公立はこだて未来大学 2015 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2015 System Information Science Practice

Group Report

プロジェクト名

FUN-ECM プロジェクト

Project Name

FUN-ECM Project

グループ名

グループ A

Group Name

Group A

プロジェクト番号/Project No.

15-A

プロジェクトリーダ/Project Leader

1013186　山本健太　Kenta Yamamoto

グループリーダ/Group Leader

1013186　山本健太　Kenta Yamamoto

グループメンバ/Group Member

1013044　石川夏樹　Natsuki Ishikawa
1013049　小野嘉翔　Yoshiharu Ono
1013052　九島拓実　Takumi Kushima
1013157　土田祐介　Yusuke Tsuchida
1013186　山本健太　Kenta Yamamoto
1013203　辻田陸　Riku Tsujita

指導教員

白勢政明　由良文孝

Advisor

Masaaki Shirase　Fumitaka Yura

提出日

2016 年 1 月 20 日

Date of Submission

January 20, 2016

概要

私たちのプロジェクトの目的は、大きな桁数の素因数を見つけることである。現在のインターネットの通信において、公開鍵暗号の 1 つである RSA 暗号が広く利用されている。RSA 暗号は、巨大な数の素因数分解が困難であることを利用した暗号である。よって、RSA 暗号の安全性は、巨大な数の素因数分解の困難さで評価される。よって、素因数分解のチャレンジは、重要なテーマとなっている。

素因数分解の優れたアルゴリズムの 1 つに楕円曲線法がある。また、「ECMNET」という楕円曲線法によって大きな桁数の素因数を見つけることを目的とするサイトがある。現在記録されている素因数よりも大きな素因数を見つけることで、ランキングに名前を載せることができる。私たちは、このサイトのランキングに載っている素因数よりも大きな素因数を見つけたい。

大きな桁数の素因数の発見のために、楕円曲線法のアルゴリズムでプログラムの高速化につながるような理論を学習する理論班、理論班が学習した高速化につながる手法を実装するプログラム班に分かれて活動を行った。

理論班は、プログラムの計算量の削減を目標とした。楕円曲線法のプログラムは加算公式の計算を繰り返し行うため、加算公式の計算コストを減らすために活動をした。射影座標系を導入した場合、エドワーズ曲線の加算公式は、一般的な楕円曲線の加算公式よりも計算コストが 15 %削減できることを理論上証明した。

プログラム班は、エドワーズ曲線を用いた楕円曲線法のプログラム実装を目標とした。素因数分解では大きな数を扱うために、任意精度演算ライブラリの GMP を導入した。また、OpenMP を用いた並列処理を導入した。OpenMP とはマルチスレッド並列プログラミングのための API である。これにより多くの異なる楕円曲線を並列実行することができるようになった。作成したプログラムは Xeon Phi 上で実行している。Xeon Phi とは Intel 社で開発された 60 個のコアをもつコプロセッサーである。

理論班とプログラム班の活動により、昨年度のプログラムよりも高速なプログラムが完成した。

キーワード　楕円曲線法, 素因数分解, ECMNET, Xeon Phi, エドワーズ曲線, 射影座標系, GMP, OpenMP, RSA 暗号

（※文責: 小野嘉翔）

Abstract

The purpose of our project is to find prime factors as large as possible. RSA cryptosystem has been widely used for the Internet communication all over the world. RSA cryptosystem is one of the public key encryption. RSA cryptosystem is based on the difficulty of prime factorization of large number. Therefore, the security of RSA cryptosystem is evaluated by prime factorization of large number. Therefore, it is a important theme to try to do prime factorization of large number.

Elliptic Curve Method(ECM) is one of the best algorithms for prime factorization. ECMNET is a website to find large factor by using ECM. This website records anyone's name who gets larger factor. We would like to discover a larger factor than current record of ECMNET.

Our project is divided into two teams, theory team and program team for the sake of finding large prime factorizations. The theory team learns the theory that makes algorithm of ECM faster. The program team implements an ECM program which is based on the algorithm the theory team learns.

The purpose of the theory team is to reduce computational complexity of the ECM program. The ECM program repeats process of addition law many times over. Therefore, the theory team reduces the computational cost of addition law. The theory team proves that computational cost of additional law of the Edwards curve is 15 percent faster than one of the general elliptic curve in the case where projective coordinate is used.

The purpose of the program team is to implement the ECM program using Edwards curve. The program team introduces the GMP and OpenMP to the ECM program. The GMP is an arbitrary precision arithmetic library. Many different ECM programs can be executed by OpenMP in parallel. We run the ECM program on Xeon Phi. The Xeon Phi is a co-processor which has 60 cores and is developed by Intel Corporation.

We make the more faster ECM program than program of last year.

Keyword Elliptic Curve Method, prime factorization, ECMNET, Xeon Phi, Edwards curve, projective coordinate, GMP, OpenMP, RSA cryptosystem,

（※文責: 小野嘉翔）

目次

第 1 章　プロジェクトの目的　1

1.1　プロジェクトの背景 1

1.2　ECMNET の説明 1

1.3　プロジェクトの最終目標 1

第 2 章　プロジェクトの概要　2

2.1　問題の設定 2

2.2　課題の設定 2

2.3　到達レベル (目標) 3

2.4　課題の割り当て 3

2.4.1　前期 3

2.4.2　後期 3

第 3 章　学習成果　4

3.1　前期 4

3.1.1　基礎学習 4

3.1.2　中間発表 6

3.2　後期 7

3.2.1　射影班 8

3.2.2　プログラム班 10

3.2.3　統計班 20

3.2.4　最終発表 23

第 4 章　プロジェクト内のインターワーキング　25

第 5 章　結果　28

5.1　前期 28

5.1.1　成果物 28

5.1.2　プログラム 28

5.2　後期 30

5.2.1　成果物 30

5.2.2　射影班 30

5.2.3　プログラム班 30

5.2.4　統計班 31

5.3　解決手順と評価 31

5.3.1　前期 31

5.3.2　後期 31

第 6 章　まとめ　33

6.1 プロジェクトの成果 33
6.1.1 前期 33
6.1.2 後期への課題 33
6.1.3 後期 33
6.1.4 今年度の成果 33
6.2 プロジェクトにおける各人の役割 34
6.2.1 前期 34
6.2.2 後期 34
6.3 今後の課題と展望 34

付録 A 新規習得技術 **36**

付録 B 相互評価 **37**

付録 C その他製作物 **39**

参考文献 **41**

第 1 章　プロジェクトの目的

本プロジェクトの目的は、楕円曲線法 (Elliptic Curve Method) を理解し、そして楕円曲線法を用いてできるだけ大きな素因数を見つけ、「ECMNET」に登録することである。

（※文責: 山本健太）

1.1　プロジェクトの背景

現在のインターネットの通信において、公開鍵暗号の 1 つである RSA 暗号が広く用いられている。RSA 暗号は、巨大な数の素因数分解が困難であることを利用した暗号である。よって、RSA 暗号の安全性は、巨大な数の素因数分解の困難性によって評価される。よって、素因数分解のチャレンジは、重要なテーマとなっている。素因数分解の優れたアルゴリズムの 1 つに楕円曲線法がある。楕円曲線法を研究しより大きな桁数の素因数分解にチャレンジすることでインターネットのセキュリティに貢献することができる。より大きな桁数の素因数分解に挑むために、楕円曲線法による素因数分解のプログラムを並列処理できるようにし、高速化できるようにすることを目指す。

（※文責: 山本健太）

1.2　ECMNET の説明

ECMNET[1] とは楕円曲線法で求められた素因数の桁数を競うコンペティションである。登録されている素因数よりも大きな桁数の素因数を見つけることで ECMNET のランキングに登録することができる。2016 年 1 月 5 日時点で ECMNET に登録された素因数の最大の桁数は 83 桁で、今年度登録されている素因数の最大桁数は 71 桁である。

（※文責: 山本健太）

1.3　プロジェクトの最終目標

本プロジェクトは ECM プログラムを改良し並列化と高速化を実装し、ECMNET への登録を目的としている。今年度 ECMNET に登録された素因数の最大の桁数の 71 桁であり、ランクインには 64 桁以上の素因数の発見が必要となる。最終目標の ECMNET へのランクインのために 64 桁以上の素因数の発見を目指す。

昨年度の同プロジェクトでの反省点 [2] として、プログラムを実行する時間が十分ではなかったこと、楕円曲線法のアルゴリズム改良のために参考にする文献が少なかったこと、そしてプログラムに実装できなかった手法があったことが挙げられている。目標達成のために、昨年度の反省点を活かし、計画的な作業予定を設定して文献調査やコーディングを進めていく必要がある。

（※文責: 山本健太）

第 2 章　プロジェクトの概要

2.1　問題の設定

本プロジェクトの目標を達成するにあたり、以下を主な問題として取り上げることとした。

- メンバ全員の楕円曲線法の理解
- ECMNET の現状調査
- ECM プログラムの実装
- プログラムの改良
- プログラムの並列化
- 昨年度のプログラムと今年度のプログラムの比較評価

（※文責: 石川夏樹）

2.2　課題の設定

楕円曲線法は最新の暗号技術の安全性評価において必須の素因数分解法である。また、私たちが学習することに適した難易度であり、比較的速く素因数分解を行うことが可能である。さらに、プログラムに実装する際、並列化が容易であることから、楕円曲線法の学習とプログラムへの実装を本プロジェクトの主な課題とした。しかしメンバ全員の楕円曲線法の知識が不足していたため、プログラムの学習に入る前に、前期はまず楕円曲線法を学習することとし、以下を主な学習内容として設定した。

- 楕円曲線の基本
- 剰余について
- $\mathbb{Z}/n\mathbb{Z}$ の四則演算
- 位数
- ラグランジュの定理と点の位数
- 無限遠点と因数発見の関連について

また、後期では射影班、プログラム班、統計班に分かれ、それぞれのグループで学習を行った。後期に設定した課題を以下に示す。

- 楕円曲線の式の変更
- 射影変換の学習
- 昨年度作成されたプログラムの改良
- プログラムの並列化
- 統計の取り方の学習
- プログラムの比較に使用するサンプルデータの採取
- プログラムの比較評価

（※文責: 石川夏樹）

2.3 到達レベル (目標)

課題を実行することにより、メンバ全員が出来るだけ早く楕円曲線法を理解する。また、ECM プログラムの改良を行い、より高速にプログラムを実行できるようにする。最終的には ECMNET への登録を目標とする。

（※文責: 石川夏樹）

2.4 課題の割り当て

2.4.1 前期

楕円曲線法を学習するにあたり、前期は担当教員である白勢先生に準備していただいたカリキュラムに沿い、メンバ全員で 2.2 節の内容の基礎学習を行い、楕円曲線法のアルゴリズムを理解することとした。中間発表の準備の際、今まで学習した内容を分かりやすく説明できるようにメンバ全員で復習した。また、全く予備知識のない人に対して発表を行うため、発表に使用するスライドと原稿を細部まで見直し、伝えるべき内容を噛み砕いて作成した。

（※文責: 石川夏樹）

2.4.2 後期

後期は射影班、プログラム班、統計班に分かれ、それぞれのグループに課題を割り当てることとした。射影班とプログラム班がプログラムの改良と高速化を行い、統計班が昨年度作成されたプログラムと今年度作成されたプログラムの比較評価を行った。最終発表の準備では中間発表の反省を活かし、活動内容だけでなく社会との関連性や、活動で得られた結果を重点的にまとめた。また、使用するスライドを簡潔にまとめ、グラフや図を取り入れることで視覚的に理解しやすくなるよう工夫した。

（※文責: 石川夏樹）

第 3 章　学習成果

3.1　前期

前期は、メンバ全員が楕円曲線法についての理解が乏しかった点から、担当教員の白勢先生・由良先生による授業計画に従い、楕円曲線法のアルゴリズムや基礎知識などを学んだ。また、PARI/GP[3] を使い前期で学習してきた内容が実際にどのように動作するのかを確認するための簡単な ECM プログラムを作成した。プロジェクトメンバ全員で楕円曲線法の基礎的な理解という課題に取り組むことで、お互いに教え合ったり、不明点を共有することができ、プロジェクトメンバの理解度の向上につながった。

（※文責: 山本健太）

3.1.1　基礎学習

楕円曲線法とは

楕円曲線法とは素因数分解の最良な方法の一つである。楕円曲線法は $y^2 = x^3 + ax + b$ で表される式を用いて、a と b のパラメータにより位数が変化することを利用して短い時間で約数を発見できる。異なる a と b の処理を独立に実行できるため、並列処理にも向いているという側面も持ち合わせている。

（※文責: 山本健太）

楕円曲線

3 次式

$$E\colon y^2 = x^3 + ax + b$$

等で与えられる曲線 E を楕円曲線という。楕円曲線 E の特徴として、点 $P, Q \in E$ に対して以下の操作により $P + Q \in E$ が定義されることである。

1. P, Q に対して 2 点を通る直線を L とする。
2. 楕円曲線 E と直線 L が交わる第三の点を $P * Q$ とする。
3. $P * Q$ を通る y 軸に平行な直線と楕円曲線 E が交わる点を $P + Q$ とする。

この $+$ を Q を $P + Q$ に置き換えて同じ操作を繰り返すことで、

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ 個の和}}$$

が定義される。nP を求めるこの操作をスカラー倍と言う。直線 L が y 軸に平行な場合、$P * Q$、$P + Q$ が無限遠点となる。

$P + Q$ は P と Q の座標からも計算できる。P の座標を (X_1, Y_1)、Q の座標を (X_2, Y_2) としたとき、2 点を通る直線の式を $L{:}y = \lambda x + v$ とすると、$P \neq Q$ の場合は $\lambda = \frac{(Y_2 - Y_1)}{(X_2 - X_1)}$、$P = Q$

の場合は $\lambda = \frac{(3X_1^2+a)}{2Y_1}$ で表される。すると、$P*Q$ の座標 (X_3, Y_3) は $X_3 = \lambda^2 - X_1 - X_2$ 、$Y_3 = Y_1 - (X_1 - X_3)\lambda$ と表される。$P+Q$ の座標は $P*Q$ の x 軸に対称な点であるので $(X_3, -Y_3)$ となる。

（※文責: 山本健太）

剰余について

ある整数 a をある自然数 n で割った時の非負 (正の整数) の余り r を $r = a \bmod n$ のように表す。$0 \cdots n-1$ の範囲の整数の集合を $\mathbb{Z}/n\mathbb{Z}$ とすると、0 または r は $\mathbb{Z}/n\mathbb{Z}$ の元と対応付けることができる。

（※文責: 土田祐介）

$\mathbb{Z}/n\mathbb{Z}$ の四則演算

$\mathbb{Z}/n\mathbb{Z}$ における加算、減算、乗算は、実数と同じような計算の後 $\bmod n$ を求める操作を行う。除算においては、ある数に割る数の逆元をかけることに等しい。$\mathbb{Z}/n\mathbb{Z}$ での整数 a における逆元が存在するための必要十分条件は $\gcd(a, n) = 1$、つまり a と n が互いに素であることである。また、n が素数ならば $\mathbb{Z}/n\mathbb{Z}$ の 0 以外の元はそれぞれの逆元を持つ。

（※文責: 土田祐介）

位数

$\mathbb{Z}/n\mathbb{Z}$ における無限遠点を含めた楕円曲線上の点の集合は有限である。ある点 P での $+$ の操作、$P+P = 2P$,$P+2P = 3P$ という操作を繰り返し行うと操作後の点が無限遠点となる m が存在する。この時、m を P の位数と呼ぶ。

（※文責: 山本健太）

ラグランジュの定理と点の位数

ラグランジュの定理

有限群 G に対して、$G \ni a$ の位数は、$\#G$ の約数になる。

楕円曲線 $E : y^2 = x^3 + ax + b$ の係数や点の座標を $\mathbb{Z}/n\mathbb{Z}$ で考えるものを $E(\mathbb{Z}/n\mathbb{Z})$ と表す。n が素数のとき、$E(\mathbb{Z}/n\mathbb{Z})$ は有限群となるのでラグランジュの定理を適用でき、楕円曲線のどの点 P も位数は $\mathbb{Z}/n\mathbb{Z}$ 上の楕円曲線 E の点の個数 $\#E(\mathbb{Z}/n\mathbb{Z})$ の約数となることがわかる。また、$\#E(\mathbb{Z}/n\mathbb{Z})$ は a と b の値を変化させると変動する。よって、楕円曲線 E の a と b の値を上手く取れると、点の位数が比較的小さな素数同士の積になるものがある。

（※文責: 小野嘉翔）

楕円曲線法の手順

n を素因数分解したい合成数とする。

1. $a, b \in \mathbb{Z}/n\mathbb{Z}$ を選んで、楕円曲線 $E : y^2 = x^3 + ax + b$ に対して $P = (x, y) \in E(\mathbb{Z}/n\mathbb{Z})$ を 1 つとる。
2. 適切な自然数 k を選ぶ。
3. $(k!)P$ を計算する。
4. この計算の途中で、$\lambda = (y_1 - y_2)/(x_1 - x_2)$ または $\lambda = (3x_1^2 + a)/2y_1$ を求める部分が生じるが、$x_1 - x_2$ または $2y_1$ が正則でなければ計算は続行不可能となる。
5. しかし、計算が続行不可能ならば、$\gcd(x_1 - x_2, n) \neq 1$ または $\gcd(2y_1, n) \neq 1$ なので、n の約数が求まる。
6. $(k!)P$ が計算できてしまったら n の約数は求まらず、a, b を変えてステップ 1 からやり直す。

上記の手順で、ステップ 4 が続行不可能となる場合を説明する。p を n の素因数の 1 つとする。ステップ 1 で選んだ $P = (x, y)$ に対して、$P_p = (x_p, y_p) \in E_p(\mathbb{Z}/p\mathbb{Z})$ とする。$l = \#E_p(\mathbb{Z}/p\mathbb{Z})$ の素因数分解を

$$l = p_1^{e_1} \cdot p_2^{e_2} \cdot \cdots \cdot p_t^{e_t}$$

とする。もし、

$$p_i^{e_i} \leq k \quad (i = 1, 2, \cdots, t)$$

ならば、ラグランジュの定理より

$$(k!)P_p = \text{無限遠点}$$

となる。つまり、$(k!)P_p$ の計算過程の途中のどこかで無限遠点が生じる。この時、$x_{1p} = x_{2p}$ または $y_{1p} = 0$ であり、$x_1 - x_2$ または y_1 は p の倍数、つまり $\mathbb{Z}/n\mathbb{Z}$ の非正則元となる。よって、ステップ 4 が続行不可能となり、n の約数が求まる。

a, b を変えて上記手順を繰り返すと、$p_i^{e_i} \leq k \quad (i = 1, 2, \cdots, t)$ を満たすような、つまり n の約数の発見に成功するような、$E : y^2 = x^3 + ax + b$ が選ばれる。

このような E が選ばれる確率は、自然数の中での素数の割合から計算することができ、楕円曲法の計算コストを見積もることができる。

（※文責: 九島拓実）

3.1.2 中間発表

準備

- ポスター
  初めに、過去のポスターを参考に構成を決定した。次に、概要、活動内容、活動計画の 3 つに班をわけた。各班は 2 人ずつの構成とした。班はそれぞれ担当した箇所の文章とデザインを作成した。ポスターの制作には「Adobe Illustrator」というドローソフトを用いた。ポスターができると一度全員で見直し、誤字脱字やわかりにくい表現などを修正した。これを何度か繰り返すことで、より見やすくわかりやすいポスターが完成した。
- 原稿
  原稿制作は主に 2 つの班に分かれて行った。概要班は 2 人、学習内容班は 4 人とした。原稿はそれまで学習した内容を基に制作した。その際、ECM を全く知らない人にもわかりやすくするよう先生に相談しながら内容をまとめた。また、学習内容の量に比べて発表時間が短いため、スムーズに説明するために話す順序を調整したり、比較的重要でない箇所を省いたりした。

- スライド
  ポスターだけでは聴講者にうまく伝わらないのではということで、スライドを制作した。スライドは原稿を基に制作した。スライドの制作には「Prezi」[4] というプレゼンテーションソフトを用いた。また、一度完成したスライドを見直したところ、先生から一枚一枚の情報量が多すぎるという助言を受けた。そこで、文章の見直しや例を提示するなど大幅に改編した。これにより、より見やすいスライドが完成した。

（※文責: 辻田陸）

当日

中間発表では、前半と後半で 3 人ずつに分かれて発表を行った。発表場所と日当たりの関係から、スライドを映すスクリーンの位置をずらし、日光が当たらないように調整した。発表では各自原稿を担当した部分を読み上げることとなった。3 人のうち 1 人が原稿を読み上げ、その間もう 1 人はスライドと並行してホワイトボードに楕円曲線と数式を描き、さらにもう 1 人はスライドのページをめくる作業を担当した。

（※文責: 石川夏樹）

評価

発表の評価は五項目について五段階で評価したのち総合的に評価した。

- 目的 : 明確でわかりやすいと評価した人が多かったが、理解できなかった人が一部いた。(4 点)
- 現状把握 : メンバ側はきちんとプロジェクトを計画的に進められていることを理解していたが、うまくは伝えることが出来なかった。(4 点)
- 今後の計画性 : ある程度の方向性は決まっているが、細かい部分までは伝わっていなかった。(4 点)
- 表現力 : 内容が難解だったために伝わりづらい部分もあったが、Prezi とホワイトボードを使用したことで、多少なりとも理解してもらえる部分があった。(3 点)
- チームワーク : ハプニングが発生したことがあったが、慌てることなくみんなで協力して発表することが出来た。(5 点)
  以上の評価より私たちの発表の総合評価は五段階中 4 点とした。

（※文責: 九島拓実）

3.2 後期

後期では、プロジェクトの目的である ECMNET へのランクインのために必要な ECM プログラムの高速化と並列化の実現のために、ECM プログラムの改良と評価を行った。活動にあたって、プロジェクト全体を射影班、プログラム班、統計班に分け活動を行った。射影班は ECM プログラム内の計算量を減らすためにエドワーズ曲線を導入し射影変換を適用する取り組みを行った。プログラム班は高速処理できる ECM プログラムの実装と Xeon Phi 上で実行するための簡単なマニュアルを作成する取り組みを行った。統計班は今年度作成した ECM プログラムがどのくらい高速化

したのか評価をする取り組みを行った。班分けによって各メンバの学習予定や担当作業が散り散りになってしまいグループとしての統率が取りにくくなる恐れがあった。そこで、毎回のプロジェクトの開始と終了時に、各班のその日の活動予定、到達目標、活動反省を全体で共有し、プロジェクト全体で活動目標と問題意識をもって活動に取り組むようにした。それに伴って、後から確認できるように各班の議事録を作成した。

（※文責: 山本健太）

3.2.1 射影班

目的

射影班の活動目的は、ECM プログラム内の加算公式部分に射影変換を用いて計算量を減らすことである。具体的には今年度から採用したエドワーズ曲線を利用した ECM プログラム内の加算公式、2 倍算公式の中の計算コストを減らすことを目標とした。

（※文責: 九島拓実）

活動内容

まず昨年度のプログラムの計算コストを減らすために使用された射影変換に着目した。今年度のプログラムにも採用するためにまずは書籍、論文、白勢先生の講義等で射影変換について学習した。その後、各計算の計算コストを算出し、昨年度と今年度の ECM プログラム内の加算公式の計算コストを数え上げた。

（※文責: 九島拓実）

射影変換と新しい楕円曲線について

射影変換とは、2 次元の座標に対し、3 つの変数を用いて表現する手法のことである。

例えば、2 次元の座標 (x_1, y_1) が存在しているとき、

$x_1 = \frac{X_1}{Z_1}$, $y_1 = \frac{Y_1}{Z_1}$

といったように変換を行う。

加算公式内でこの操作を行うことで、逆元計算の回数を減らすことができる。逆元計算は他の演算に比べて非常に処理が遅いので、射影変換を用いることでプログラムの高速化を図ることができる。また、今年度からエドワーズ曲線と呼ばれる新しい楕円曲線を導入した。射影変換を用いた場合、昨年度まで使用していた式よりもプログラムを高速化することができる。

（※文責: 石川夏樹）

計算コストの算出

まず加算公式内で使われている計算の乗算、2 乗算、逆元計算のそれぞれの計算にかかる時間を調べた。これは C 言語環境下でそれぞれの計算を 1 億回行うプログラムを作成し、それらのプログラムにかかる時間を計測した。次にプログラムにかかった時間を乗算を基準に比較した。そして昨年度と今年度の射影変換前後の加算公式のプログラムに使用される各計算コストを数えた。この際に昨年度の射影変換前後のプログラムの計算コストについては、昨年度の報告書に記載があった

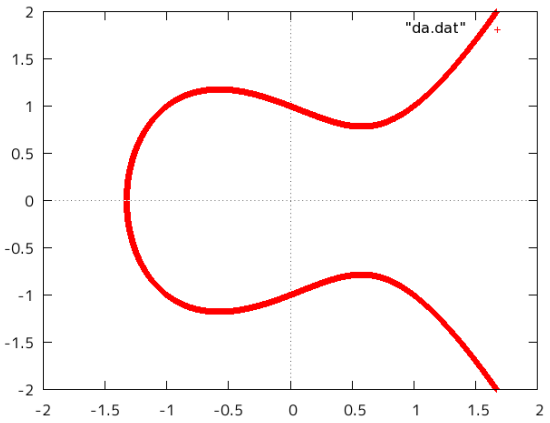

図 3.1　昨年度まで使用していたのヴァイエルシュトラス方程式： $y^2 = x^3 + ax + b$

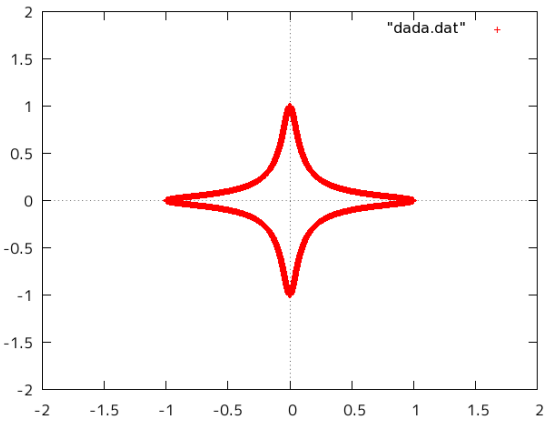

図 3.2　新しく導入したエドワード曲線： $x^2 + y^2 = 1 + dx^2y^2$

ためそちらを使用した。今年度の計算コストは実際にプログラムを確認して数えた。最後に、昨年度と今年度の加算公式の射影変換前後について、乗算の計算コストを M、二乗算の計算コストを S、逆元計算の計算コストを I として算出した。更にコストを比較しやすくするために M だけを用いる比較も行った。

（※文責: 九島拓実）

結果

計算コストを算出した際に調べた、各計算の計算コストは乗算を基準とした際に、二乗算は 0.8 倍、逆元計算は 27.5 倍であることが判明した。また、昨年度と今年度の加算公式一回分の計算コストの比較は以下のようになった。表記の際は乗算を M、二乗算を S、逆元計算を I とした。

- 昨年度射影変換前 ： I+2M+S ≒ 30.3M
- 今年度射影変換前 ： 2I+6M ≒ 61M
- 昨年度射影変換後 ： 12M+2S ≒ 13.6M
- 今年度射影変換後 ： 11M+S ≒ 11.8M

これらを全て M に統一して比較すると上記より、昨年度射影変換前が 30.3M、今年度射影変換前が 61M、昨年度射影変換後が 13.6M、今年度射影変換後が 11.8M となった。これらのことから射影変換前の計算コストに比べ射影変換後の計算コスト、昨年度の射影変換後のプログラムの計算コストに比べ今年度の射影変換後の計算コストがそれぞれ少なくなっていることがわかった。

（※文責: 九島拓実）

考察

射影班の計算コスト算出の結果から、射影変換を用いることでプログラムは高速化に成功したと考えられる。また、具体的には昨年度射影変換後 13.6M に対して、今年度射影変換後が 11.8M の計算コストであることから、加算公式一回に対し約 15% の高速化が図られたと考えられる。また、統計はとらなかったが、同様に二倍算の公式にも射影変換を用いたため、そちらでも高速化に成功したと考えられる。

（※文責: 九島拓実）

3.2.2　プログラム班

活動目的

プログラム班の活動目標は 2 つあった。1 つ目に、昨年度の ECM プログラムよりも高速に処理ができる ECM プログラムの実装を行うことであった。ECM プログラムを用いて大きな桁数の素因数を見つけようとした場合、プログラムを長期にわたって動かし続ける必要がある。また、ECM プログラムは、加算公式の計算処理を繰り返すことで素因数分解を行っている。よって、加算公式の計算処理コストを減らすことで、ECM プログラムの処理が高速になり、大きな桁数の素因数発見の効率化を図れると考えた。そこで、プログラム班は昨年度のプログラムの加算公式の計算処理部分を中心に改良を行うこととした。具体的には、後期に理論班が作成した射影座標系を導入したエドワーズ曲線を用いた楕円曲線法のアルゴリズムを基に、昨年度の ECM プログラムの改良を行った。2 つ目に、ECM プログラムを Xeon Phi 実行するための簡単なマニュアルの作成を行うことであった。ECM プログラムは C 言語で記述したが、OpenMP を導入し Xeon Phi 上で並列処理できるようなものにした。よって、Xeon Phi 上で ECM プログラムを実行するために、いくつかの特殊な操作を行う必要があった。そこで、特殊な操作の手順をマニュアルにすることで、誰でも Xeon Phi 上で ECM プログラムを動かせるようにした。

（※文責: 小野嘉翔）

活動内容

GMP の学習

GMP とは

GMP とは GNU Multi-Precision Library のことであり、暗号・インターネットセキュリティ・数式処理システムなどに使うことを目的とした任意精度演算を行える算術ライブラリである。GMP を用いることで、計算の桁数がハードウェアのメモリ容量以外に制限されなくなる。GMP はフリーソフトウェアとして公開されている。

目的

このプロジェクトでは、大きな桁数の素因数を見つけるために、数百桁もの合成数の素因数分解を行う必要がある。ところが、C 言語の基本データ型、例えば、int 型の場合だと最大で 10 桁の範囲までの数値しか扱うことができない。これでは、プロジェクトで行う素因数分解の合成数を扱うことが困難である。そこで、GMP ライブラリを導入することにした。GMP は、任意精度演算を行うことができるライブラリである。GMP を導入することで、扱える数値の桁数の制限が、システムのメモリの容量だけになる。これにより、このプロジェクトで用いる大きな桁数の数値を扱うことが可能になる。昨年度の ECM プログラムもこの GMP ライブラリを用いて、任意精度演算を行っていた。今年度も引き続き GMP ライブラリを用いて計算を行うことにした。よって、GMP ライブラリを用いた数値計算のプログラミング方法を学習する必要があった。

（※文責: 辻田陸）

GMP ライブラリのインストール

ECM プログラムは、Xeon Phi 上で並列処理させて動作させることとした。ただし、プログラムの作成、テスト等は、Xeon Phi 上ではなく、各自インストールしていた CentOS 上で行うこととした。理由として、プログラムの作成期間中は Xeon Phi を動かすことができる時間が限られていたため、プログラムのテスト等が効率良く行えないと判断したためである。また、CentOS 上できちんと動作すれば、Xeon Phi 上でも動作することが分かっていたためである。よって、始めに、並列処理化する前の ECM プログラムを作成し、それを CentOS 上でテストを行い、修正していくこととした。よって、GMP ライブラリを導入することから始めた。GMP ライブラリのインストールは Linux の yum コマンドを用いて行った。また、正常にインストールができたかを確認するために lgmp オプションで gcc コンパイルを行った。コンパイルがきちんと通れば、GMP ライブラリのインストールができたことが分かる。コンパイルが通ることを確認し、GMP の導入を終えた。

GMP の計算には独自の関数が多く用いられている。よって、C 言語の数値計算のプログラミング方法と異なるため、それぞれの関数の使い方を知るために、GMP のマニュアルを読み込み、分からないところは使い方が掲載されているサイト [5] を調べることとした。また、実際に GMP を用いた小規模の数値計算のプログラムを組み、学んだ内容を確かめながら GMP ライブラリのプログラミングの学習を進めた。

GMP ライブラリのインストールは、以下の環境で行った。

- ホスト OS: `Windows8.1 64bit`
- 仮想化ソフト : `VirtualBox 5.0.8 r103449`
- ゲスト OS: `CentOS 7.1 x86_64`

私たちの環境下では、GMP ライブラリがインストールされていなかったために、次の手順で CentOS7.1 に GMP ライブラリのインストールを行った。

1. 端末を開く
2. 端末上で、`yum list available | grep gmp` を実行する。実行後、私たちの環境では以下が表示された。

```
gmp.x86_64                  1:6.0.0-12.el7_1          @updates
gmp-devel.x86_64            1:6.0.0-12.el7_1          @updates
gmp-static.x86_64           1:6.0.0-12.el7_1          @updates
gmp.i686                    1:6.0.0-12.el7_1          updates
gmp-devel.i686              1:6.0.0-12.el7_1          updates
gmp-static.i686             1:6.0.0-12.el7_1          updatesx
```

   `x86_64` は 64bit 版、`i686` は 32bit 版である。私たちの環境は、64bit なので、64bit 版をインストールすることにした。
3. `yum -y install gmp.x86_64` を実行し、`gmp.x86_64` パッケージをインストールする。
4. `yum -y install gmp-devel.x86_64` を実行し、`gmp-devel.x86_64` パッケージをインストールする。
5. `yum -y install gmp-static.x86_64` を実行し、`gmp-static.x86_64` パッケージをインストールする。

これにより、CentOS7.1 上で、GMP ライブラリが導入できた。

（※文責: 小野嘉翔）

GMP ライブラリの基本的な使い方

GMP ライブラリを使った C 言語プログラムは、ヘッダファイル gmp.h をインクルードする必要がある。
#include ”gmp.h” によって、gmp.h をインクルードする。GMP ライブラリで、整数を格納する変数型は、 mpz_t である。変数は宣言した後、初期化が必要である。例えば、変数名が、example の整数型を初期化する場合は、

```
mpz_t ex;
mpz_init(example);
```

となる。
変数への数値の代入は、初期化した後に、

```
mpz_set_str(ex, “12345”,10);
```

を記述する。使わなくなった変数は、

```
mpz_clear(example);
```

で開放しなければいけない。
GMP の四則演算は、以下通りになる。
加算：a = b + c の場合

```
mpz_add(a, b, c);
```

減算：a = b - c の場合

```
mpz_sub(a, b, c);
```

乗算：a = b * c の場合

```
mpz_mul(a, b , c);
```

除算：a = b / c の場合

```
mpz_cdiv_q(a, b, c);
```

となる。
また、四則演算以外にも ECM プログラムに使用した関数として、
Mod: a = b mod c の場合

```
mpz_mod(a, b, c);
```

a に b と c の最少公約数を代入する場合

```
mpz_gcd(a, b, c);
```

がある。

（※文責: 小野嘉翔）

導入と学習

初めに、GMP を導入することから始めた。GMP のインストールは Linux の yum コマンドを用いて行った。正常にインストールができたかを確認するために lgmp オプションで gcc コンパイルを行った。コンパイルが通ることを確認し GMP の導入を終えた。
GMP の計算には独自の関数が用いられる。それぞれの使い方を知るために、GMP のマニュアルを読み込み、調べることとした。また、実際に GMP を用いた小規模のプログラムを組み、学んだ内容を確かめながら進めた。

（※文責: 辻田陸）

目的

楕円曲線法で素因数分解をしても、必ずしも素因数が見つかるとは限らない。昨年度は、多くの異なる楕円曲線で楕円曲線法を並列実行して素因数を発見する確率を向上させることを目的として並列処理を導入した。今年度もそれ引き継いで実装することとした。並列処理を行うために OpenMP を用いたプログラムを Xeon Phi 上で実行している。

Xeon Phi とは

Xeon Phi とは Intel 社が開発した 60 個のコアを持つコプロセッサーである。また、最大 240 スレッドの処理が可能、つまり 240 個の楕円曲線法での素因数分解を同時に行える。本プロジェクトで使用される Xeon Phi には 60 個のコアを持つコプロセッサー 5110P が二つと、64GB のメモリ、1TB SATA HDD が搭載されており、OS は Red Hat Enterprise Linux v6 である。

OpenMP とは

OpenMP とは Open Multi-Processing のことであり、共有メモリマルチプロセッサ上のマルチスレッド並列プログラミングのための API である。マルチコア CPU を持つコンピューター上で複数の処理をする。C 言語では、プラグマと呼ばれる指示文を既存の逐次プログラムに書き込むだけで、簡単に並列化できる。以下は for 文を並列化する際の例である。

```
#pragma omp parallel{
    #pragma omp for
    for(i=0;i<100;i++){
        //処理
     }
}
```

`#pragma omp parallel` に続くブロック文が並列化の対象となり、`#pragma omp for` に続く for 文で i の異なる値で処理が並列実行される。

（※文責: 辻田陸）

計画

プログラム制作に当たって、簡単な計画を立てた。以下にそれを記述する。

1. 昨年度のプログラムを、理論班から受け取ったアルゴリズムを基に書き換える
2. メンバの PC 上でテストを行う
3. 不具合があった場合、コードを修正し、2 へ戻る
4. Xeon Phi に実装し、何度かプログラムを動かす
5. 不具合があった場合、コードを修正し、4 へ戻る

6. 一時完成
7. インターフェース面の改良
8. 最終完成

コーディング

プログラム班は 2 人のみであり、かつ変更が必要な部分が小さいため、混乱を避ける目的で担当箇所を振り分けた。アルゴリズム担当とインターフェース担当に分かれ、それぞれが前半・後半でのコーディングをすることにした。

このプログラムでは点の加算の計算をすべて GMP の関数を用いて行う。四則演算や累乗、剰余の計算には GMP の専用関数が存在している。例えば、C 言語での「A=B+C」という加算の式は `mpz_add_ui(A, B, C)` というように書き換えることになる。アルゴリズム担当はこの書き換えを新しいアルゴリズムに適用し、コーディングした。またこのとき、可能な限り計算の数を減らすため temp 変数を用いた。この変数には途中の計算を保存し、数値を使いまわす役割がある。最後に、変数宣言、初期化、関数計算、解放が書かれているか確認し、説明が必要な項目にコメントアウトを載せてアルゴリズムのコーディングを終了した。

インターフェース担当は ECM に直接関係しないが今後の運用、テストがしやすくなることを目的としたコーディングをした。主な機能として、ループ機能を実装した。昨年度のプログラムでは一度素因数を発見するとそのまま終了していた。しかし、このままでは効率が悪いため、発見した素因数でない方の因数を新たな合成数とし再度プログラムを実行するようにした。この処理はプログラム実行時にオプションをつけることで行われる。最後に、説明が必要な項目にコメントアウトを載せてアルゴリズムのコーディングを終了した。

テスト

テストはメンバ 2 人で協力して行った。アルゴリズムのテストでは変数を表示させるプログラムを作り、異常な値をとっていないか観測することとした。また、適当な桁の素因数分解を行わせ正しいな答えが返ってくるか検証した。また、インターフェースのテストでは実装前後で結果が変わらないことを確認し正常と判断した。

テスト時にはコンパイルのときに Makefile を用いた。これは、Makefile という名前のファイルにコンパイルの手順を記入しておくことで、make コマンドを入力すればコンパイルの手間を減らすことができるものである。これは大きなプログラムになるほど効果的になる。Makefile の存在により、プログラム制作の効率が上がった。

プログラム仕様書

目的

プログラム班で、昨年度の ECM プログラムを読み、理解した機能を共有することと、プログラミングを円滑に行うために、昨年度に作成されたプログラム仕様書を基に作成した。

内容

内容は主に更新履歴、概要、関数一覧、マクロ一覧、構造体一覧から構成されている。各々がはたしている機能やその役割を説明したものを各項目ごとに記述を行った。まず、1 番最初に更新履歴を記載した。更新を行った日付、更新を行った担当者、更新した内容を記載した。次に、当プログラムの概要を記載した。 概要は、プロジェクトメンバが読んで理解できるような記述を心がけた。続いて、以下の関数の内容を記載した。

- 「normal_add 関数」
  加算する 2 点 P と Q が、「P != Q」のとき、射影座標系にて加算を行う関数である。

- 「double_add 関数」
  加算する 2 点 P と Q が、「P == Q」のとき、射影座標系にて 2 倍算を行う関数である。

- 「ecm 関数」
  実際に、ECM により素因数分解を行う関数である。

- 「scalar 関数」
  スカラー倍の計算を行うための関数である。

- 「afftopro 関数」
  直交座標系の点から射影座標系の点に変換を行うための関数である。

- 「protoaff 関数」
  射影座標系の点から直交座標系の点に変換を行うための関数である。

- 「affine_point_init」
  直交座標系の点 P の初期化を行うマクロである。

- 「affine_point_clear」
  直交座標系の点 P の開放を行うマクロである。

- 「affine_point_set」
  直交座標系の点 P を点 R に代入するマクロである。

- 「affine_point_cmp」
  直交座標系の点 P と点 Q の比較を行うマクロである。点が等しければ 0、等しくなければ 1 の値を返す。

- 「projective_point_init」
  射影座標系の点 P の初期化を行うマクロである。

- 「projective_point_clear」
  射影座標系の点 P の開放を行うマクロである。

- 「projective_point_set」
  射影座標系の点 P を点 R に代入するマクロである。

- 「projective_point_cmp」
  射影座標系の点 P と点 Q を比較を行うマクロである。等しければ 0、等しくなければ 1 の値を返す。
  最後に、 構造体を記載した。以下が記載された構造体である。

- 「projective_POINT」
  射影座標系の点を表す構造体。

- 「affine_POINT」
  直交座標の点を表す構造体。

（※文責: 小野嘉翔）

コーディング規約

目的

プログラム班でのプログラム作成を円滑に行うために、昨年度のコーディング規約を基に作成した。

内容

プログラムの命名規則やコーディングスタイル、その他重要と思われる点を網羅した。
演算子や制御文の記載方法等を規定した。
まず、命名規則を定めた. 以下がその内容である。
・変数の使用目的が分かるような変数名をつける。
・意味の区切りではアンダーバーを使用する。
・構造体名は大文字を使用する。
・関数は小文字を使用する。
・ループ変数は小文字一文字を使用する。
次に、コーディングスタイルを定めた. 以下がそのポイントである。
・変数宣言
・演算子
・制御文
・関数宣言
・インデント
・コメント
最後に、その他の重要項目を定めた.。以下がその内容である。
・意味の区切りでは一行空白を入れる
・R = P + Q を function(R, P, Q) と表記する。
・プログラムを更新したら、どこを変更したのかをコメントに分かりやすく書く
そして、これらを以てプログラム仕様書とした。

（※文責: 小野嘉翔）

実装

プログラムの制作・テストの段階では gcc でコンパイルしていた。しかし、Xeon Phi

上動かせるようにコンパイルするには、インテルが開発したコンパイラである icc を用いる必要がある。icc は基本的に gcc と同様のオプションを利用することができる。以下が icc コンパイルの方法である。

```
icc -mmic -openmp *.c -o funecm -lgmp -L/usr/local/lib/
```

オプションをそれぞれ解説すると、GMP ライブラリを指定する lgmp オプション、ライブラリの場所の指定をする L オプション、mic 向けにコンパイルするオプションである mmic オプション、OpenMP を利用するための openmp オプションとなる。これらのオプションを付けて icc コンパイルをすることで、Xeon Phi 上で並列実行可能なプログラムをコンパイルすることが出来る。

（※文責: 辻田陸）

運用

オンライン作業

Xeon Phi を使うためには、コンピューターが設置している部屋まで足を運ばなければならない。さらに、その部屋は入室時間が限られているため、非常に効率が悪い。そのため、コンピューターを学内ネットワークにつなげることにした。設定は GUI から行った。これによって、学内からならばいつでもコンピューターにアクセスできるようになり、作業効率が大幅に上がった。コンピューターへの接続には TeraTerm を用いた。ファイルのやりとりには WinSCP を用いた。

手順

ここでは、コンピューターへのアクセスからプログラム実行、終了までの手順を説明する。

初めに、TeraTerm を用いて SSH ログインをする。TeraTerm を起動し、ホスト欄に IP アドレスを入力、 サービスは SSH を選択しバージョンは SSH2 を選択する。OK を押すとユーザー名とパスワードを入力する欄が表示されるため、それぞれに適切な値を入力する。入力が正しければ、OK を押すとログインに成功する。学外からはアクセスできないため注意する。

次に、仮想 OS へプログラムなどをコピーする。Xeon Phi を使用するには仮想 OS でプログラムを実行する必要があるためである。まずは su コマンドで root ユーザーに切り替える。次に cd コマンドで ECM プログラムがあるディレクトリ (下記例では ecm2015_b) へ移動する。そうしたら、scp コマンドで OpenMP のライブラリである”libiomp5.so”と ECM プログラム本体である”funecm”、そして ssh 先で動的ライブラリを使用可能にするシェルの設定ファイルである”.profile”を mic0 へコピーする。シェルには「export LD_LIBRARY_PATH=/home/shirase/」が記述されている。また、mic0 とは Xeon Phi のことを示す。

コンピューターへのアクセスから仮想 OS へのコピーまで

```
$ su
$ cd /ecm2015_b
$ scp /opt/intel/composerxe/lib/mic/libiomp5.so /home/
  project8/ecm2015_b/funecm /home/project8/ecm2015_b/.profile
  mic0:/home/shirase/
```

次に、ssh コマンドで仮想 OS に SSH ログインする。このときパスワードを求められるので、適切に入力する。ログインできたら、cd コマンドで先ほどプログラムをコピーしたディレクトリまで移動する。そうしたら、export コマンドでパスを通す。これで ECM プログラムを実行する準備が整う。

仮想 OS へログインから実行準備まで

```
$ ssh mic0
$ cd /home/shirase
$ export LD_LIBRARY_PATH=.
```

ECM プログラムは以下の形式で実行する。

```
$ ./funecm [options] <composite number> <k> <filename> &
```

角括弧でくくられているものは任意のパラメータ、山括弧でくくられているものは必須の引数を意味する。引数のそれぞれの意味を解説する。options はオプションの種類を決める引数で、h でヘルプ呼び出し、l でループ処理を行うようにされている。composite number は素因数分解の対象の合成数を意味する。k は点のスカラー倍の量や回数を決定するのに使われる数値で、必ず 3 以上でなければならない。filename は実行結果を保存するファイル名を意味する。入力した名前を持つ txt ファイルが生成される。ECM プログラムの出力にはデバッグ用の処理を除き全て fprintf 関数が使われている。& はバックグラウンド実行することを指定している。

最後に結果の確認を行う。結果は txt ファイルに出力されているが、その量は膨大なものとなり、そのままでは素因数が見つかったかを調べるには大変な手間がかかる。従って、txt ファイルの中から grep コマンドを用いて指定の文字列が含まれる行を抜き出すことで手間を減らす。素因数が発見された時には”factor found” と出力されているので、それを検索するとよい。しかし、何度も”factor found” を入力するのは面倒なため、”@” も一緒に出力するようにした。”factor found” の代わりにこれで検索してもよい。

```
$ grep "factor found" result.txt     もしくは
$ grep "@" result.txt
```

仮想 OS から抜け出すときには logout コマンドを用いて終了する。su を終了し TeraTerm を閉じる際には exit コマンドを用いる。

（※文責: 辻田陸）

マニュアル

目的

Xeon Phi の操作は複雑な手順を要し、一から実行手順を調べようとすると大変な労力が必要となるだろう。他のプロジェクトメンバや次のプロジェクトのためにもわかりやすいマニュアルを作成することにした。

PDF ファイル

運用の項でまとめたような手順を記した。同時に、下記のスクリプトの利用方法も記した。同じ環境であれば動作するようになっている。

TTL· SH ファイル

運用の手間を少しでも減らすため、スクリプトを用意した。TTL ファイルは TeraTerm 用のマクロである。中にはログインの手順が記述されており、TTPMACRO で起動することでパスワード入力以外の処理を自動で行える。SH ファイルはシェルスクリプトのファイルである。中には scp を用いた仮想 OS へのファイルコピーが記述されている。このファイルを実行するとすべてのファイルコピーが完了する。

（※文責: 辻田陸）

STUDIO KAMADA の説明

実装した ECM プログラムを Xeon Phi 上で実際に動作させたときに用いた合成数として、「STUDIO KAMADA」[6] というサイト に掲載されている合成数を利用をした。「STUDIO KAMADA」は、ニアレプディジット関連の数の素因数分解表の作成を行っている。分解対象としては、レピュニット、ニアレプディジット、プラトウアンドデプレッション、クワージレプディジット、およびニアレプディッジト回分数である。分解対象のそれぞれの言葉の意味は以下の通りである。

レピュニット (Repunit)

すべての桁の値が 1 であるような自然数のことである。例えば、1、11、111、1111、... である。反復単位数の意味である repeated unit の省略が由来となっている。

レプディジット (Repdigit)

レピュニットを含んだ、すべての桁の値が同じであるような自然数（ぞろ目）のことである。repeated digit を短くしてレプディジットと呼ばれている。

ニアレプディジット (Near-repdigit)

ニアレプディジットは、レプディジットの数字を 1 つだけ別の数字に置き換えた自然数のことである。

クワージレプディジット (Quasi-repdigit)

クワージレプディジットは、レプディジットの数字を 2 つだけ別の数字に置き換えた自然数のことである。

プラトウアンドデプレッション (Plateau-and-depression)

プラトウアンドデプレッションは、両端に別の数字を持つクワージレプディジットである。

ニアレプディジット回文数 (Near-repdigit-palindrome)

ニアレプディジット回文数桁数が奇数かつ真ん中の桁の数字だけが他と異なるニアレプディジットである。

「STUDIO KAMADA」に素因数を掲載するための手順は以下の通りとなる。

1. 「分解が期待される合成数」か「素因数分解表」から、素因数分解を行いたい合成数を選択する。
2. 「Free to factor」ボタンをクリックし、その合成数の投稿か予約を行うためのページが表示される。
3. その合成数を素因数分解するための基本的な手順を読み、素因数を後で投稿したいときは予約を行う。
4. 素因数分解を行い、見つかった素因数を投稿する。

また、予約ページにある「ECM の試み」では、合成数から素因数を探すために、ECM プログラムで素因数分解を行った結果を報告でき、これによりその合成数の未知の素因数がだいたい何桁以上あるかが分かるようになる。私たちは、ここで報告されている結果を基に、どの合成数を素因数分解するのかを決めた。

（※文責: 小野嘉翔）

結果

昨年度の ECM プログラムにエドワーズ曲線の加算公式に射影変換を適用し、それを Xeon Phi 上で並列処理できるようなプログラムが完成した。また、昨年度と今年度のプログラムを比較できるような機能を、それぞれのプログラムに追加した。昨年度と今年度のプログラムの比較を行った結果、昨年度よりも 15% の高速化ができた。また、プログラム班以外でも簡単に利用できるような簡単なマニュアルの作成もした。

（※文責: 小野嘉翔）

3.2.3 統計班

活動目的

統計班は、射影班によって取り入れた理論と、プログラム班が実装した機能により ECM プログラムがどれほど高速化したのかの検証を目的とした。ECM プログラムの高速化は本プロジェクトの目的の ECMNET へのランクインに必要不可欠な要素である。高速化のために取り入れた手法が処理時間などにどのような効果をもたらすかを調べることは、その手法やプログラムの評価をするにあたって非常に重要な項目となる。今回は昨年度作成された ECM プログラムと今年度作成されたプログラムの比較評価を行った。昨年度作成された ECM プログラムと今年度作成されたプログラムの大きな違いは、楕円曲線の式と加算公式を変更したことである。加算公式は ECM の計算の根幹を担う存在であるため、処理速度の変化はきわめて重要視すべきものである。また、この変更点がプログラムにどのような効果をもたらすのかを知ることで、今後のプログラムの改良に役立てることができる。

（※文責: 山本健太）

活動内容

始めに、どのようにして高速化を検証するのかを話し合った。今回は ECM プログラムの処理性能の評価をしたいと考えた。合成数を入力したときの出力までの時間は素因数の大きさなどに左右されるため正確な処理速度の測定には適さない。素数を入力することによってプログラムのすべての処理を終了するまで時間を計測することができるため、素数を入力することにした。実際のプログラムには素数判定の機能が組み込まれているが今回はプログラム班に依頼してその機能を削除して実測に臨んだ。また、処理速度のデータの信頼性を得るためいくつかの統計方法を提案した。

（※文責: 山本健太）

- 度数分布

  度数分布のデータの求め方は次の通りである。

  データの数 n を求める。度数分布表を作るには n は少なくとも 50、できれば 100 以上が望ましい。そしてデータを大きい順に並べる。次に範囲 R を求める。データの中から最大値 max と最小値 min をみつけ、範囲を計算する。R は、$R = max - min$ で求まる。次に階級間隔 h を決める。範囲 R を 10 ～ 20 の適当な数で割る。この時の商を k とする。階級間隔を測定値の測定単位の整数倍とするために k を整数値になるように丸める。階級分けの出発点を決める。最初の階級は最小値を含み、階級間隔 h であり、次の階級との境界値は共通し、しかもデータが 2 階級にまたがらないように、決めなければならない。そのため、最小値から測定単位の 2 分の 1 を引いた値を階級の出発点とするとよい。（こうすると階級の境界値は測定単位の整数倍プラス測定単位の 2 分の 1 となり、境界値にデータが重なることはなくなる。）出発点の値に階級間隔 h を随時加え、最大値を含む階級までの境界値を決定する。これらによって度数分布表を作成する。各階級の中央の値をその階級の階級値とする。次にヒストグラムを作成する。横軸に階級の境界値を取る。縦軸に度数を取る。このとき、できる図形の縦 · 横の比が 2: 3 程度になるのがよい。

  データの数が数千を超えたり、数十万になった時にそのままデータの点図などを示すとなると、不可能ではないがデータの全体の様子を見るには大きくなりすぎて役に立たない。そこで、データのサイズを圧縮した図表が度数分布の図表である。度数分布は、とても大きな数のデータを扱うには度数分布は適しているが、そこまでの大きいサイズの統計を取るわけではないので、今回は採用しなかった。

（※文責: 土田祐介）

- 相関関係

  2 変数間の関係を数値で記述する分析方法を相関分析 [7] と呼ぶ。2 変数をそれぞれ x、y としたとき、x と y の組み合わせ $(x_i, y_i), i = 1, 2, \cdots, n$ が得られる。これを直交座標系の点 (x_i, y_i) と考えたグラフのことを相関図と呼ぶ。x に対して y が一定の傾向をもって帯状領域の幅内で定まっている時 x と y には相関関係があるという。帯状領域が右上がり、つまり x と y の大小関係が同じ傾向にある場合は x と y には正の相関があるという。逆に、帯状領域が左上がり、つまり x と y の大小関係が逆である傾向にある場合は x と y には負の相関があるという。また、2 変数間の相関にどの程度の直線的な関係があるのかを表す測度として相関係数がある。相関係数を r とし、r は x と y の共分散 S_{xy} を x と y のそれぞれの分

散 S_x および S_y の積で割った比で表す。式で表すと $r = \frac{S_{xy}}{S_x S_y}$ となる。r は $-1 \leq r \leq 1$ の範囲にあり、一般的に一般的に $|r|$ が 1.0 に近いほど高い相関があり、$|r|$ が 0.0 に近くなるほど相関がなくなる。

（※文責: 山本健太）

- 実際に使用した統計方法

  上の 2 つの統計方法を調べたが、今回の計測にはそぐわないと判断したため、今回調べた統計方法は使用しなかった。そこで先生に相談して教えてもらった統計方法で計測をした。私たちは各プログラムの性能を計る為にそれぞれ 5〜 40 桁の素数を 5 桁毎に計測した。各桁の素数はその桁の素数をランダムに抽出するプログラムを作成した。a 桁の素数を抽出する時、その素数の範囲は $[10^{a-1}\ ,\ 10^{a-1} + 10^{\lceil \frac{a-1}{2} \rceil}]$ とした。これによって抽出された素数を用いて計測に使用した。各桁に対し 16 個の素数を用意し、合計 128 回の計測を行った。私たちは今回データを計測するにあたり、4 人で 4 台のパソコンを使用した。パソコンの性能は以下の通りである。

  - ホスト OS ： Windows8.1
  - ゲスト OS ： CentOS6.3
  - プロセッサ ： Intel(R) Core(TM)i3-3120M CPU @2.50Ghz 2.50Ghz
  - 実装メモリ (RAM) ： 8.00GB
  - システムの種類 ： 64 ビット オペレーティングシステム x64、 ベースプロセッサ

（※文責: 土田祐介）

結果

計測を行った結果、今年度と昨年度の ECM プログラムの処理速度は図 3.3 のようなグラフの結果になった。そしてプログラムの処理速度と向上率の数値を示した表が表 3.1 である。向上率が最大になったのは桁数が 10 桁の時で、約 18% も処理速度が速くなった。各桁数の向上率の平均を出したところ、約 15% 処理速度が向上した。

表 3.1　今年度と昨年度の ECM プログラムの処理速度を比較した向上率

桁数	新プログラム	旧プログラム	向上率(%)
10	43.562	53.297	18.265
15	54.812	64.412	14.905
20	56.079	68.229	17.808
25	69.612	78.78	11.637
30	72.822	82.503	11.734
35	74.455	84.608	12
40	96.555	114.179	15.436

（※文責: 土田祐介）

考察

上述の通り、今年度作成した ECM プログラムは昨年度使用した ECM プログラムよりも約 15% 処理速度が向上した。射影班が導入した射影変換により今年度の ECM プログラムは昨年度使用した ECM プログラムよりも計算コストを約 15% 削減したことが分かっている。つまりこの結果は、

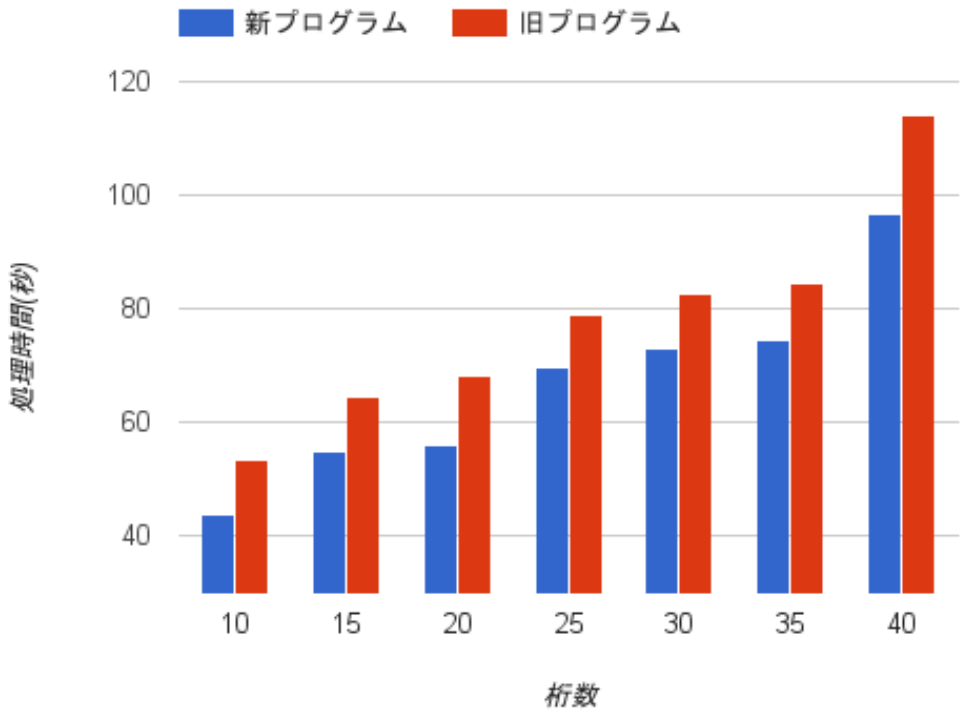

図 3.3　今年度と昨年度の ECM プログラムの処理速度の比較グラフ

射影変換によって削減された計算コストが理論通りに処理速度を向上させたことを示している。

（※文責: 山本健太）

3.2.4　最終発表

準備

- ポスター

  まず、前期に使ったポスター、昨年度のポスターを参考にしてポスターの構成を決めた。ポスターの制作には前期と同様に「Adobe Illustrator」というドローソフトを用いた。今回はメインポスターの他に 2 枚のポスターを作成した。後期から分かれた 3 つの班でそれぞれ、活動目的、活動内容、活動成果を記述した。難しい内容のところは図や表などを使って観る人が理解しやすいものとなるよう努力した。完成後、誤字や脱字がないか、見にくいところがないかを探しよりよいポスターを作成した。

- 原稿

  原稿の作成は統計班、射影班、プログラム班の 3 つに分かれて活動を行った。それぞれの班が、自分達で学習、担当した部分の原稿を書いた。中間発表の反省を踏まえて、専門的な内容になりすぎないように活動の成果を発表するということを意識した。最初に完成した原稿は詳細な説明を避け簡潔な内容にしていたため、内容が薄く発表時間が短いものになってしまった。そこでメンバで話し合いをしたり、先生に修正案をもらうことで、詳細な説明をする部分を決めた。また説明不足な部分や補足説明が必要なところを追加して発表の質を高め

ていった。

- スライド
  スライドの作成には中間発表の時は「Prezi」というプレゼンテーションソフトを使ったが、スライドの無駄な動きが多いなどの指摘があった為、今回は使わず、google drive を使い、全員でスライドを共有しながら作業を進めた。完成したスライドを先生とメンバで見直したところ、説明不足なところや文章量が多いところなどがあったので文章を減らしたり、ページを分けるなどして対策をした。これにより見やすいスライドを作ることができた。

（※文責: 土田祐介）

最終発表当日

最終発表では、前半と後半で 3 人ずつに分かれて発表を行った。3 人のうち 1 人が原稿を読み上げている間、もう 1 人はスライドのページをめくり、さらにもう 1 人は発表評価シートを配布した。発表内容はプロジェクトの概要、理論班の活動内容、プログラム班の活動内容をそれぞれ 3 人で分担した。後期は理論班とプログラム班に分かれて活動し、活動内容が多くなったため、ポスターを 3 枚用意した。また、記入しやすいよう発表評価シートをバインダーに綴じて配布した。

（※文責: 石川夏樹）

評価

発表の評価は以下の五項目について五段階中で評価した後、総合的に評価した。

- 目的 : 明確である、分かりやすいといった意見が多くを占めた。(5 点)
- 現状把握 : グラフや図などを用いることで、難解である現状の内容をできる限り分かりやすく伝えたが、目的と現在行っているプログラムの高速化の関連や並列化は新規なのか既存なのかがいまいち見えていない人もいた。(3 点)
- 今後の計画性 : 今後は更なるプログラムの高速化を図ると同時に ECMNET へのランクインを目標といった内容は、多くの人にはしっかり伝わっていたように感じた。(4 点)
- 表現力 : 中間発表と同様に専門的で難解な内容が多かったが、評価の内容を見る限りでは例やグラフ、図などを用いたことで伝えることが出来たと考えられる。また最終発表では、中間発表では賛否両論だった Prezi の使用をやめ、Google のプレゼンテーション機能を使用したことで、無駄な動きなどは減った。しかし、そもそもの配色であったり、さらに図やグラフなどがあればいいなどの意見もあり、表現方法においては更なる改善の余地があった。(3 点)
- チームワーク : 少ない人数ではあるがアクシデントや作業分担などしっかりと協力して行うことが出来た。(5 点)
  以上五項目の評価より、私たちの発表の操業評価は五段階中 4 点とした。

（※文責: 九島拓実）

第 4 章　プロジェクト内のインターワーキング

- 山本健太 (プロジェクトリーダー、統計班)
  (1) 楕円曲線法の基礎理論の理解。
  (2) 必要な提出物の確認。その締切についての情報共有を行い、担当者の指名を行った。
  (3) スケジュールを作成し、いつまでに、誰が、どこを担当しているのかを分かりやすくした。
  (4) 中間発表会のポスターの活動計画の下書きの作成を行った。
  (5) 中間発表会の原稿のプロジェクト目標、背景の説明部分の作成を行った。
  (6) 中間発表会のスライドのプロジェクト目標、背景の説明部分の作成を行った。
  (7) 昨年度と今年度の ECM プログラムの比較をために、統計班の土田と統計の学習を行った。
  (8) 統計の学習のために、情報ライブラリで統計班の土田と協力して書籍を探した。
  (9) 昨年度と今年度の ECM プログラムの比較のために、射影班と協力して、桁数ごとの処理速度のデータを取った。
  (10) 加算公式の計算コストが実際にどのくらいかを知る必要があったので、統計班の土田と協力して GMP を用いた四則演算の処理速度のデータを取った。
  (11) 最終成果発表に向けて、昨年度と今年度のプログラムのどちらが処理速度が速いかを、第三者から見て分かりやすくなるように、統計班の土田とスライドに載せるグラフを工夫した。
  (12) 最終成果発表では、第三者が聞いて理解できるような発表を行う必要があるので、統計をどのようにして取ったのかを詳しく説明するスライドと文章の作成を統計班の土田と協力して行った。
  (13) 統計結果を第三者にとって分かりやすいものになるように、最終成果発表のポスター作製を統計班の土田と協力して行った。
- 辻田陸 (担当: プログラム班)
  (1) 楕円曲線法の基礎理論の理解。
  (2) 中間発表会に向けて、ポスターの活動内容の下書きの作成を行った。
  (3) 中間発表会に向けて、原稿のプログラム説明、今後の予定部分の作成を行った。
  (4) 中間発表会に向けて、スライドのプログラムの説明、今後の予定部分の作成を行った。
  (5) プログラム作成のためには、GMP ライブラリを用いた数値計算を行う必要があった。そのため、プログラム班の小野と協力して GMP ライブラリの学習を行った。
  (6) プログラムを並列処理をさせて、効率化を図ろうとした。そこで、Xeon Phi 上で並列処理するプログラムの作成方法と手順について、プログラム班の小野と協力して学習を行った。
  (7) 昨年度と今年度のプログラム比較には、処理時間の表示とテキストファイルから数値を読みだす機能が必要であった。そこで、統計班と射影班と協力してデータが取りやすいようにプログラムの変更を行った。
  (8) プログラムを変更した際のコンパイルを効率良く行うために、プログラム班の小野と協

力して Makefile の作成を行った。

(9) 並列処理を行う Xeon Phi の使い方について、プログラム班以外のメンバでも使えるように簡単なマニュアルをプログラム班の小野と作成を行った。

(10) 最終成果発表に使うスライドと原稿の作成において、ECM プログラムに重要な項目である並列処理と GMP についての説明を、第三者が聞いて理解できるように簡潔にかつ分かりやすいものを目指して、プログラム班の小野と協力して行った。

- 石川夏樹 (担当: 射影班)
  (1) 楕円曲線法の基礎理論の理解。
  (2) 中間発表会に向けて、ポスターの活動内容の作成を行った。
  (3) 中間発表会に向けて、原稿の学習内容の剰余系に関する説明部分の作成を行った。
  (4) 中間発表会に向けて、スライドの剰余系の説明部分の作成を行った。
  (5) プログラムの高速化には射影変換の導入が重要であった。よって、射影座標系を楕円曲線法の加算公式に適用させるために、射影班の九島と協力して射影座標系の基礎的な学習を行った。
  (6) 射影変換の学習のために、射影班の九島と協力して、情報ライブラリで書籍をさがした。
  (7) 昨年度と今年度の ECM プログラムの比較のために、統計班と協力して、桁数ごとの処理速度のデータを取った。
  (8) 射影座標系を適用した加算公式をプログラム班がすぐに実装できるように、射影班の九島と協力して、加算公式のアルゴリズムを作成した。
  (9) 最終成果発表で、射影座標系を適用した加算公式のメリットを第三者が聞いても理解できるように、射影班の九島と協力して、スライドと原稿の作成を行った。
  (10) 射影変換がどのようなものかを分かりやすく説明できるように、射影班の九島と協力して、最終成果発表のポスター作製を行った。
- 九島拓実 (担当:　射影班)
  (1) 楕円曲線法の基礎理論の理解。
  (2) 中間発表会に向けて、ポスターの概要の日本語文の作成を行った。
  (3) 中間発表会に向けて、原稿の学習内容のラグランジュの定理、点の位数に関する説明部分の作成を行った。
  (4) 中間発表会に向けて、スライドのラグランジュの定理、点の位数に関する説明部分の作成を行った。
  (5) 射影変換の学習のために、射影班の石川と協力して、情報ライブラリで書籍をさがした。
  (6) 昨年度と今年度の ECM プログラムの比較のために、統計班と協力して、桁数ごとの処理速度のデータを取った。
  (7) 射影座標系を適用した加算公式をプログラム班がすぐに実装できるように、射影班の石川と協力して、加算公式のアルゴリズムを作成した。
  (8) 最終成果発表で、射影座標系を適用した加算公式のメリットを第三者が聞いても理解できるように、射影班の石川と協力して、スライドと原稿の作成を行った。
  (9) 射影変換がどのようなものかを分かりやすく説明できるように、射影班の石川と協力して、最終成果発表のポスター作製を行った。
- 土田祐介 (担当: 統計班)
  (1) 楕円曲線法の基礎理論の理解。
  (2) 中間発表会に向けて、ポスターの概要の英語文の作成を行った。

(3) 中間発表会に向けて、原稿の学習内容の無限遠点と因数の関係に関する説明部分の作成を行った。
(4) 中間発表会に向けて、スライドの無限遠点と因数の関係に関する説明部分の作成を行った。
(5) 昨年度と今年度の ECM プログラムの比較をために、統計班の山本と統計の学習を行った。
(6) 統計の学習のために、情報ライブラリで統計班の山本と協力して書籍を探した。
(7) 昨年度と今年度の ECM プログラムの比較のために、射影班と協力して、桁数ごとの処理速度のデータを取った。
(8) 加算公式の計算コストが実際にどのくらいかを知る必要があったので、統計班の山本と協力して GMP を用いた四則演算の処理速度のデータを取った。
(9) 最終成果発表に向けて、昨年度と今年度のプログラムのどちらが処理速度が速いかを、第三者から見て分かりやすくなるように、統計班の山本とスライドに載せるグラフを工夫した。
(10) 最終成果発表では、第三者が聞いて理解できるような発表を行う必要があるので、統計をどのようにして取ったのかを詳しく説明するスライドと文章の作成を統計班の山本と協力して行った。
(11) 統計結果を第三者にとって分かりやすいものになるように、最終成果発表のポスター作製を統計班の山本と協力して行った。

- 小野嘉翔 (担当: プログラム班)
  (1) 楕円曲線法の基礎理論の理解。
  (2) 中間発表会に向けて、ポスターの活動計画の作成を行った。
  (3) 中間発表会に向けて、原稿の学習内容の下書きを行った。
  (4) 中間発表会スライドの学習内容の記述の最終的な確認を行った。
  (5) プログラム作成のためには、GMP ライブラリを用いた数値計算を行う必要があった。そのため、プログラム班の辻田と協力して GMP ライブラリの学習を行った。
  (6) プログラムを並列処理により、効率化を図ろうとした。そこで、Xeon Phi 上で並列処理するプログラムの作成方法と手順について、プログラム班の辻田と協力して学習を行った。
  (7) プログラムを変更した際のコンパイルを効率良く行うために、プログラム班の辻田と協力して Makefile の作成を行った。
  (8) 並列処理を行う Xeon Phi の使い方について、プログラム班以外のメンバでも使えるように簡単なマニュアルをプログラム班の辻田と作成を行った。
  (9) 最終成果発表に使うスライドと原稿の作成において、ECM プログラムに重要な項目である並列処理と GMP についての説明を、第三者が聞いて理解できるように簡潔にかつ分かりやすいものを目指して、プログラム班の辻田と協力して行った。

（※文責: 小野嘉翔）

第 5 章　結果

5.1　前期

5.1.1　成果物

ECM のプログラムを作成するために必要な楕円曲線の基礎を学んだ。 ECMNET への登録に向けて簡単な ECM のプログラムを PARI/GP を使用して作成した。PARI/GP とは、数値計算ソフトである。これらを中間発表で発表する為に、スライド、原稿、ポスターを作成した。スライドは、Prezi というプレゼンテーションソフトを使って作成した。そして、ただ原稿の文章を載せるだけでなく、視覚的に見やすい様に文字を減らし、例などを用いることによって見ている人がわかりやすいと感じるように工夫をした。原稿は、学習している内容が専門用語が多く理解するのが難しいと判断した為、聞いている人がわかりやすいように専門用語の説明を入れたり、例で実際に示すなどをしてわかりやすい様に工夫をした。ポスターは、図を用いたり、配色に気を配り、見やすいようにした。

（※文責: 土田祐介）

5.1.2　プログラム

ここではプロトタイプとして開発した楕円曲線法を用いた素因数分解の PARI/GP のプログラム「ECM.gp」の解説をする。初めに、素因数分解したい数を p として、その値を決める。$(x, y) \in (\mathbb{Z}/p\mathbb{Z})^2$ を 1 組とり、変数 `x1`、`y1` へそれぞれ代入する。楕円曲線の式における x の係数 a の値を決める。

```
p=15;
x1=Mod(12,p);
y1=Mod(4,p);
a=Mod(1,p);
```

次に、適当な自然数をとり、変数 k を用いて 2 からその数までの for 文ループを行う。ここではその数を 10000 とした。このループ内ではスカラー倍の計算、つまり、kP の計算を行う。この計算のために scalar 関数を呼び出す。スカラー倍された点は `x1` 、`y1` に格納され、次のループで使われる。これによって、scalar 関数で求められるのは $(k!)P$ となる。

```
for(k=2,10000,
  Q=scalar(x1,y1,k,a,p);
  x1=Q[1];
  y1=Q[2];
);
```

scalar 関数は x1、y1、k、a、p を引数とし、スカラー倍後の点の座標を返す。初めに k の値を binary 関数で 2 進数に変換し変数 N に代入する。また、そのビット長を length 関数で求め変数 len に代入する。for 文内では点の加算を行う。変換されたビットを調べていき、ビットが立っていれば元の点の加算を行う。また、加算は kasan 関数で行う。

```
scalar(x1,y1,n,a,p)={
  local(x2,y2,N,len,i,Q);
  N=binary(n);
  len=length(N);
  x2=x1;
  y2=y1;

  for(i=2,len,
    Q=kasan(x2,y2,x2,y2,a,p);
    x2=Q[1];
    y2=Q[2];

    if(N[i]==1,
      Q=kasan(x1,y1,x2,y2,a,p);
      x2=Q[1];
      y2=Q[2];
      );
    );
  return([x2,y2]);
}
```

kasan 関数は計算する 2 点の座標と a、p を引数とし、加算して出た点の座標を返す。初めの if 文では与えられた 2 点の傾きを求める計算をする。次に gcd 関数を用いて素因数を発見したか判定している。gcd 関数は与えられた 2 つの数の最大公約数を求める関数である。ここで素因数が見つかればそれを表示し、プログラムを終了する。そうでなければ続行する。最後に計算後の点を求め、それを返す。

```
kasan(X1,Y1,X2,Y2,a,P)={
  local(bunshi,bunbo,L,X3,Y3);
  if(X1==X2 && Y1==Y2,
    bunshi=3*(X1)^2+a;
    bunbo=2*Y1,
    bunshi=Y2-Y1;
    bunbo=X2-X1);

    if(gcd(lift(bunbo),P)!=1, print(gcd(lift(bunbo),P)); return(gcd(lift(bunbo),P)));
    L=bunshi/bunbo;

    X3=L^2-X1-X2;
```

```
    Y3=L*(X3-X1)+Y1;
    Y3=-Y3;
    return([X3,Y3]);
}
```

（※文責: 辻田陸）

5.2 後期

5.2.1 成果物

最終発表に向けて、スライド、原稿、ポスターの作成を行った。スライドは中間発表では「Prezi」を使用したが、無駄な動きが多いなどの評価があったため、今回は google drive のプレゼンテーションを使い、全員で共有を行いながら作業を進めた。スライド 1 枚 1 枚に文字数が多くなりすぎないように、要点だけをまとめ、図を用いるなどして見やすくした。また、色を使うことによって、聴講者の視線を集めるなどの工夫をした。原稿は、前期の反省を活かし、専門的な内容で話が難しくならないよう、成果発表ということを意識して作成を進めた。ポスターは、プロジェクトの成果をしっかりと載せる為に、メインポスターの他にサブポスターを 2 枚作成した。サブポスターは班毎に活動目的、活動内容、活動成果を記載した。これにより内容が充実したポスターを完成させることができた。

（※文責: 土田祐介）

5.2.2 射影班

射影班は射影変換に関する学習を行い、各計算の計算コストを C 言語環境下で各計算を 1 億回行う ECM プログラムを作成し、そのプログラムをそれぞれ動かし調べて比較した。また射影変換前後の計算コストの算出を行った。今年度のプログラムに射影変換を用いた結果、昨年度の完成したプログラムの加算公式一回の計算コストが 13.6M だったのに対し、今年度のプログラムの加算公式一回の計算コストが 11.8M だったことから、理論上プログラムを約 15% の高速化をすることに成功したと考えられた。

（※文責: 九島拓実）

5.2.3 プログラム班

プログラム班は初めにコーディングを担当する箇所を振り分けることでソースコードの共有における食い違いを防いだ。また、コーディングの際には今後の改良やメンテナンスがしやすいよう書くことを心掛けた。こうして新しいアルゴリズムを適用した素因数分解プログラムを完成させた。主な改良点は新アルゴリズムによる高速化、ループ実装による効率化である。また、学内ネットワークを通じてプログラムを実行できるようにしたことで効率的な素因数探しを行えるようになった。

（※文責: 辻田陸）

5.2.4 統計班

昨年度の ECM プログラムと今年度の ECM プログラムを比較する為に、5 ～ 40 桁の素数を 5 桁毎に 16 個ずつ入力し、プログラム内部で 1 万回計算を行った時の処理速度の結果を図と表に出力した。得られた結果から、昨年度のプログラムよりも、今年度のプログラムの方が約 15% 処理速度が向上した。射影班が算出した、昨年度のプログラムと今年度のプログラムの計算コストを比較した時に、計算コストが今年度のプログラムの方が 15% 削減されていたことがわかる。よって、射影変換によって、理論通りに計算処理速度が向上したことが言える。

（※文責: 土田祐介）

5.3 解決手順と評価

5.3.1 前期

- 楕円曲線について学習し、ECM のプログラムを作成する為の基本的な知識を得ることができた。
- 簡単な ECM のプログラムを作成したことにより、後期からの ECMNET に挑戦するための難しいプログラムを作成する為の知識を得ることができた。
- PARI/GP で作成したプログラムのため、計算の処理速度が遅いため、大きい桁数の素因数を求めるのに向いていない。

（※文責: 土田祐介）

5.3.2 後期

射影班

射影変換及び射影座標系についての知識が不足していたため、ライブラリで書籍を探し、学習を進めようとした。しかし、楕円曲線法に直接適用できる手法が見つからなかったため、白勢先生に教えていただいた書籍のみに絞り込み、楕円曲線法に関連する部分を輪読した。さらに、射影変換が ECM プログラムの中の加算公式内で行われることに加え、実際に使用する楕円曲線の式をもとに射影変換の手順を教えていただいた。射影変換を基礎から学び、理論的な部分を細部まで理解することが目的ならば、書籍を輪読することは妥当な手段であったが、楕円曲線法に適用することが目的ならば、白勢先生に楕円曲線法との関連性を交えながら教えていただくことが最も最適な手段であったといえる。

（※文責: 石川夏樹）

プログラム班

楕円曲線法のプログラム実装のために、GMP ライブラリを用いたプログラミング方法と並列処理プログラミングについて学習を行った。まず始めに、並列処理化していない ECM プログラムを作成することを目標とした。使用する言語には、C 言語を用いた。C 言語を用いた理由としては、C 言語は他の言語に比べて計算速度が比較的速いので、ECM プログラムに適してるためである。

また、数値計算には、GMP ライブラリを用いることとした。GMP ライブラリは、任意精度計算ライブラリである。ECM プログラムでは、非常に大きな桁数の数値計算を扱う。よって、C 言語の基本データ型では数値精度に限界があるために、数値計算にほぼ制限がない GMP ライブラリを導入することとした。GMP ライブラリを用いた数値計算は、C 言語の数値計算とは少し異なるため、GMP を用いた数値計算になれる必要があった。よって、GMP のマニュアルや参考サイトを読み、実際に C 言語で GMP を用いた数値計算のプログラムを書くことを行った。これにより、GMP ライブラリを用いた計算ができるようになった。GMP についての学習を行った後に、昨年度のプログラムを読むことで、ECM プログラムの理解をスムーズに行うことができた。また、昨年度のプログラムを手分けして読み、理解したことを共有することで、効率化を図った。昨年度のプログラムの理解を終えた後に、射影班が作成した、エドワーズ曲線に射影変換を導入した楕円曲線法のアルゴリズムを昨年度のプログラムに適用した。この段階では、並列処理化を行わずに実装することで、プログラムにバグがないかの確認をし易くした。一通り改良を終えた後は、小さな桁数の素因数分解を行い、結果を確認することで、バグがないかの確認を行った。他にも、加算公式の計算で出力された点が、実際の楕円曲線上にあるかの確認も行った。確認が終わった後に、改良をしたプログラムを並列処理するために、昨年度の ECM プログラムで並列処理化されている部分を読んだあと、その部分がどのように処理されているのか、どのような記述すればいいのかを白勢准教授に教えて頂いた。その後、実際に改良したプログラムを OpenMP を用いて並列化した。また、並列処理は、Xeon Phi 上で行った。これらにより、エドワーズ曲線を用いた楕円曲線法のプログラムの実装ができた。

（※文責: 小野嘉翔）

統計班

統計を取るにあたり、統計方法がわからなかったのでライブラリで統計に関する書籍を探し、統計の取り方を調べた。度数分布や相関関数などの統計の取り方を見つけたが、今回の実験には適さないと判断しこれらの統計方法は使用しなかった。そして、由良先生にアドバイスをいただき、そのアドバイスに基づいて、統計を行った。統計方法の調査は、実験自体に時間をかけたかったため、あまり時間をかけられる内容ではなかったので、詳細に調べることができなかったが、アドバイスに基づいた統計でしっかりとデータを取ることが出来たので問題はなかった。

（※文責: 土田祐介）

プロジェクト全体

前期に引き続き ECMNET へのランクインを目指した。ECMNET へのランクインの為に、ECM プログラムの高速化を目指した。そして、プログラムの高速化を行う為に 3 つの班にわかれた。統計班、射影班、プログラム班にわかれてそれぞれが独自に活動を行った。現時点では ECMNET に登録できるような素因数は見つかっていない。3 つの班にわかれたことによって、効率的に作業を進めることができた。加算公式の計算コストを約 15% 削減することができ、プログラムの処理速度が約 15% 向上できたので、プログラムの高速化をすることに成功した。

（※文責: 土田祐介）

第 6 章　まとめ

6.1　プロジェクトの成果

6.1.1　前期

前期では ECM プログラム開発のために楕円曲線法の学習を行った。その結果、プロジェクトメンバ全員が開発のために十分な楕円曲線法の基本的な知識を得ることができた。また、PARI/GP を用いた小規模な ECM プログラムを作成した。これにより、後期で行うプログラム作成のイメージがしやすくなった。

（※文責: 辻田陸）

6.1.2　後期への課題

前期で行った学習は楕円曲線法の基本的な部分のみであり、アルゴリズム改良のためにはさらなる学習と調査が必要となる。また、後期では C 言語を用いたプログラム開発を行うため、その学習と並列化実装の学習も行っていくこととなる。これらの学習を行い、プログラムの改良と ECMNET へのランクインができるように計画を立てて進行していくことも課題とする。

（※文責: 辻田陸）

6.1.3　後期

後期では射影班、プログラム班、統計班に分かれ活動した。射影班は楕円曲線法にエドワーズ曲線を導入し射影変換を適用した。その結果計算コストを理論上約 15% 削減することに成功した。プログラム班は射影班が行ったエドワーズ曲線の導入と射影変換を組み込み、Xeon Phi 上で並列処理できるプログラムを作成した。統計班は今年度作成した ECM プログラムと昨年度作成された ECM プログラムを比較し、昨年度よりも約 15% 処理速度を向上したことを証明した。

（※文責: 山本健太）

6.1.4　今年度の成果

本プロジェクトの今年度の目的は楕円曲線を理解し、ECMNET へランクインすることであった。目的達成のために楕円曲線法の理解、 ECM プログラムの高速化と並列化の実装、ECM プログラムの性能評価の 3 つを主軸において活動を進め、以下のことを達成した。
1 つ目は前期に基礎学習を行い楕円曲線への理解を深めた。2 つ目は従来のヴァイエルシュトラス方程式からエドワーズ曲線に変更し射影変換を導入することで計算コストを約 15% 削減し、Xeon Phi 上で並列化処理を施した ECM プログラムを完成させた。3 つ目は昨年度と今年度の ECM プログラムを比較し、約 15% 処理速度を向上したことを証明した。

（※文責: 山本健太）

6.2 プロジェクトにおける各人の役割

6.2.1 前期

基礎学習については、役割分担をせずに取り組んだ。発表準備の段階では、ポスター制作を概要班（土田・九島)、活動内容班（辻田・石川)、活動計画班（山本・小野）にわけて制作した。また、原稿とスライド制作では概要・その他班（山本・辻田)、学習内容班（土田・九島・石川・小野）にわかれて制作した。中間発表では前半組（山本・土田・九島）と後半組（辻田・石川・小野）にわかれた。

（※文責: 石川夏樹）

6.2.2 後期

後期ではプログラム開発を円滑に行うため、楕円曲線のアルゴリズムの調査・作成や性能比較を担当する理論班 (石川・九島・土田・山本) とコーディング・実装を担当するプログラム班 (小野・辻田) に分かれて活動した。また、理論班の中でさらに射影変換を加算公式に適用させることを目的とした射影班 (石川・九島) とプログラムの性能の統計を取り比較する統計班 (土田・山本) に分かれた。

最終発表の準備においては、射影班、統計班、プログラム班のそれぞれのメンバがスライド・ポスター・原稿で説明する部分を制作した。最終発表では前半組 (石川・小野・辻田) と後半組 (九島・土田・山本) に分かれ発表を行った。

（※文責: 九島拓実）

6.3 今後の課題と展望

基本的な理論の学習、プログラムの学習、発表の準備などが主な活動だったため、他の活動に手が回らなかった。来年度のプロジェクト学習に向け、今後の展望を以下に示す。

- 素因数分解を 2 つのステージに分けて行う方法があり、その場合、今年度行っている ECM はステージ 1 で使用される。ステージ 2 では、誕生日攻撃法を利用したロー法による素因数分解を行う。まず、ECM で素因数分解に失敗した場合、座標が算出される。その座標をステージ 2 で使用することで、素因数分解が成功する場合がある。それにより、素因数分解が成功する確率を高めることができる。よって、ステージ 2 のプログラムを実装し、ECM と連携させる。
- 論文を読む時間があまりとれなかったため、特に楕円曲線法に関する論文をプロジェクトメンバ全員で輪読し、理解を深める。
- プログラムを実際に動かす時間があまりとれなかったため、プログラムの動作回数を稼ぎ、目標とする桁数の素因数分解など、ECMNET への登録に向けた活動をする。
- 成果発表会で何のために素因数分解をするのかわからないという意見が多かったため、楕円

曲線法や素因数分解がどのように現代社会に関係しているか学ぶ。また、プロジェクト学習を行う背景や概要などに関係することについて簡潔にまとめ、発表する。特に、公開鍵暗号や RSA 暗号は素因数分解と密接に関係しているため、成果発表会でプロジェクト学習との関連性を示す。

（※文責: 石川夏樹）

付録 A　新規習得技術

本プロジェクトで修得した新しい技術は以下である。

- PARI/GP の使用
- Adobe Illustrator の使用
- Prezi の使用
- Xeon Phi の使用
- GMP の使用
- Open MP の使用
- gnuplot の使用
- Google スライドの使用

（※文責: 山本健太）

付録 B　相互評価

ここでは各メンバに評価されたコメントを列挙していく。

- 石川夏樹
  - 他のプロジェクトメンバが理解に困っていたときに進んで説明を行ってくれていた。
  - 難しいところをみんなにわかりやすく説明していた。講習会に積極的に参加していた 。
  - 講習会などに参加するなどプロジェクトに貢献し積極的に取り組んでいた 。
  - 講習会に率先して参加していた 。
  - TeX やポスターなどの講習会に積極的に参加してくれていた。
  - 加算公式の射影変換を頑張っていた。
  - 同じ射影班として互いに足りない部分をカバーしてくれた
  - 射影班の中心として、射影の理論を学びそれをわかりやすくまとめていた
  - 新式に射影変換を取り入れるための活動を精力的に行っていた
  - 数学的な部分で頑張ってくれた
- 小野嘉翔
  - 先生に分からない所を積極的に質問していた。
  - 皆で話し合ってわからないところがあった時に先生に聞いたり、それをわかりやすくみんなに説明してくれた。
  - PC 関係で他のメンバのサポートを行うなど献身的に取り組んでいた。
  - 疑問解決に積極的だった。
  - 他の人の不明な点に対して。わかりやすく説明してくれた。
  - 理論班が学習した内容をプログラムに実装してくれた
  - 理論班が出した理論をプログラムにしっかりと作り上げてくれた
  - プログラム班として、ECM プログラムの改良に専念していた
  - ECM プログラムの改良や他のメンバへのプログラムの環境の整備などを積極的に行っていた
  - 同じ班で自分の足りないところを補ってくれた
- 九島拓実
  - 疑問や問題点を積極的にメンバに伝えていた
  - 自分が疑問に思っているところを他のメンバに尋ねることで、相互の理解に努めてくれていた。
  - みんなで議論をする時に積極的に発言をしていた。
  - ポスターやスライドの作成に試行錯誤しより良いものを追求し精力的に取り組んでいた
  - 発表物制作においてよく疑問を投げかけたり提案を行っていた
  - 加算公式を読み計算コストを調べてくれた
  - 射影変換についての学習を頑張っていた
  - 射影班として理論を学びプログラム班に的確に理論を伝えていた
  - 射影変換の他班への説明やポスターや原稿など主体的に他者への理解の助けを行っていた

  - 計算コスト調査を担当してくれた
- 土田祐介
  - 様々な視点から問題を分析していた。
  - 講習会などのプロジェクトに関連するイベントに積極的に参加してくれていた。
  - メンバと協力し新しい考えを提案し能動的に取り組んでいた。
  - 講習会や議論に積極的に参加していた。
  - 話し合いの場において積極的に発言していた。
  - 積極的に発言し理論班とプログラム班の活動内容の把握に貢献した
  - 統計の学習を頑張っていた
  - 期末発表のスライドの全体的な修正や原稿のまとめを行ってくれた
  - 統計方法の模索や全体意見交換に意欲的に取り組んでいた
  - 統計の部分で頑張ってくれた
- 山本健太
  - 今何が問題になっているかメンバに分かりやすく伝えていた。
  - リーダーとして、プロジェクトをまとめてくれていた。
  - リーダーとしてみんなをまとめてくれて、様々な作業を率先しておこなってくれた。
  - スケジュールの管理や提出物の確認、指示出しなどを仕切っていた。
  - 常にリーダーとして去年のメンバとのやり取りや今後の予定などをまとめてくれていた。
  - リーダーとして予定管理してくれた
  - リーダーとして、プロジェクトを引っ張ってくれていた
  - 統計班として統計を取る前段階のプログラムを作成しつつ、リーダーとして全班のパイプ役としてつないでいた
  - リーダーとしてみんなに指示を出し、率先して仕事をこなしメンバをまとめてくれた
  - リーダーとしての統率が上手だった
- 辻田陸
  - ポスターやプログラムを工夫して製作に取り組んでいた。
  - 中間発表に必要なプログラムを、見やすさを考慮して作成してくれた。
  - 発表に使う ECM のプログラムの作成の大半を担当した。
  - プログラムなどの成果物を率先的に制作し取り組んでいた 。
  - プログラムの面で常に中心的人物だった。
  - プログラムの扱い方を理論班に伝えてくれた
  - 楕円曲線法のプログラムの作成を頑張っていた
  - 統計用のプログラムの作成を何度も手直ししてより良いものにしていた
  - プログラムなどの成果物を率先的に制作し取り組んでいた
  - 他班のメンバにプログラム関係の説明や支援を活動的に行っていた

（※文責: 九島拓実）

付録 C　その他製作物

§ 5.1.2 を参照

- ECM.gp

```
kasan(X1,Y1,X2,Y2,a,P)={
    local(bunshi,bunbo,L,X3,Y3);
    if(X1==X2 && Y1==Y2,
        bunshi=3*(X1)^2+a;
        bunbo=2*Y1,
        bunshi=Y2-Y1;
        bunbo=X2-X1);

    if(gcd(lift(bunbo),P)!=1, print(gcd(lift(bunbo),P)); return(gcd(lift(bunbo),P)));
    L=bunshi/bunbo;

    X3=L^2-X1-X2;
    Y3=L*(X3-X1)+Y1;
    Y3=-Y3;
    return([X3,Y3]);
}

scalar(x1,y1,n,a,p)={
    local(x2,y2,N,len,i,Q);
    N=binary(n);
    len=length(N);
    x2=x1;
    y2=y1;

    for(i=2,len,
        Q=kasan(x2,y2,x2,y2,a,p);
        x2=Q[1];
        y2=Q[2];

        if(N[i]==1,
            Q=kasan(x1,y1,x2,y2,a,p);
            x2=Q[1];
            y2=Q[2];
        );
```

```
    );
    return([x2,y2]);
}

{
    p=15;
    x1=Mod(12,p);
    y1=Mod(4,p);
    a=Mod(1,p);
    for(k=2,10000,
        Q=scalar(x1,y1,k,a,p);
        x1=Q[1];
        y1=Q[2];
    );
}
```

（※文責: 辻田陸）

参考文献

[1] ECMNET
http://www.loria.fr/~zimmerma/records/ecmnet.html (最終アクセス 2016 年 1 月 5 日)
[2] 公立はこだて未来大学 2014 年度 システム情報科学実習グループ報告書
http://www.fun.ac.jp/~sisp/old_report/2014/08/document08_A.pdf
(最終アクセス 2016 年 1 月 8 日)
[3] PARI/GP Development Headquarters
http://pari.math.u-bordeaux.fr/ (最終アクセス 2015 年 12 月 11 日)
[4] プレゼンソフト — オンラインプレゼンツール — Prezi
https://prezi.com/ (最終アクセス 2015 年 12 月 11 日)
[5] GNU MP
http://sehermitage.web.fc2.com/etc/gmp.html (最終アクセス 2016 年 1 月 15 日)
[6] STUDIO KAMADA
http://stdkmd.com/ (最終アクセス 2016 年 1 月 15 日)
[7] 質問紙調査における相関係数の解釈について
http://www.mizumot.com/method/2011-06_Mizumoto.pdf
(最終アクセス 2015 年 12 月 17 日)