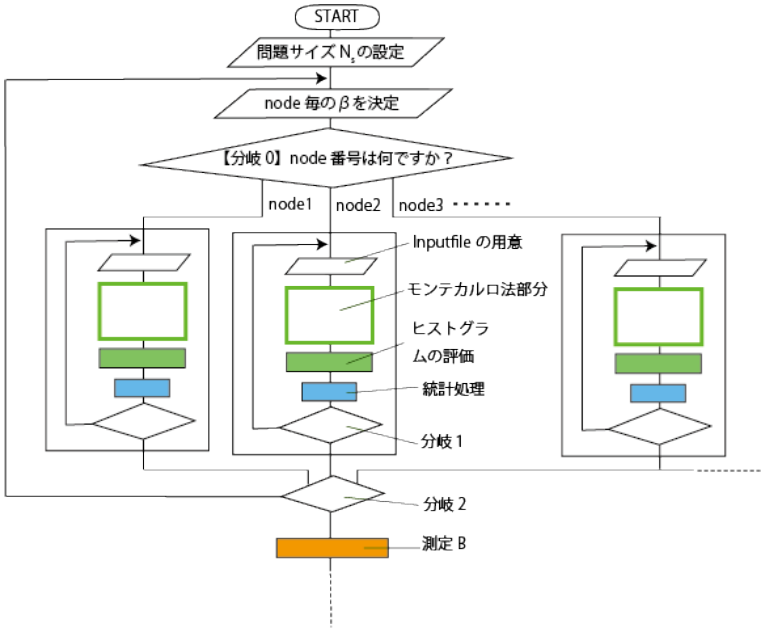

図 9 複数ノードで実行した場合のフローチャート

体はモンテカルロ法のシミュレーションであり，比較的小規模だが，並列性は十分にある．そこでマルチコア構成の計算ノードを用いて計算を加速する．本研究で対象としている QCD シミュレーションの個々のモンテカルロ法処理は小規模であるため，T2K 程度のマルチコア環境では OpenMP によるスレッド並列程度で十分実用的な実行速度が得られるため，MPI 並列は用いず，これらのスーパーコンピュータをノード単位で多数同時に用いるようなシミュレーションを考えている．最終的には全シミュレーションを完全自動化・並列化し，同種の研究を推進するツールとして役立てて行きたい．

謝　　辞

本研究を進めるにあたり，Xcrypt の使い方，動作確認等をご指導頂いた，京都大学学術情報メディアセンター中島浩教授並びに，平石拓助教に深く感謝致します．

参 考 文 献

1) Ken-ichi Ishilkawa: Recent algorithm and machine development for lattice QCD, [arXiv:hep-lat/0811.1661]
2) K. Kanaya : Finite Temperature QCD on the Lattice – Status 2010, [arXiv:hep-lat/1012.4247]
3) H. Saito et al. : Phase structure of finite temperature QCD in the heavy quark region, [arXiv:hep-lat/1106.0974]
4) Xcrypt チュートリアル，https://www2.cc.u-tokyo.ac.jp/procon2010/Tutorial/Xcrypt.pdf
5) 平石 拓, 安部 達也, 三宅 洋平, 岩下 武史, 中島 浩 : 柔軟かつ直観的な記述が可能なジョブ並列スクリプト言語 Xcrypt, 先進的計算基盤システムシンポジウム（SACSIS2010）, P.183-191
6) Xcrypt manual, http://super.para.media.kyoto-u.ac.jp/xcrypt/Xcrypt_manual.pdf
7) T2K-Tsukuba システム, http://www.ccs.tsukuba.ac.jp/CCS/t2k-tsukuba