UNIVERSIDADE ABERTA
DEPARTAMENTO DE CIÊNCIAS e TECNOLOGIAS
MESTRADO em ESTATÍSTICA, MATEMÁTICA E COMPUTAÇÃO

10^{o} Problema de Hilbert para SubAnéis de $\mathbb{Q}$

António Pedro da Silva Raimundo
Dissertação de Mestrado

Lisboa, dezembro 2014

MESTRADO em ESTATÍSTICA, MATEMÁTICA E COMPUTAÇÃO

10^{o} Problema de Hilbert para SubAnéis de $\mathbb{Q}$

António Pedro da Silva Raimundo

Orientador: professor doutor Mário Jorge Edmundo
Co-Orientador: doutor Marcelo Mamino

Dissertação apresentada ao programa de **Mestrado em Estatística, Matemática e Computação do Departamento de Matemática e Tecnologia da Universidade Aberta** como requisito parcial para obtenção do grau de Mestre em **Estatística, Matemática e Computação.**

Lisboa, dezembro 2014

Agradecimentos

À minha mulher Teresa, aos meus filhos David e Guilherme, a toda a família e amigos pelo apoio e compreensão ao longo da elaboração da dissertação.

Ao professor doutor Mário Jorge Edmundo, pela sugestão do tema, pelo apoio e pela permanente disponibilidade no acompanhamento do trabalho.

Ao doutor Marcelo Mamino, pelo apoio dado na fase inicial do trabalho.

Ao meu filho Guilherme Raimundo, nas sugestões e críticas ao trabalho desenvolvido.

À minha colega de mestrado Denise Candeias pela motivação e discussões sobre o tema.

Ao meu colega de empresa João Peixoto pelas sugestões na solução de questões na área técnica das ferramentas usadas.

Ao meu colega de empresa José Coelho pela colaboração na revisão do texto final.

Á Sage que ao disponibilizar um software matemático de alta qualidade, facilitou o desenvolvimento do estudo prático.

A todos, obrigado.

Resumo

Nesta dissertação estudamos o artigo de Bjorn Poonen [Poo03b], Hilbert‘s tenth problem and Mazur‘s conjecture for large subrings of $\mathbb{Q}$ e investigamos computacionalmente os diversos conjuntos caracterizados no artigo.
Começamos por introduzir a teoria referente às variedades algébricas e às curvas elípticas , conceitos necessários ao entendimento do artigo em estudo.
No artigo de Poonen é demonstrada a inexistência dum algoritmo para decidir se equações polinomiais com coeficientes em certos subanéis de $\mathbb{Q}$ têm ou não solução nesses subanéis ou seja o 10^{o} problema de Hilbert para esses anéis tem uma solução negativa. A ideia da prova é a partir duma curva elíptica construir um modelo diofantino do anel $\mathbb{Z}$. Com esse fim, partindo duma curva elíptica estuda-se alguns conjuntos infinitos de números primos que são recursivos.
Na parte prática da dissertação definimos alguns algoritmos e calculamos alguns elementos destes conjuntos.

Palavras-chave: 10^{o} Problema de Hilbert, modelo diofantino, curva elíptica.

Abstract

This thesis studies the article Bjorn Poonen, Hilbert‘s tenth problem and Mazur‘s conjecture for large subrings of $\mathbb{Q}$ and investigates computationally the various sets featured in the article. We begin by introducing the basic theory of algebraic varieties and elliptic curves, concepts necessary for understanding the article under consideration. In Poonen article the absence of an algorithm to decide if polynomial equations with coefficients in certain subrings of $\mathbb{Q}$ have solutions in those subrings is proved, that is, Hilbert‘s tenth problem for these rings has a negative solution. The idea of proof is to use an elliptic curve to build a diophantine model of the ring $\mathbb{Z}$. To this end, we study some infinite recursive sets of primes that are built from an elliptic curve.
The idea of proof is of an elliptic curve from building a diophantine model of the $\mathbb{Z}$ ring To this end, and from a elliptic curve is studied some infinite sets of primes that are recursive. In the practical part of the thesis we define algorithms and calculate some elements of these sets.

Keywords: Hilbert‘s tenth problem, diophantine model, elliptic curve.

Conteúdo

Capítulo 1

Introdução

Figura 1.1: David Hilbert

O 10^o problema de Hilbert [Hil02] , enunciado em 1900, continua passados mais de cem anos a ser objeto da investigação matemática.
Na sua versão original pretendia-se saber se existe um algoritmo geral que decide se uma equação polinomial em várias variáveis, com coeficientes no anel dos números inteiros tem soluções no anel dos números inteiros. O problema foi generalizado a outros anéis, nomeadamente ao anel dos números racionais.
O âmbito do nosso estudo é um trabalho de Bjorn Poonen de 2003, em que o 10^o problema de Hilbert para determinados subanéis dos números racionais tem uma resposta negativa.

Traçaram-se essencialmente 2 grandes objectivos para este trabalho:

- 10^o problema de Hilbert e a contribuição de Bjorn Poonen:

  (1) Contextualização do tema.
  (2) Estudo detalhado do artigo de Bjorn Pooenen. Demonstração do teorema de Poonen.

- Estudo computacional duma curva elíptica na perspetiva do artigo de Bjorn Poonen:

  (1) Verificação que a curva elíptica escolhida cumpre os requisitos definidos pela curva de Poonen.
  (2) Cálculos de pontos de coordenadas racionais.
  (3) Verificação para diversos casos concretos de [Poo03a, Corolário 3.3] e de [Poo03a, Lema 3.4].

(4) Construção de um subconjunto de ℓ_i com verificação das condições impostas [Poo03b, Secção 7].

(5) Construção de subconjuntos de T_1, T_2 e S.

Sendo o artigo de Bjorn Poonen construído com base em pontos com coordenadas racionais de curvas elípticas, procedeu-se previamente à análise dos conceitos básicos de variedades algébricas e curvas elípticas, conhecimentos indispensáveis à interpretação do artigo.
Para o estudo computacional seleccionou-se um software da área da matemática (Sage). O facto dos objetos matemáticos *nP* (pontos racionais de curvas elípticas), crescerem em número de dígitos à medida que *n* aumenta, teve impacto na quantidade de resultados obtidos, como se detalha no respetivo capítulo.

Em resultado desta metodologia, vem o trabalho organizado em 3 grandes áreas:

- Revisão Bibliográfica - variedades algébricas e curvas elípticas (cap. 2 e 3):

- O 10^{o} problema de Hilbert e o artigo de Bjorn Poonen (cap. 4).

- Estudo computacional da curva elíptica na perspetiva do artigo de Bjorn Poonen (cap. 5).

Capítulo 2

Variedades Algébricas

Justifica-se uma revisão dos conceitos básicos de variedades algébricas afins e projetivas para um melhor entendimento do desenvolvido no capítulo seguinte sobre curvas elípticas.
Começamos pela noção prévia de fecho algébrico.

2.1 Fecho Algébrico

Ao falarmos do fecho algébrico dum corpo K, estamos a referir-nos a uma extensão algébrica de K que é algébricamente fechada.

DEFINIÇÃO 2.1.1. *Seja K um corpo. O fecho algébrico de K é o menor corpo algébricamente fechado que contém K. Representamo-lo por $\overline{K}$.*

EXEMPLOS 2.1.2. O fecho algébrico dos racionais $\mathbb{Q}$ é o corpo dos números algébricos $\mathbb{Q}_{alg}$; dos reais $\mathbb{R}$ é o corpo dos números complexos $\mathbb{C}$ e dos complexos $\mathbb{C}$ é o corpo dos próprios números complexos $\mathbb{C}$.

$$\overline{\mathbb{Q}} = \mathbb{Q}_{alg}, \qquad \overline{\mathbb{R}} = \mathbb{C}, \qquad \overline{\mathbb{C}} = \mathbb{C}$$

2.2 Espaço Afim

Vamos definir conjuntos algébricos, variedades algébricas e pontos singulares/não singulares. Veremos alguns exemplos orientados a equações em 2 incógnitas, que são o foco deste estudo.

2.2.1 Conjuntos Algébricos Afins

Comecemos por definir um espaço afim:

DEFINIÇÃO 2.2.1. *Um n-espaço afim sobre um corpo K é o conjunto de n-tuplas*

$$\mathbb{A}^n := \mathbb{A}^n(\overline{K}) = \left\{ P = (x_1, ..., x_n) : x_i \in \overline{K} \right\}$$

O conjunto dos K-racionais em $\mathbb{A}^n$ é o conjunto

$$\mathbb{A}^n(K) = \left\{ P = (x_1, ..., x_n) \in \mathbb{A}^n : x_i \in K \right\}$$

Seja o anel dos polinómios em n variáveis, $\overline{K}[X] = \overline{K}[X_1, ..., X_n]$com coeficientes em $\overline{K}$ e $I \subseteq \overline{K}[\overline{X}]$ um ideal. E sendo assim temos que:

DEFINIÇÃO 2.2.2. *Um conjunto algébrico afim V é um conjunto*

$$V = \left\{ P \in \mathbb{A}^n : f(P) = 0 \qquad \forall f \in I \right\}$$

O ideal de V é definido por:

$$I(V) = \left\{ f \in \overline{K}[X] : f(P) = 0 \qquad \forall P \in V \right\}$$

Um conjunto algébrico, diz-se definido sobre um corpo K, que representamos por V/K se o seu ideal pode ser gerado por polinómios em $K[X]$, ou seja $I(V) = I(V/K)$ onde:

$$I(V/K) = \left\{ f \in K[X] : f(P) = 0 \qquad \forall P \in V \right\} = I(V) \cap K[X]$$

Note-se que para o conjunto dos K-racionais pontos do conjunto algébrico V se tem

$$V(K) = V \cap \mathbb{A}^n(K)$$

FACTO 2.2.3. *[Sil08, Nota* 1.1*]*
Todos os ideais $I \subseteq \overline{K}[\overline{X}]$ são finitamente gerados, isto é, existem $f_i, ..., f_n \in I$ tais que $\langle f_1, \ ..., \ f_n \rangle = I$. Em particular se $\langle f_1, \ ..., \ f_n \rangle = I(V)$, então

$$P \in V \text{ sse } f_1(P) = \ldots = f_n(P) = 0$$

EXEMPLOS 2.2.4. Analisemos os seguintes conjuntos algébricos V definidos sobre um corpo K, no plano afim.

(1) Seja o conjunto algébrico V definido pela equação

$$V : X^2 - Y^2 = 1$$

Se considerarmos $K = \mathbb{C}$, teremos V

$$V(\mathbb{C}) = \left\{ (x,y) \in \mathbb{C}^2 : X^2 - Y^2 - 1 = 0 \right\}$$

a que corresponde a figura geométrica: hiperbole.
Se considerarmos $K = \mathbb{Q}$, teremos V

$$V(\mathbb{Q}) = \left\{ (x,y) \in \mathbb{Q}^2 : X^2 - Y^2 - 1 = 0 \right\}$$

(2) Seja o conjunto algébrico V definido pela equação

$$V : X^n + Y^n = 1$$

O conjunto $V(\mathbb{Q})$, foi estudado por Fermat que afirma no denominado Último Teorema de Fermat (este Teorema só foi provado pelo inglês Andrew Wiles em 1995) quais as soluções em $\mathbb{Q}$ para valores de $n \geq 3$

$$V(\mathbb{Q}) = \begin{cases} (1,0),(0,1) & \text{se } n \text{ é par} \\ (\pm 1,0),(0,\pm 1) & \text{se } n \text{ é ímpar} \end{cases}$$

(3) Seja o conjunto algébrico V definido pela equação

$$V : Y^2 = X^3 + 17$$

O conjunto $V(\mathbb{Q})$ tem infinitas soluções, de que são exemplos:

$$V(\mathbb{Q}) = (-2,3),(5234,378661),(\frac{137}{64},\frac{2651}{512})$$

[Sil08, Exemplo 1.3.3]

2.2.2 Variedades Algébricas Afins

Uma variedade algébrica afim é um conjunto algébrico que satisfaz determinados requisitos. Este conceito é importante no trabalho, dado que as curvas elípticas são variedades algébricas. Vejam-se as seguintes definições:

DEFINIÇÃO 2.2.5. *Um variedade algébrica afim V é um conjunto algébrico afim, tal que, $I(V)$, é um ideal primo de $\overline{K}[X]$*

DEFINIÇÃO 2.2.6. *Seja V uma variedade afim, a dimensão de V , que representamos por dim(V) é o grau transcendental [Row14, Definição] de $\overline{K}[V]$ sobre $\overline{K}$*

EXEMPLO 2.2.7. Se $V \subset \mathbb{A}^n$, é dado por uma equação polinomial não constante

$$f(X_1,...,X_n) = 0$$

então, temos que $dim(V) = n-1$. Todos os exemplos de 2.2.4 na página 4 têm dimensão 1.

2.2.3 Pontos Singulares e Não Singulares em Variedades Algébricas Afins

Como veremos adiante a singularidade/não singularidade das variedades algébricas é uma classificação determinante no estudo das curvas elípticas. Sendo assim, temos:

DEFINIÇÃO 2.2.8. *Seja V uma variedade algébrica afim , $P \in V$ e $f_1,...,f_m \in \overline{K}[X]$ um conjunto de geradores para $I(V)$. Diz-se que V é não singular no ponto P se a matriz $m \times n$*

$$(\frac{\partial f_i}{\partial X_j}(P)) \text{ onde } 1 \leq i \leq m \text{ e } 1 \leq j \leq n$$

tem característica $= n - dim(V)$. Se V for não-singular em todos os pontos P, diz-se que V é não-singular.

EXEMPLO 2.2.9. Seja V dado por uma equação polinomial não constante

$$f(X_1,...,X_n) = 0.$$

Teremos então $dim(V) = n-1$. Estando $P \in V$ então P é um ponto singular se e só se

$$(\frac{\partial f}{\partial X_1}(P)) = ... = (\frac{\partial f}{\partial X_n}(P)) = 0$$

EXEMPLOS 2.2.10. Cálculo da existência de pontos singulares em variedades algébricas afins

(1) Seja V uma variedade algébrica dada pela equação polinomial não constante

$$V:\ Y^2 = X^3$$

Para calcularmos a existência de pontos $P \in V$ que sejam singulares, determinamos as derivadas parciais para X e Y:

$$\frac{\partial f}{\partial X} = 3X^2 \qquad \frac{\partial f}{\partial Y} = 2Y$$

Sendo P um ponto em V, temos que $\frac{\partial f}{\partial X}(P) = \frac{\partial f}{\partial Y}(P) = 0$ são os pontos singulares. Resolvendo as equações, obtemos $x = 0$ e $y = 0$ sendo $P(0,0)$ a solução das derivadas parciais. Verificamos que $P(0,0)$ é igualmente um ponto em V. Conclusão: $P(0,0)$ é um ponto singular e é unico em $V:\ Y^2 = X^3$

(2) Seja V uma variedade algébrica dada pela não-constante equação polinomial não constante

$$V:\ Y^2 = X^3 + X$$

Para calcularmos a existência de pontos $P \in V$ que sejam singulares, determinamos as derivadas parciais para X e Y:

$$\frac{\partial f}{\partial X} = 3X^2 + 1 \qquad \frac{\partial f}{\partial Y} = 2Y$$

Sendo P um ponto em V, temos que $\frac{\partial f}{\partial X}(P) = \frac{\partial f}{\partial Y}(P) = 0$ são os pontos singulares. Resolvendo as equações, obtemos $x = \pm\frac{1}{\sqrt{3}}i$ e $y = 0$ sendo $P(\pm\frac{1}{\sqrt{3}}i, 0)$ a solução das derivadas parciais. Mas $f(\pm\frac{1}{\sqrt{3}}i, 0) \neq 0$. Conclusão: não existem ponto singulares em $V:\ Y^2 = X^3 + X$

2.3 Espaço Projetivo

2.3.1 Polinómios Homogéneos

Na definição das curvas no plano $\mathbb{P}^2$, utilizaremos polinómios em 3 variáveis. Dado que os pontos em $\mathbb{P}^2$ são representados por triplas homogéneas. Definamo-lo a um nivel geral, para *n* variáveis

DEFINIÇÃO 2.3.1. *Um polinómio $f \in \overline{K}[\overline{X}] = \overline{K}[X_1, ..., X_n]$ é homogéneo de grau d se:*

(1) $f(\lambda x_0, ..., \lambda x_n) = \lambda^d f(x_0, ..., x_n)$*para todo o* $\lambda \in \overline{K}$*. Esta identidade é equivalente a dizer que f é uma combinação linear de monómios em n variáveis em que a soma dos expoentes das variáveis é igual a d*

(2) *Um ideal* $I \subseteq \overline{K}[\overline{X}]$ *é homogéneo se é gerado por polinómios homogéneos*

2.3.2 Conjuntos Algébricos Projetivos

DEFINIÇÃO 2.3.2. *Definimos*

$$\mathbb{P}^n := \mathbb{P}^n(\overline{K}) = \mathbb{A}^{n+1}/\ \sim$$

onde

$$(x_0, ..., x_n)\ \sim\ (y_0, ..., y_n)$$

se existe

$$\lambda \in \overline{K}:\ x_i = \lambda y_i \qquad \forall i$$

em que denotamos $[x_0, ..., x_n]$ *por classe de equivalência*

DEFINIÇÃO 2.3.3. *Um conjunto algébrico projectivo V é:*

$$V = \left\{ [x_0, ...x_n] \in \mathbb{P}^n :\ \ f(x_0, ...x_n) = 0 \qquad \forall f \in I \right\}$$

onde I é um ideal homogéneo.
O ideal de V, denotado por $I(V)$ *é o ideal de* $\overline{K}[X]$

$$I(V) = \left\{ f \in \overline{K}[X] \text{: } f \text{ é um polinómio homógeneo e } f(P) = 0\ \ \forall P \in V \right\}$$

V, diz-se definido sobre um corpo K, que representamos por V/K *se o seu ideal pode ser gerado por polinómios homogéneos em* $K[X]$ *e definimos*

$$I(V/K) = \left\{ f \in K[X] :\ f \text{ é um polinómio homogéneo e } f(P) = 0 \qquad \forall P \in V \right\} = I(V) \cap K[X]$$

Se V está definido sobre K tem-se que o conjunto dos K-racionais pontos é

$$V(K) = V \cap \mathbb{P}^n(K)$$

EXEMPLOS 2.3.4. São exemplos de conjuntos algébricos projetivos:

(1) A linha projetiva em $\mathbb{P}^2$

$$V : aX + bY + cZ = 0$$

(2) A curva elíptica em $\mathbb{P}^2$

$$V : Y^2Z - X^3 + 5X^2Z - 8Z^3 = 0$$

2.3.3 Variedades Algébricas Projetivas

DEFINIÇÃO 2.3.5. *Uma variedade algébrica projetiva V é um conjunto algébrico projetivo, tal que $I(V)$ é um ideal primo.*

FACTO 2.3.6. *Seja V um variedade projetiva e para cada $i = 0,...n$ seja $\phi_i : \mathbb{A}^n \to \mathbb{P}^n$ dada por*

$$\phi_i(x_1,..,x_n) = [x_1,..,x_{i-1},1,x_i,...,x_n]$$

Tem-se, identificando $\mathbb{A}^n$ com a sua imagem por ϕ_i,

$$V \cap \mathbb{A}^n \text{ é uma variedade algébrica afim}$$

Analisemos alguns exemplos de aplicação deste Facto, para obtenção da variedade algébrica em função do indice de ϕ utilizado no mapeamento de $\phi_i : \mathbb{A}^n \to \mathbb{P}^n$

EXEMPLOS 2.3.7. Dada a variedade algébrica projetiva $V : Y^2Z = X^3 + 17Z^3$

(1) seja $\phi_0 : \mathbb{A}^2 \to \mathbb{P}^2$

$$\text{em que : } \phi_0(x_1,x_2) = [1,x_1,x_2]$$

então,

$$V \cap \mathbb{A}^2 : Y^2Z = 1 + 17Z^3$$

(2) seja $\phi_1 : \mathbb{A}^2 \to \mathbb{P}^2$

$$\text{em que : } \phi_1(x_1,x_2) = [x_1,1,x_2]$$

então,

$$V \cap \mathbb{A}^2 : Z = X^3 + 17Z^3$$

(3) seja $\phi_0 : \mathbb{A}^2 \to \mathbb{P}^2$

$$\text{em que : } \phi_2(x_1,x_2) = [x_1,x_2,1]$$

então,

$$V \cap \mathbb{A}^2 = Y^2 = X^3 + 17$$

Reciprocamente, se tivermos V/K, uma variedade algébrica afim, então existe uma variedade projetiva $\bar{V}$, o fecho projetivo de V,

$$I(\bar{V}) = \langle \{ f^* : f \in I(V) \} \rangle$$

onde para $f \in K[X]$ tem-se:

$$f^*(X_0,...,X_n) = X_i^d \, f[\frac{X_0}{X_i}, \frac{X_1}{X_i}, ..., \frac{X_{i-1}}{X_i}, \frac{X_{i+1}}{X_i}, ..., \frac{X_n}{X_i}]$$

em que d é o grau de f

FACTO 2.3.8. *[Sil08, Proposição 2.6 a)]*
Se V é uma variedade algébrica afim, então $\bar{V}$ é uma variedade algébrica projetiva e

$$V = \bar{V} \cap \mathbb{A}^n$$

FACTO 2.3.9. *[Sil08, Proposição 2.6 b)]*
Se $\bar{V}$ é uma variedade algébrica projetiva, então:

$$\bar{V} \cap \mathbb{A}^n \text{ é uma variedade algébrica ou é vazio}$$

2.3.4 Pontos No Infinito em Variedades Algébricas Projetivas

DEFINIÇÃO 2.3.10. *Os pontos $\bar{V} \setminus V$, são chamados os pontos no infinito*

$$V \subseteq \mathbb{A}^n \to \mathbb{P}^n \supseteq \bar{V} \supseteq V$$

Vejamos alguns exemplos de cálculo de pontos no infinito para variedades algébricas projetivas em $\mathbb{P}^n$ para os 3 possiveis modelos projetivos:

EXEMPLOS 2.3.11. Seja $V: Y^2 = X^3 + X^2$ uma variedade algébrica afim em $\mathbb{A}^2$
Fazendo a homogeneização do polinómio (grau 3) $Y^2 = X^3 + X^2$, obtemos: $Y^2Z = X^3 + X^2Z$, variedade algébrica projetiva

(1) Para o modelo projetivo $(x/z, y/z, z)$, em que $z = 1$ para os pontos $\in V$ e $z = 0$ para os pontos no infinito temos (notando que se $z = 0$, então $X^3 = 0$ logo $x = 0$ e então $y = 1$) :

$$\bar{V} = \Big\{ [X_1, X_2, 1] \ : \ (X_1, X_2) \in V \Big\} \bigcup \Big\{ [0,1,0] \Big\}$$

(2) Para o modelo projetivo $(x/y, y, z/y)$, em que $y = 1$ para os pontos $\in V$ e $y = 0$ para os pontos no infinito temos (notando que se $y = 0$, então $x^3 + x^2 z = 0$ logo $z = -x$ e então $x = 1 \wedge z = -1$) :

$$\bar{V} = \Big\{ [X_1, 1, X_3] \ : \ (X_1, X_3) \in V \Big\} \bigcup \Big\{ [1,0,-1] \Big\}$$

(3) Para o modelo projetivo $(x, y/x, z/x)$, em que $x = 1$ para os pontos $\in V$ e $x = 0$ para os pontos no infinito temos (notando que se $x = 0$, então $Y^2Z = 0$ então $y = 0 \vee z = 0$ logo $((y = 0 \wedge z = 1) \vee (y = 1 \wedge z = 0))$):

$$\bar{V} = \Big\{ [1, X_2, X_3] \ : \ (X_2, X_3) \in V \Big\} \bigcup \Big\{ [0,0,1], [0,1,0] \Big\}$$

2.3.5 Pontos Singulares e Não-Singulares em Variedades Algébricas Projetivas

As variedades algébricas projetivas V, à semelhança das variedades algébricas afins, têm pontos singulares e não singulares, classificando-se igualmente de variedades não singulares caso todos os pontos em V, sejam não singulares. As definições são iguais às já referidas para as variedades afins (ver definição 2.2.8 na página 5 e exemplo 2.2.9 na página 5). Exemplifiquemos o cálculo de pontos singulares numa variedade projetiva V

EXEMPLOS 2.3.12. Seja V uma variedade algébrica projetiva dada por uma equação polinomial não-constante

$$V : Y^2Z = X^3 + 17Z^3$$

Para calcularmos a existência de pontos $P \in V$ que sejam singulares, determinamos as derivadas parciais para X, Y e Z

$$\frac{\partial f}{\partial X} = 3X^2 \qquad \frac{\partial f}{\partial Y} = 2YZ \qquad \frac{\partial f}{\partial Z} = Y^2 - 51Z^2$$

Sendo P um ponto em V, temos que $\frac{\partial f}{\partial X}(P) = \frac{\partial f}{\partial Y}(P) = \frac{\partial f}{\partial Z}(P) = 0$ são os pontos singulares. Resolvendo as equações, obtemos $x = 0$, $y = 0$ e $z = 0$. As coordenadas do nosso(s) ponto(s) são $P(0:0:0)$. A coordenada tripla de zeros não é um ponto do plano projetivo. Conclusão: Não existem pontos singulares em $V: \ Y^2Z = X^3 + 17Z^3$

Capítulo 3

Curvas Elípticas

3.1 Definições

Das variadas definições de curvas elípticas, optamos pela dada pela equação de Weierstrass. Outras de carácter mais geral, poderiam ser aduzidas mas reduzir-se-iam à definição apresentada.

DEFINIÇÃO 3.1.1. *Seja K um corpo, uma curva elíptica sobre K é uma variedade projetiva E sobre K, não-singular, de dimensão 1 dada pela equação de Weierstrass (em coordenadas não projetivas):*

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

$$onde \;\; a_1, a_2, a_3, a_4, a_6 \in K$$

Nota:

(1) Se $ch(K) \neq 2$, temos uma simplificação da equação de Weierstrass:

$$E : Y^2 = 4X^3 + b_2X^2 + 2b_4X + b_6$$

(2) Se $ch(K) \neq 2, 3$, temos uma simplificação adicional da equação de Weierstrass:

$$E : Y^2 = X^3 + 27c_4X - 59c_6$$

Seja $\triangle_E$ o discriminante da equação de Weierstrass e j_E a invariante de E, obtidos de acordo com as seguintes definições:

(1) $b_2 = (a_1)^2 + 4a_2$

(2) $b_4 = 2(a_4) + a_1a_3$

(3) $b_6 = (a_3)^2 + 4a_6$

(4) $b_8 = (a_1)^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2(a_3)^2 - (a_4)^2$

(5) $c_4 = (b_2)^2 - 24b_4$

(6) $c_6 = (b_2)^3 + 36b_2b_4 - 216b_6$

(7) $\triangle_E = (b_2)^2b_8 - 8(b_4)^3 - 27(b_6)^2 + 9b_2b_4b_6$

(8) $j_E = \frac{(c_4)^3}{\triangle_E}$

temos ainda, que:

(9) $4b_8 = b_2b_6 - (b_4)^2$

(10) $1728\triangle_E = (c_4)^3 - (c_6)^2$

PROPOSIÇÃO 3.1.2. *As curvas obtidas pela equação de Weierstrass são classificadas em função de* $\triangle$ *e de* c_4 *da seguinte forma:*

(1) *Se* $\triangle_E \neq 0$ *então a curva é não singular.*

(2) *Se* $\triangle_E = 0$ *e* $c_4 \neq 0$ *então a curva é um nó.*

(3) *Se* $\triangle_E = 0$ *e* $c_4 = 0$ *então a curva é um cúspide.*

3.2 Exemplos

Vamos exemplificar os 3 casos referidos anteriormente de equações de Weierstrass:

EXEMPLO 3.2.1. 1º caso E: $Y^2 = X^3 + 1$

Cálculo de $\triangle_E$ de acordo com as definições anteriores:

(1) $\triangle_E = (b_2)^2b_8 - 8(b_4)^3 - 27(b_6)^2 + 9b_2b_4b_6$

(2) $a_1 = 0$; $a_2 = 0$; $a_3 = 0$;$a_4 = 0$; $a_6 = 1$

(3) $b_2 = 0$; $b_4 = 0$; $b_6 = 4$;$b_8 = 0$

(4) $c_4 = 0$

(5) $\triangle_E = 27 \times 4^2$, logo diferente de zero

Temos assim:

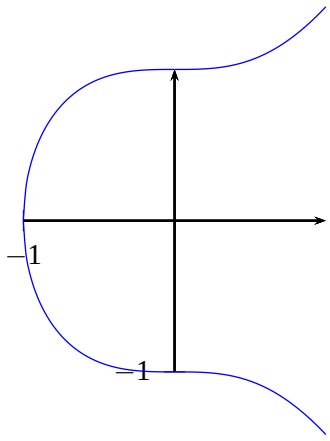

Figura 3.1: Curva Não Singular

EXEMPLO 3.2.2. 2º caso E: $Y^2 = X^3 + X^2$

Calculo de $\triangle_E$ de acordo com as definições anteriores:

(1) $\triangle_E = (b_2)^2 b_8 - 8(b_4)^3 - 27(b_6)^2 + 9b_2 b_4 b_6$

(2) $a_1 = 0$; $a_2 = 1$; $a_3 = 0$;$a_4 = 0$; $a_6 = 0$

(3) $b_2 = 4$; $b_4 = 0$; $b_6 = 0$;$b_8 = 0$

(4) $c_4 = 4^3$;

(5) $\triangle_E = 0 \wedge c_4 \neq 0$

Temos assim:

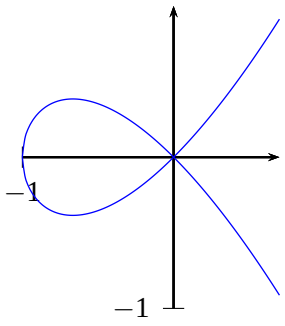

Figura 3.2: Curva com Nó

EXEMPLO 3.2.3. 3º caso E: $Y^2 = X^3$
Calculo de $\triangle_E$ de acordo com as definições anteriores:

(1) $\triangle_E = (b_2)^2 b_8 - 8(b_4)^3 - 27(b_6)^2 + 9b_2 b_4 b_6$

(2) $a_1 = 0$; $a_2 = 0$; $a_3 = 0$;$a_4 = 0$; $a_6 = 0$

(3) $b_2 = 0$; $b_4 = 0$; $b_6 = 0$;$b_8 = 0$

(4) $c_4 = 0$

(5) $\triangle_E = 0 \wedge c_4 = 0$

Temos assim:

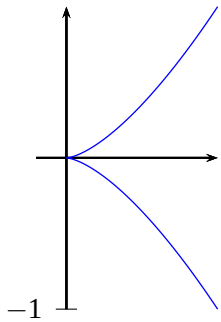

Figura 3.3: Curva com Cúspide

3.3 Curvas Elípticas - Geometria

3.3.1 Pontos Racionais numa Curva Elíptica

Os pontos racionais na curva elíptica são o objeto principal do nosso estudo.

Um ponto no plano (x,y) é um número racional, se quer x quer y forem números racionais e uma reta definida pela equação $ax+by+c=0$ é uma reta racional se $a,b,c \in \mathbb{Q}$.

Poderemos aplicar o princípio geométrico, que se tivermos encontrado 2 pontos racionais na curva elíptica, os pontos P e Q, então, na generalidade podemos encontrar um terceiro ponto, $P \otimes Q$ igualmente racional. Será suficiente desenhar a reta que une esses 2 pontos e o terceiro ponto, igualmente racional é o ponto de interseção da reta racional com a curva elíptica. Se tivermos um único ponto P e desenharmos a reta tangente à curva elíptica no ponto P,é como se desenhássemos uma reta de P a P. A reta tangente encontra a curva elíptica num ponto racional $P \otimes P$.

Mas, nem sempre existe no plano afim o 3º ponto de intersecção, daí a necessidade de considerarmos um ponto no infinito. Considerando geometria projetiva, façamos a homogeneização da equação $y^2 = x^3+ax+b$ em que $x = \frac{X}{Z}$ e $y = \frac{Y}{Z}$, obtendo $Y^2Z = X^3 + aXZ^2 + bZ^3$. Qual é então a interseção desta curva com a reta no infinito em que $Z = 0$? É um ponto no infinito não singular, contado como ponto racional, que tomamos como o elemento zero e que denominaremos por O. Adicionamos então esse ponto à curva elíptica e obtemos:

$$E = \{(x,y) : y^2 = x^3 + ax + b\} \cup O$$

Uma linha vertical interseta a curva elíptica em 2 pontos do plano xy e no ponto O. Uma linha não vertical interseta a linha curva em três pontos do plano xy.

Consideremos a seguinte curva elíptica dada pela equação de Weierstrass, para estudo das operações de soma de pontos racionais em E.

$$E : Y^2 = X^3 - 5X + 8$$

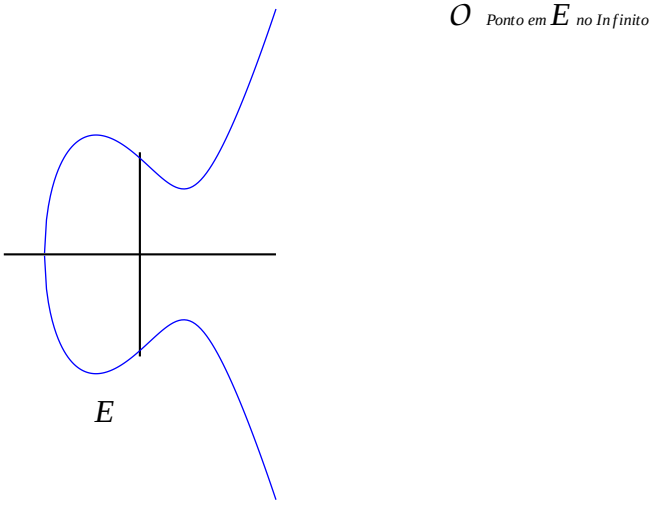

Figura 3.4: Curva Elíptica

3.3.2 Soma de Pontos P e Q numa Curva Elíptica

Consideremos os pontos racionais P e Q pertencentes a E e representemo-los no gráfico da curva

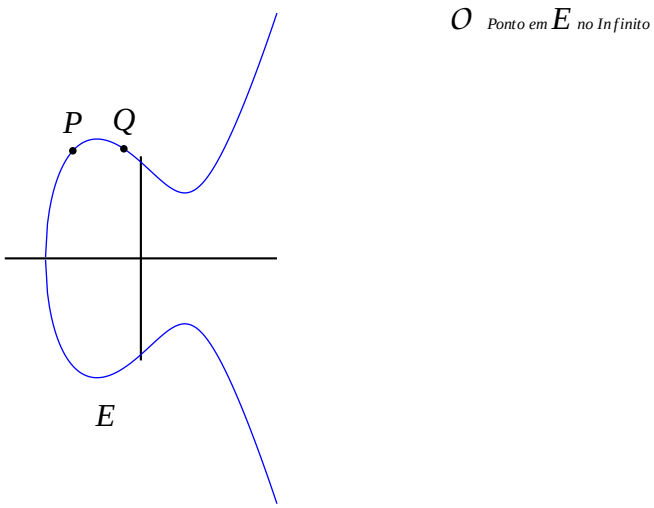

Figura 3.5: Curva Elíptica - Soma de P e Q (1)

Tracemos a reta que contém os pontos P, Q. Esta reta interseta a curva elíptica E num terceiro ponto racional, $P \otimes Q$.

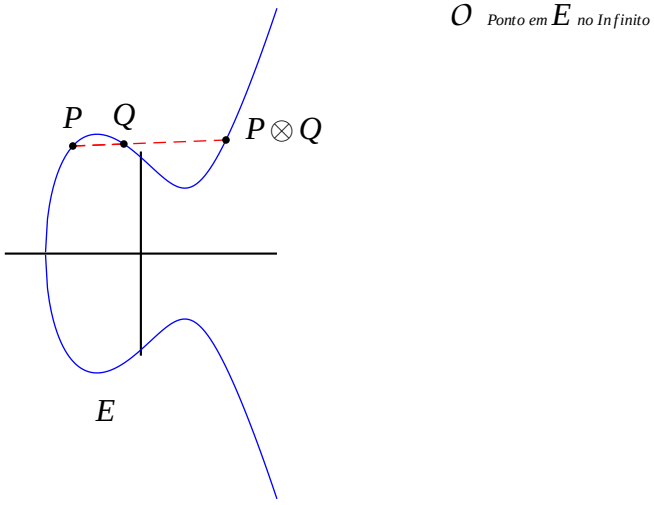

Figura 3.6: Curva Elíptica - Soma de P e Q (2)

Tracemos a reta paralela ao eixo de y, ligando desta forma o ponto no infinito O com $P \otimes Q$. A reta interseta a curva elíptica num terceiro ponto racional $P \oplus Q$.
Definimos este ponto, como a soma de P e Q, igualmente representado por $P + Q$.

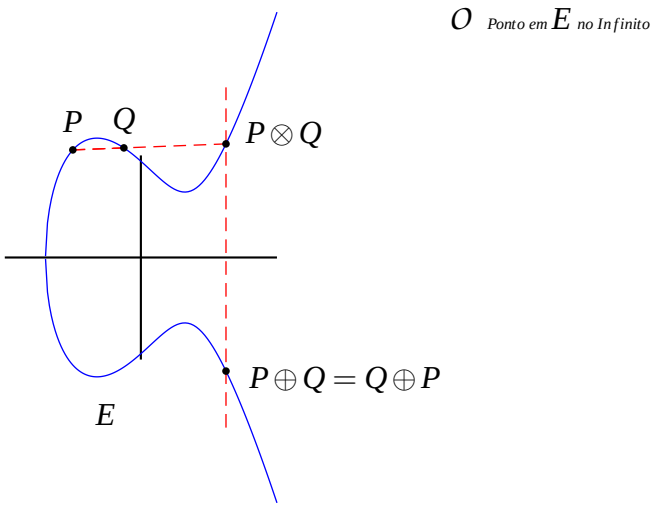

Figura 3.7: Curva Elíptica - Soma de P e Q (3)

3.3.3 Soma de P com P numa Curva Elíptica

Para adicionarmos o ponto racional P com P, tracemos a reta tangente à curva elíptica em P. Esta reta interseta a curva no ponto racional $P \otimes P$. Tracemos a reta paralela ao eixo de y, ligando desta forma o ponto no infinito O com $P \otimes P$. A reta interseta a curva elíptica num terceiro ponto racional $P \oplus P$.
Definimos assim a soma de P com P que representamos por $P \oplus P$ ou $2P$.

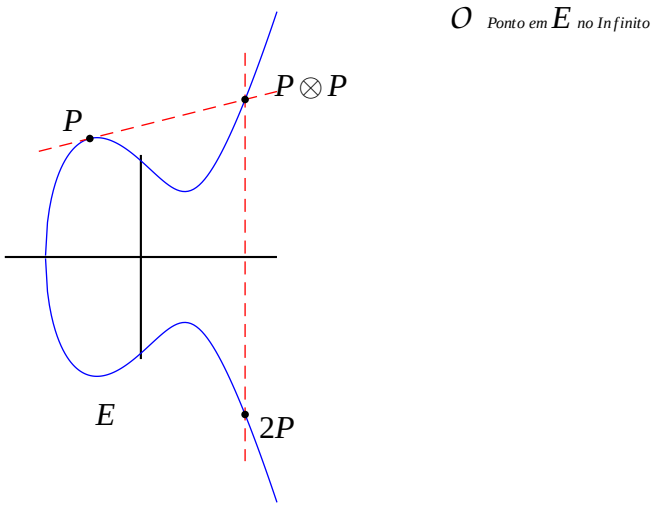

Figura 3.8: Curva Elíptica - Soma de P com P

3.3.4 Simétrico de P numa Curva Elíptica

Tracemos a reta parela ao eixo de y , ligando o ponto racional P ao ponto racional no infinito O. A reta interseta a curva elíptica ($ch(K) \neq 2$) E num ponto racional. Representemo-lo por $-P$. Definimos assim o simétrico de P que representamos por $-P$.

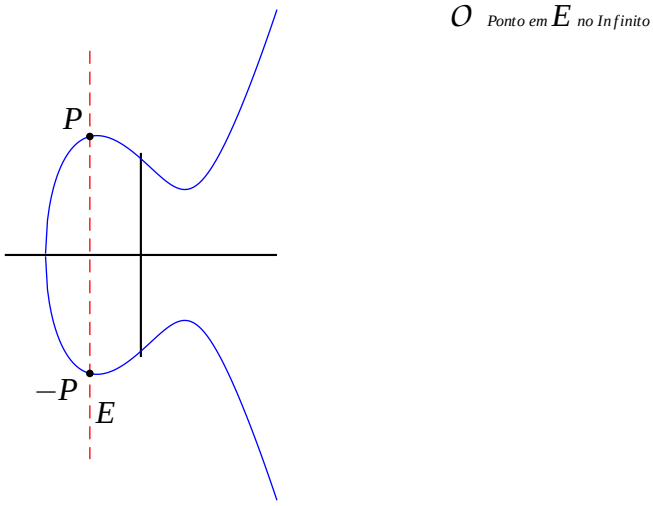

Figura 3.9: Curva Elíptica - Simétrico P: $-P$

3.3.5 Soma de P com $-P$ numa Curva Elíptica

Tracemos a reta parela ao eixo de y , ligando o ponto racional P e o seu simétrico $-P$. A reta interseta a curva elíptica no ponto racional no infinito O, ponto este que está em toda a reta vertical. Definimos assim a soma de P com o seu simétrico $-P$ que é O.

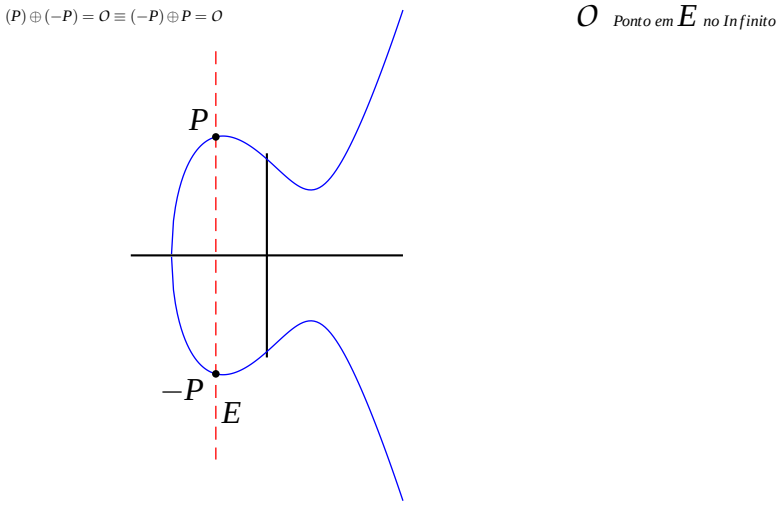

Figura 3.10: Curva Elíptica - Soma de P e $-P$

3.4 Curvas Elipticas - Álgebra

3.4.1 Fórmulas de Adição

Consideremos a equação geral da reta:

$$Y = \lambda X + u \qquad \text{(equação da reta)}$$

$$\text{em que } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ e } u = y_1 - \lambda x_1 = y_2 - \lambda x_2$$

Consideremos a curva elíptica E, sobre um corpo K, dada pela equação

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad \text{(equação de Weierstrass)}$$

Seja $P(x_1, y_1) \in E$

(1) Simétrico de P
Para calcularmos o simétrico de P, que representaremos por $-P$, precisamos de encontrar a linha definida por P e o ponto no infinito O e determinar o terceiro ponto de interseção.

(a) A reta que contém P e O é a reta vertical em P, logo é $X = x_1$

(b) Conhecida a coordenada X determinamos a coordenada Y por substituição na [equação de Weierstrass]:

$$Y^2 + (a_1x_1 + a_3)Y - x_1^3 + a_2x_1^2 + a_4x_1 + a_6 = 0$$

Sabendo que uma solução é $Y = y_1$, a outra solução é então $-a_1x_1 - a_3 - y_1$. isto é:

$$-P = (x_1, -a_1x_1 - a_3 - y_1)$$

No caso em que char$K \neq 2$, em que $a_1 = a_3 = 0$ o valor de Y é o simétrico de y_1

(2) Adição de Pontos P e P
Representaremos a adição de $P + P$ por $2P$. Para calcularmos as coordenadas de $2P$, que designaremos por (x_2, y_2), consideramos os seguintes passos:

(a) Necessitamos de determinar a equação da reta tangente à curva E no ponto P. O declive da reta tangente à curva no ponto (X, Y)é dado por

$$(\mathrm{d}E/\mathrm{d}X)/(\mathrm{d}E/\mathrm{d}Y) = (3X^2 + 2a_2X - a_1Y + a_4)/(2Y + a_1X + a_3)$$

substituindo temos,

$$\lambda = \frac{3x_1^2 + 2a_2x_1 - a_1y_1 + a_4}{2y_1 + a_1x_1 + a_3}$$

em que a equação da reta tangente em P é $Y = \lambda X - \lambda x_1 + y_1$

(b) Por substituição na [equação de Weierstrass] e fazendo o simétrico do coeficiente de X^2, temos a soma das raizes:

$$\lambda^2 + \lambda a_1 - a_2$$

Então a coordenada x do terceiro ponto de interseção é:

$$x_2 = \lambda^2 + \lambda a_1 - a_2 - 2x_1$$

(c) Então a coordenada y do terceiro ponto de interseção pode ser encontrada pela equação da reta tangente à curva em P:

$$y' = \lambda x_2 - \lambda x_1 + y_1$$

Finalmente, precisamos de calcular o simétrico de (x_3, y') e obtemos:

$$y_2 = -a_1 x_2 - a_3 - \lambda x_2 + \lambda x_1 - y_1$$

(3) Adição de Pontos P e Q
Seja $Q = (x_2, y_2) \in E$ um ponto diferente de P e $-P$. Para calcularmos as coordenadas de $P + Q$, que designaremos por (x_3, y_3), consideramos os seguintes passos:

(a) Determinamos λ, o declive da reta definida pelos pontos P e Q:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ e } u = y_1 - \lambda x_1 = y_2 - \lambda x_2$$

(b) Determinamos u, utilizando as coordenadas de $P(x_1, y_1) \in E$ e o valor de λ calculado anteriormente. Temos assim a equação da reta que passa por P e Q:

$$Y = \lambda X - \lambda x_1 + y_1$$

(c) Por substituição na [equação de Weierstrass], obtemos :

$$(\lambda X - \lambda x_1 + y_1)^2 + (a_1 X + a_3)(\lambda X - \lambda x_1 + y_1) = X^3 + a_2 X^2 + a_4 X + a_6$$

A soma das raízes, isto é o simétrico do coeficiente de X^2 é $\lambda^2 + a_1\lambda - a_2$. Consequentemente

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$$

A coordenada y_3, pode ser encontrada substituindo na equação da reta definida por P e Q

$$y' = \lambda x_3 - \lambda - x_1 + y_1$$

Finalmente, precisamos de calcular o simétrico de (x_3, y') e obtemos:

$$y_3 = -a_1 x_3 - a_3 - \lambda x_3 + \lambda x_1 - y_1$$

3.4.2 Exemplos

Dada a curva elíptica $E : Y^2 = X^3 - 5X + 8$ e o ponto $P(1,2) \in E$, calcular:

(1) Ponto $2P$
Designemos as coordenadas do ponto P por (x_1, y_1)e de $2P$ por (x_2, y_2).

(a) $\lambda = (\mathrm{d}E/\mathrm{d}X)/(\mathrm{d}E/\mathrm{d}Y) = \frac{3x^2+5}{2y} = -1/2$
$x_2 = \lambda^2 + \lambda a_1 - a_2 - 2x_1 = (-1/2)^2 - 2 = -7/4$
$y_2 = -a_1 x_2 - a_3 - \lambda x_2 + \lambda x_1 - y_1 = -(-1/2 \times -7/4) + (-1/2 \times 1) - 2 = -27/8$
então $2P(-7/4, -27/8)$ que passaremos a designar por ponto Q.

(2) Ponto $P + Q$
Designemos as coordenadas dos ponto P por (x_1, y_1)e de Q por (x_2, y_2).

(a) $\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-27/8 - 2}{-7/4 - 1} = 43/22$
$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 = (43/22)^2 - 1 - (-7/4) = 553/121$ $y_3 = -a_1 x_3 - a_3 - \lambda x_3 + \lambda x_1 - y_1 = -(43/22 \times 553/121) + (43/22 \times 1) - 2 = -11950/1331$ então $P + Q(553/121, -11950/1331)$

3.5 Curvas Elipticas - Grupo Comutativo

A adição em E tem as seguintes propriedades, o que torna os pontos de E num grupo comutativo:

(1) $P + O = O + P = P$ $\quad \forall P \in E$

(2) $P + -P = O$ $\quad \forall P \in E$

(3) $P + Q = Q + P$ $\quad \forall P, Q \in E$

(4) $P + (Q + R) = (P + Q) + R$ $\quad \forall P, Q, R \in E$

Vamos ilustrar geometricamente as propriedades enunciadas, com as seguintes figuras:

(1) Propriedade do Elemento Identidade

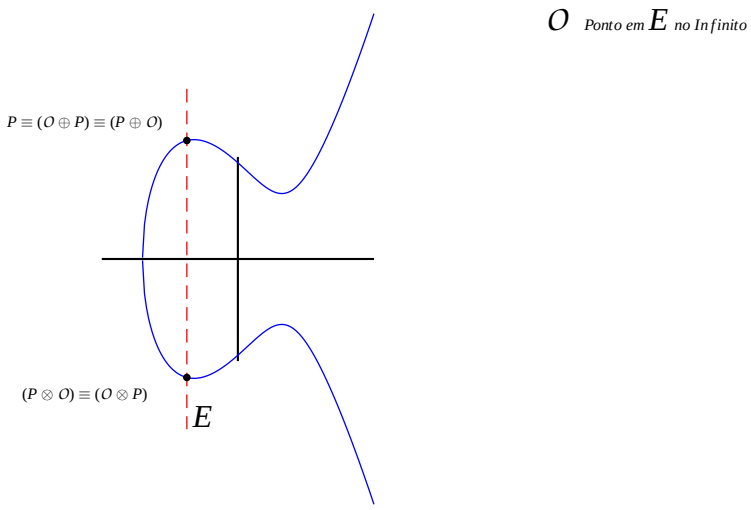

Figura 3.11: Curva Elíptica - Soma P e O

(2) Propriedade do Elemento Simétrico

Ver figura 3.9 na página 16.

(3) Propriedade Comutativa

Ver figura 3.7 na página 15.

(4) Propriedade Associativa

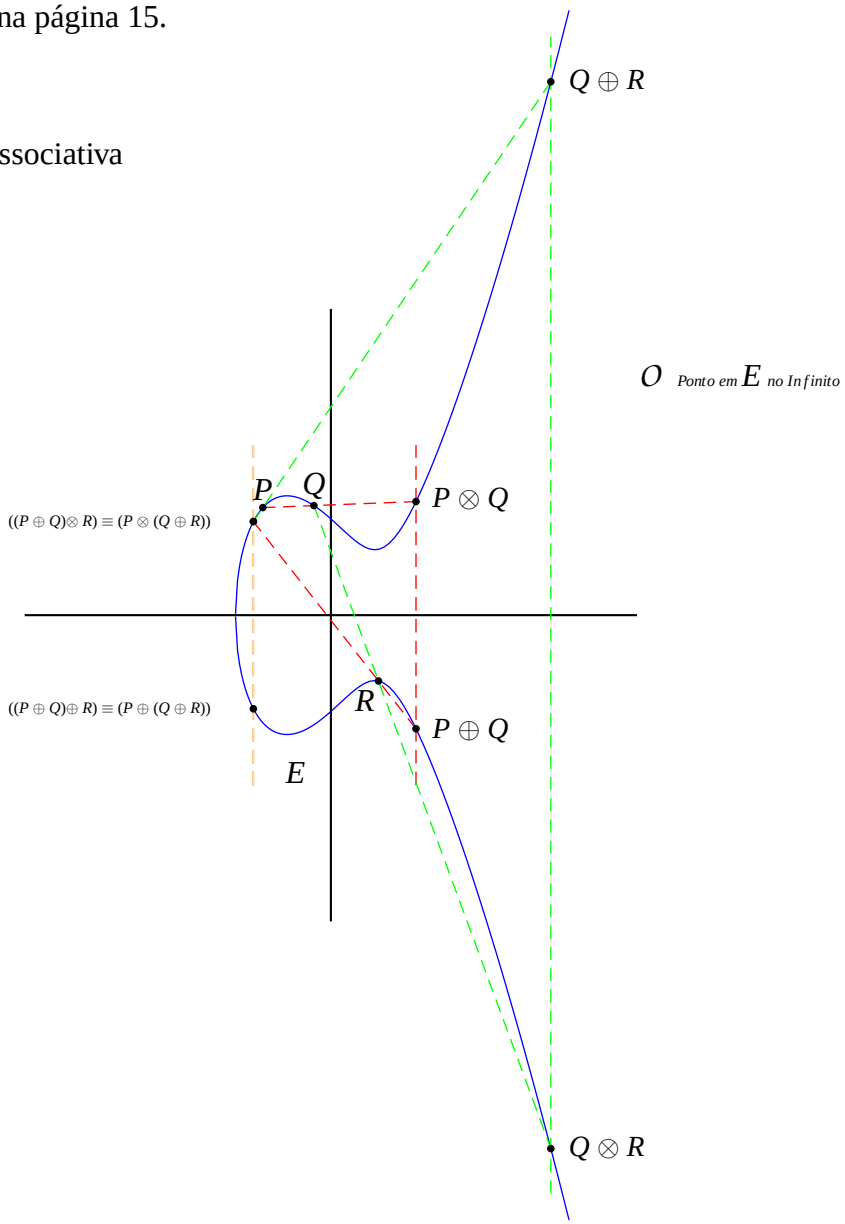

Figura 3.12: Curva Elíptica - Adição: Propriedade Associativa

3.6 Pontos de Ordem Finita ou Infinita

DEFINIÇÃO 3.6.1. *Consideremos uma curva elíptica E, definida pela equação de Weierstrass*

$$E : Y^2 = X^3 + aX^2 + bX + c$$

com um ponto no infinito O considerado o elemento zero.
Um ponto P diz-se de ordem m, se

$$mP = \underbrace{P + P + \ldots + P}_{m \text{ vezes}} = O$$

em que $m'P \neq O$ para todos os inteiros $1 \leq m' < m$
Se existe m, então P é de ordem finita m, caso contrário P é de ordem infinita

3.7 Subgrupos dos m-Pontos de Torsão

Os pontos de ordem finita m constituem um subgrupo dos $m-$pontos de torsão, de acordo com:

DEFINIÇÃO 3.7.1. *Seja E uma curva elíptica, $m \in \mathbb{Z}$ com $m \geq 1$.*
O sub-grupo dos $m-$pontos de torsão em E, i.e. dos pontos de ordem m em E, representado por $E[m]$ é o conjunto dos pontos de E de ordem m

$$E[m] = \left\{ P \in E : mP = O \right\}$$

Podemos então definir o conjunto dos pontos de ordem finita duma curva elíptica:

$$E_{tors} = \bigcup_{m=1}^{\infty} E[m]$$

Considere-se agora o seguinte Facto:

FACTO 3.7.2. *[Sil08, Corolário 6.4]*
Seja E uma curva elíptica sobre um corpo K, $m \in \mathbb{Z}$ com $m \geq 1$

(1) *Se $char(K) = 0$ ou se m é primo com $char(K)$, então:*

$$E[m] = \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$$

(2) *Se $char(K) = p > 0$ então uma das seguintes opções é verdadeira:*

(a) $E[p^e] = O$ *para todo o e=1,2,3,...*
(b) $E[p^e] = \mathbb{Z}/p^e\mathbb{Z}$ *para todo o e=1,2,3,...*

Incidindo o nosso estudo nas curvas elípticas sobre o corpo dos racionais $\mathbb{Q}$, os Teoremas de Nagel-Lutz e Mordell-Weil são extremamente importantes. É de notar, que ambos não são reversíveis, isto é, que a condição de : se e só se não é aplicável.
Comecemos por enunciar o Teorema de Nagell-Lutz:

TEOREMA 3.7.3. *Seja $E/\mathbb{Q}$ uma curva elíptica*

$$E : Y^2 = X^3 + aX^2 + bX + c$$

em que $a, b, c \in \mathbb{Z}$, com $\triangle_E = -4a^3c + a^2b^2 + 18abc - 4b^3 - 27c^2$
Se $P(x,y) \in E(\mathbb{Q})$ tem ordem finita, então:

(1) $x, y \in \mathbb{Z}$

(2) $y = 0$ *ou* $y | \triangle_E$

O Teorema de Nagell-Lutz, pode por exemplo ser utilizado como prova que a ordem de um ponto P é infinita, computando sucessivamente $2P$, $3P$, $\ldots$, mP até que se obtenha um ponto de coordenadas não inteiras.
Precisamos da noção de corpo numérico, antes de enunciarmos o Teorema de Mordell-Weil :

DEFINIÇÃO 3.7.4. *Um corpo numérico K é uma extensão finita do corpo dos números racionais $\mathbb{Q}$*

Assim temos que $\mathbb{Q} \subseteq K$
Enunciemos então o Teorema de Mordell-Weil :

TEOREMA 3.7.5. *Seja K um corpo numérico e E/K uma curva elíptica sobre K, temos que*

$$\text{o grupo } E(K) \text{é finitamente gerado}$$

De facto, $E(K) \cong E_{tors}(K) \times \mathbb{Z}^r$ em que $E_{tors}(K) = \bigcup_{m\geq 0} E[m] \cap E(K)$ e r é o rank de E sobre K

O Teorema de Mordell-Weil mostra-nos como obter todos os pontos K-racionais numa curva elíptica.

3.8 Redução Módulo p

Consideremos a curva elíptica E, definida pela equação de Weierstrass

$$E:\ Y^2 = X^3 + aX^2 + bX + c$$

com coeficientes a,b,c inteiros e consideremos o mapeamento "redução módulo p":

$$\phi : \mathbb{Z} \longrightarrow \frac{\mathbb{Z}}{p\mathbb{Z}} = \mathbb{F}_p \qquad z \mapsto \widetilde{z}$$

Então podemos obter a partir da curva E, uma nova curva $\tilde{E}$, com coeficientes a,b,c num corpo finito $\mathbb{F}_p$:

$$\tilde{E}:\ Y^2 = X^3 + \tilde{a}X^2 + \tilde{b}X + \tilde{c}$$

em que o determinante é calculado por:

$$\triangle_{\tilde{E}} = -4\tilde{a}^2\tilde{c} + \tilde{a}^2\tilde{b}^2 + 18\tilde{a}\tilde{b}\tilde{c} - 9\tilde{b}^3 - 27\tilde{c}^2$$

Sendo a redução módulo p de $\mathbb{Z} \to \mathbb{F}_p$ um homomorfismo, a curva $\tilde{E}$ será uma curva elíptica, se obedecer às seguintes condições:

- $\triangle_{\tilde{E}} \neq 0$
- $p \geq 3$ e $p | \triangle_{\tilde{E}}$

Pelo Teorema de Nagell-Lutz sabemos que os pontos de ordem finita em E, têm coordenadas inteiras. Definamos então o conjunto dos pontos de ordem finita em E, a que chamaremos Φ

$$\Phi = \left\{ P = (x,y) \in E(\mathbb{Q}) : P \text{ tem ordem finita} \right\}$$

Enunciamos então:

TEOREMA 3.8.1. *Seja E uma curva elíptica*

$$E:\ Y^2 = X^3 + aX^2 + bX + c$$

com a, b, $c \in \mathbb{Z}$ e seja $\triangle_E$ o discriminante $\triangle_E = -4a^3c + a^2b^2 + 18abc - 4b^3 - 27c^2$

Seja $\Phi \subseteq E(\mathbb{Q})$ o subgrupo de todos os pontos de ordem finita em E. Para qualquer primo p, seja $P \to \tilde{P}$ o mapeamento redução módulo p

$$\Phi \longrightarrow \tilde{E}(\mathbb{F}_p) \qquad P \to \tilde{P} = \begin{cases} (\tilde{x},\tilde{y}) & \text{se } P = (x,y) \\ O & \text{se } P = O \end{cases}$$

Se p não divide $2\triangle_E$, então a redução módulo p é um isomorfismo de Φ num subgrupo de $\tilde{E}(\mathbb{F}_p)$

Capítulo 4

O 10^{o} Problema de Hilbert para Subaneis de $\mathbb{Q}$

4.1 O 10^{o} Problema de Hilbert Original

Na viragem do sec XIX para o sec XX os fundamentos da Matemática estavam abalados e eram objeto de ampla discussão. Os paradoxos de Russel, de Cantor, de Richard contribuem decisivamente para a discussão. O Logicismo, o Formalismo e o Intuicionismo são as três linhas de pensamento que se destacam na tentativa de dar um firme fundamento à Matemática.
Hilbert, destacado matemático e representante do Formalismo, apresentou em 1900 uma lista de 23 problemas que na sua opinião deveriam orientar a investigação matemática nos próximos anos. Vamos debruçar-nos sobre o 10^{o} problema.
O 10^{o} problema de Hilbert, na sua formulação original, colocava a questão da existência de um algoritmo, para decidir se dada uma equação diofantina, existe ou não solução no conjunto dos números inteiros.
Formalizando o problema:

(1) Input: um polinómio $f(x_1,...,x_n)$ com coeficientes em $\mathbb{Z}$

(2) Output: Sim ou não, consoante

$$\exists \vec{a} = (a_1,...,a_n) \in \mathbb{Z}^n \text{ tal que } f(\vec{a}) = 0$$

Só em 1970, resultante dos trabalhos de Davis, Putnam, Robinson ([DPR61]) e mais tarde de Matijasevic ([Mat70]) , sobre uma conjetura de Davis, recorrendo às definições de conjuntos diofantinos, conjuntos recursivos, conjuntos listáveis, provaram o Teorema conhecido por DPRM.
Antes de enunciarmos o Teorema, vejamos as definições.
Começamos com a noção de equação diofantina,

DEFINIÇÃO 4.1.1. *Designamos por uma equação diofantina uma equação polinomial em 2 ou mais variáveis, cujos coeficientes estão em $\mathbb{Z}$ e em que só as soluções em $\mathbb{Z}$ são permitidas*

E agora com a noção dum conjunto recursivo,

DEFINIÇÃO 4.1.2. *Um subconjunto $S \subseteq \mathbb{Z}$ é recursivo se existe um algoritmo (máquina Turing) com input e output de acordo com o seguinte:*

(1) *Input: um inteiro n*

(2) *Output: sim ou não, consoante $n \in S$*

por fim temos a noção de conjunto listável,

DEFINIÇÃO 4.1.3. *Um subconjunto $S \subseteq \mathbb{Z}$ é listável se existe uma máquina de Turing tal que S é o conjunto dos inteiros que imprime quando deixada a correr para sempre.*

Relembramos agora alguns Factos sobre as noções acima:

FACTO 4.1.4. *Todo o subconjunto recursivo $S \subseteq \mathbb{Z}$ é listável [Coo04, capítulo 5]*

FACTO 4.1.5. *Existe um subconjunto listável que não é recursivo [Coo04, capítulo 5]*

É também necessário relembrar a noção de conjunto diofantino:

DEFINIÇÃO 4.1.6. *Um subconjunto $S \subseteq \mathbb{Z}$ é diofantino, se existe um polinómio*

$$p(\vec{t},\vec{x}) \in \mathbb{Z}[t_1,...,t_n,x_1,...,x_m]$$

tal que

$$S = \{\vec{a} \in \mathbb{Z}^n : (\exists \vec{x} \in \mathbb{Z}^m) p(\vec{a},\vec{x}) = 0\}$$

EXEMPLO 4.1.7. *O subconjunto dos números naturais $\mathbb{N}$:=0,1,2... de $\mathbb{Z}$ é diofantino dado que para $a \in \mathbb{Z}$*

$$a \in \mathbb{N} \Leftrightarrow (\exists x_1,x_2,x_3,x_4 \in \mathbb{Z}) x_1^2 + x_2^2 + x_3^2 + x_4^2 = a$$

Enunciemos então o Teorema DPRM: ([DPR61, Mat70])

TEOREMA 4.1.8.

Um subconjunto S de $\mathbb{Z}^n$ é diofantino se e só se for listável

É por esta via indireta, que a resposta negativa ao 10^o problema de Hilbert é obtida. Temos então o Corolário, consequência do Teorema anterior e do Facto 4.1.5:

COROLÁRIO 4.1.9. *Existe um conjunto diofantino não recursivo, ou seja uma equação polinomial dependendo dum parâmetro para o qual não existe algoritmo para decidir para que valores do parâmetro a equação tem solução inteira.*

4.2 Generalização do 10^o Problema de Hilbert a outros Anéis

Pode-se generalizar o problema, a outros anéis contendo $\mathbb{Z}$ que designaremos por R da seguinte forma:

(1) Input: um polinómio $f \in \mathbb{Z}[X_1,...,X_n]$

(2) Output: Sim ou não, consoante

$$\exists \vec{a} = (a_1,...,a_n) \in R^n \text{ e em que } f(\vec{a}) = 0$$

Se $S \subseteq R$ é outro sub-anel, poderemos considerar o 10^o problema de Hilbert sobre R mas com coeficientes em S em lugar de $\mathbb{Z}$.
Investigações sobre vários anéis foram efetuadas , por exemplo, sobre $\mathbb{R}$ e $\mathbb{C}$ em que a resposta ao problema de Hilbert é positiva. ([Tar51])

4.3 O 10^{o} Problema de Hilbert para Subanéis de $\mathbb{Q}$

4.3.1 Aproximação de Bjorn Poonen

À pergunta da existência dum algoritmo que possa responder se dada uma equação polinomial com coeficientes em $\mathbb{Q}$, tem soluções em $\mathbb{Q}$, a resposta no presente momento da investigação é : não se sabe.

Uma aproximação possível a este problema, teria como ponto de partida o Teorema de Matijasevic para $\mathbb{Z}$, mostrando que $\mathbb{Z}$ é diofantino sobre $\mathbb{Q}$. A demonstração da existência de um modelo diofantino de $\mathbb{Z}$ sobre $\mathbb{Q}$ implicaria uma reposta negativa ao 10^{o} problema sobre $\mathbb{Q}$.

Seguindo esta estratégia Poonen, no seu trabalho restringe o objeto de estudo, a sub-anéis entre $\mathbb{Z}$ e $\mathbb{Q}$:

$$\mathbb{Z}[S^{-1}]$$

onde S é um conjunto infinito de números primos.
Note-se que se $\mathcal{P} = 2,3,5...$ é o conjunto dos números primos então há uma bijeção entre

$$\text{subconjuntos de } \mathcal{P} \leftrightarrow \text{ subanéis de } \mathbb{Q}$$

$$S \longmapsto \mathbb{Z}[S^{-1}]$$

$$\mathcal{P} \cap R^{\times} \longleftarrow\!\shortmid R$$

Observemos que, para $S = \emptyset$ então $\mathbb{Z}[S^{-1}] = \mathbb{Z}$ e a resposta é negativa como foi referido anteriormente. Para $S = \mathcal{P}$ então $\mathbb{Z}[S^{-1}] = \mathbb{Q}$ a resposta é desconhecida.
Poonen, demonstra para $\mathbb{Z}[S^{-1}]$ exemplificando com infinitos subconjuntos S de $\mathcal{P}$ para os quais o 10^{o} problema de Hilbert sobre $\mathbb{Z}[S^{-1}]$ tem uma resposta negativa.
Antes da apresentação do Teorema, vejamos a definição de modelo diofantino:

DEFINIÇÃO 4.3.1. *Modelo diofantino de $\mathbb{Z}$ sobre $\mathbb{Q}$ é um conjunto $A \subseteq \mathbb{Q}^n$ que é diofantino sobre $\mathbb{Q}$ com a bijecção $\mathbb{Z} \to A$, na qual os grafos da adição e da multiplicação em $\mathbb{Z}$, são subconjuntos de $A^3 \subseteq \mathbb{Q}^3$ que são diofantinos sobre $\mathbb{Q}$.*

Em termos gerais, Poonen prova o Teorema:

TEOREMA 4.3.2. *Existem 2 conjuntos recursivos disjuntos de números primos T_1 e T_2, ambos de densidade natural 0, tal que para qualquer conjunto S de números primos, contendo T_1 e disjunto de T_2, verifica-se o seguinte:*

(1) *O conjunto dos números inteiros positivos com adição e multiplicação, $(\mathbb{Z}_{>0}, +, .)$ admite um modelo diofantino sobre $\mathbb{Z}[S^{-1}]$*

(2) *O 10^{o}problema de Hilbert sobre $\mathbb{Z}[S^{-1}]$ tem uma resposta negativa*

Deste modo, o objetivo é encontrar um anel $(A, +, \cdot)$ que admita um modelo diofantino sobre $\mathbb{Z}[S^{-1}]$. O 10^{o} problema de Hilbert sobre $\mathbb{Q}$ é equivalente ao problema da existência dum algoritmo para decidir se dada uma variedade algébrica sobre $\mathbb{Q}$ existe um ponto racional. Esta é a estratégia seguida por Poonen, em que o estudo assenta numa curva elíptica E sobre $\mathbb{Q}$.

4.3.2 Curva Elíptica Escolhida

A curva elíptica escolhida obedece a determinados requisitos necessários na argumentação desenvolvida na demonstração.
Para o estudo, foi escolhida uma curva elíptica E sobre $\mathbb{Q}$,

$$E: Y^2 = X^3 + aX + b \text{ onde } a, b \in \mathbb{Z}$$

tal que:

(1) $E(\mathbb{Q}) \cong \mathbb{Z}$
Deste modo, não existem em $E(\mathbb{Q})$ pontos com ordem finita

(2) $E(\mathbb{R})$ é conexo

(3) E não tem multiplicação complexa

4.3.3 Denominadores de Coordenadas-x

O gerador P de $E(\mathbb{Q})$ tem coordenadas racionais. A partir de P, obtemos $2P$, $3P$,... ou seja o conjunto $E(\mathbb{Q})$. Deste modo, podemos representar qualquer ponto de $E(\mathbb{Q})$ por nP onde $nP = \underbrace{P \oplus P \oplus \ldots \oplus P}_{n}$
Denotamos por $x(nP)$ a coordenada em x e $y(nP)$ a coordenada em y do ponto nP.

Vamos agora definir o conjunto finito de primos S_{bad}, em que temos a considerar 2 originadores de elementos:

(1) O número primo 2 pertence a S_{bad}.

(2) Os números primos da fatorização do denominador da coordenada x de P

Em baixo vamos precisar de um conjunto S_n e de números d_n

DEFINIÇÃO 4.3.3. *Para $n \in \mathbb{Z}$ definimos $d_n \in \mathbb{Z}_{>0}$ como o produto obtido da fatorização em primos do denominador da coordenada $x(nP)$ omitindo as potências dos primos em S_{bad}.*
Definimos ainda $d_0 = 0$.
O conjunto S_n é o conjunto dos fatores primos de d_n.

Vejamos os Lemas e Corolários demonstrados por Poonen descrevendo algumas das propriedades dos d_n e S_n:

LEMA 4.3.4.

(1) *Para qualquer $r \in \mathbb{Z}$ o conjunto $\{n \in \mathbb{Z} : r | d_n\}$ é um subgrupo de $\mathbb{Z}$*

(2) *Existe um $c \in \mathbb{R}_{>0}$ tal que $\log d_n = (c - o(1))n^2$ quando $n \to \infty$*

A demonstração deste Lema envolve os números p-ádicos pelo que sai fora do âmbito desta tese.

COROLÁRIO 4.3.5. *Se $m, n \in \mathbb{Z}$, e (m,n) representa o seu máximo divisor comum então temos que:*

$$S_{(m,n)} = S_m \cap S_n$$

em particular se $(m,n) = 1$, então $S_m \cap S_n = \emptyset$

Procedamos à demonstração deste Corolário:

Demonstração. Seja $q \in S_{(m,n)}$. Então $q|d_{(m,n)}$ logo $(m,n) \in \{k \in \mathbb{Z} : q|d_k\}$. Como $\{k \in \mathbb{Z} : q|d_k\}$ é um subgrupo de $\mathbb{Z}$ (Lema 4.3.4(1)), m é um múltiplo de (m,n) e n é um múltiplo de (m,n), temos que $m,n \in \{k \in \mathbb{Z} : q|d_k\}$. Ou seja $q|d_m$ e $q|d_n$, logo $q \in S_m$ e $q \in S_n$ ou seja $q \in S_m \cap S_n$.
Seja $q \in S_m \cap S_n$. Então $q|d_m$ e $q|d_n$. Logo $m,n \in \{k \in \mathbb{Z} : q|d_k\}$. Como $\{k \in \mathbb{Z} : q|d_k\}$ é um subgrupo de $\mathbb{Z}$ (Lema 4.3.4(1)) e $(m,n) = km + tn$ para algum $k,t \in \mathbb{Z}$, temos que $(m,n) \in \{k \in \mathbb{Z} : q|d_k\}$ ou seja, $q|d_{(m,n)}$. Logo, $q \in S_{(m,n)}$.
Finalmente, como $S_1 = \emptyset$, se $(m,n) = 1$, então S_m e S_n são disjuntos.

□

LEMA 4.3.6. *Se l e m são primos e* $\max\{l,m\}$ *é suficientemente grande, então:*

$$S_{lm} - (S_l \cup S_m) \neq \emptyset$$

Pelas mesmas razões que no Lema 4.3.4 a demonstração será omitida.

4.3.4 Conjunto T_1

Comecemos pela definição do conjunto T_1 :

DEFINIÇÃO 4.3.7.

$$T_1 = S_{bad} \cup \bigcup_{i \geq 1} S_{\ell_i}$$

Já vimos anteriormente a definição de S_{bad}. Para a definição de S_{ℓ_i} temos de caracterizar os elementos ℓ_i, o que nos leva previamente à caracterização do conjunto L

DEFINIÇÃO 4.3.8. *Para cada número primo* ℓ*, seja* a_ℓ *o mais pequeno* $a \in \mathbb{Z}_{>0}$ *tal que* $d_{\ell^a} > 1$

Pelo Lema 4.3.4(2), a_ℓ existe e $L = \{\ell : \ell \text{ é primo e } a_\ell \neq 1\}$ é um conjunto finito de números primos.

FACTO 4.3.9. *O conjunto L e os valores* a_ℓ *em que* ℓ *pertence a L são computáveis.*

Seja $\ell_1 < \ell_2 < \ell_3 <, \ldots..$ uma sequência de números primos cujas propriedades serão definidas adiante.

4.3.5 Conjunto T_2

Comecemos pela definição de conjunto T_2 :

DEFINIÇÃO 4.3.10.

$$T_2 = T_2^a \cup T_2^b \cup T_2^c$$

Para a construção dos subconjuntos T_2^a, T_2^b e T_2^c, necessitamos da seguinte definição:

DEFINIÇÃO 4.3.11. *Para cada número primo* ℓ*, seja* $p_\ell = \max S_{\ell^a}$*, onde* $a = a_\ell$

Consequentemente, considerando o Lema 4.3.6 e sendo m e ℓ números primos, temos $p_{\ell m} = \max\left(S_{\ell m} - (S_\ell \cup S_m)\right)$ quando $\max\{\ell,m\}$ é suficientemente grande.

Os subconjuntos T_2^a, T_2^b e T_2^c, são então definidos de acordo com o seguinte:

(1) $T_2^a = \{p_\ell : \ \ell \notin \{\ell_1, \ell_2, \ell_3, ...\}\}$

(2) $T_2^b = \{p_{\ell_i \ell_j} : \ 1 \leq j \leq i\}$ quando ℓ_1 é suficientemente grande.

(3) $T_2^c = \{p_{\ell \ell_i} : \ \ell \in L,\ i \geq 1\}$ quando ℓ_1 é suficientemente grande.

4.3.6 Propriedades T_1 e T_2

Vamos enunciar e demonstrar algumas das propriedades dos conjuntos T_1 e T_2.

PROPOSIÇÃO 4.3.12. *Os conjuntos T_1 e T_2 são disjuntos.*

Demonstração. Notemos que $T_1 = S_{bad} \cup \bigcup_{i\geq 1} S_{\ell_i}$.

(1) Justifiquemos primeiro que $S_{bad} \cap T_2 = \emptyset$.

Sendo o conjunto T_2 a união de 3 conjuntos, vamos proceder à análise separadamente:

(a) O conjunto T_2^a é o conjuntos do p_ℓ primos que por definição são o máximo do conjunto respetivo S_ℓ. Por definição no cálculo de S_ℓ não entram os números primos de S_{bad}, logo a interseção é vazia.

(b) O conjunto T_2^b é o conjuntos do $p_{\ell_i\ell_j}$ primos que por definição são o máximo do conjunto respetivo $\{S_{\ell_i\ell_j} - (S_{\ell i} \cup S_{\ell j})\}$. Por definição no cálculo de $\{S_{\ell_i\ell_j} - (S_{\ell i} \cup S_{\ell j})\}$ não entram os números primos de S_{bad}, logo a interseção é vazia.

(c) O conjunto T_2^c é o conjuntos do $p_{\ell\ell_i}$ primos que por definição são o máximo do conjunto respetivo $\{S_{\ell\ell_i} - (S_\ell \cup S_{\ell i})\}$. Por definição no cálculo de $\{S_{\ell\ell_i} - (S_\ell \cup S_{\ell i})\}$ não entram os números primos de S_{bad}, logo a interseção é vazia.

Logo está provado, que a interseção S_{bad} e T_2 é vazia.

(2) Justifiquemos agora que $(\bigcup_{i\geq 1} S_{\ell_i}) \cap T_2 = \emptyset$.

(a) Por definição do conjunto T_2^a, $\ell \neq \ell_i$. Sendo $a_\ell = 1$, temos então $(\ell^{a_\ell}, \ell_i) = 1$. Pelo [Poo03b, Corolário 3.3], $S_\ell \cap S_{\ell_i} = \emptyset$, logo por definição de p_ℓ, concluímos que $p_\ell \notin S_{\ell_i}$, donde $(\bigcup_{i\geq 1} S_{\ell_i}) \cap T_2^a = \emptyset$

(b) Se $i \in \{j,k\}$, por definição de $p_{\ell_j\ell_k} = \max\{S_{\ell_j\ell_k} - (S_{\ell_j} \cup S_{\ell_k})\}$, temos que $p_{\ell_j\ell_k} \notin S_{\ell_i}$.
Se $i \notin \{j,k\}$, então $(i,j) = (i,k) = 1$ e consequentemente $(i, jk) = 1$. Pelo [Poo03b, Corolário 3.3], temos $S_{\ell_i} \cap S_{\ell_j} = S_{\ell_i} \cap S_{\ell_k} = \emptyset$ e consequentemente temos $S_{\ell_i} \cap S_{\ell_j\ell_k} = \emptyset$, implica que $p_{\ell_j\ell_k} \notin S_{\ell i}$

(c) Se $i = j$, por definição de $p_{\ell_j\ell_k} = \max\{S_{\ell_j\ell_k} - (S_{\ell_j} \cup S_{\ell_k})\}$, temos que $p_{\ell_j\ell_k} \notin S_{\ell i}$.
Se $i \neq j$, então $(i,j) = 1$ e pelo [Poo03b, Corolário 3.3], temos $S_{\ell_i} \cap S_{\ell_j} = \emptyset$, implica que $p_{\ell\ell_i} \notin S_{\ell i}$.

Logo está provado, que a interseção $(\bigcup_{i\geq 1} S_{\ell_i})$ e T_2 é vazia.

Assim, concluímos que os conjuntos T_1 e T_2 são disjuntos. □

Seja E' a curva elíptica E no anel $\mathbb{Z}[S_{bad}^{-1}]$.

PROPOSIÇÃO 4.3.13. *Se S contém T_1 e é disjunta com T_2, então $E'(\mathbb{Z}[S^{-1}])$ é união de $\{\pm\ell_i P : i \geq 1\}$ e algum subconjunto do conjunto finito $\{rP : r | \prod_{\ell\in L} \ell^{a_\ell - 1}\}$*

Demonstração. (1) Porque a equação da curva elíptica E relaciona as coordenadas x e y temos que um ponto nP pertence a $E'(\mathbb{Z}[S^{-1}])$, se e só se $S_n \subset S$. Em particular temos, $S_{\ell_i} \subseteq T_1 \subseteq S$, donde podemos concluir que $\pm\ell_i P \in E'(\mathbb{Z}[S^{-1}])$.

(2) Qualquer ponto nP fora de

$$\{\pm \ell_i P : i \geq 1\} \cup \{rP : r | \prod_{\ell \in L} \ell^{a_{\ell}-1}\}$$

é para algum n divisivel por um dos seguintes:

(a) $\ell^{a_\ell} \notin \{\ell_1, \ell_1, ...,\}$ e então pelo Lema 4.3.4(1) em que o conjunto dos n é um subgrupo de $\mathbb{Z}$ em que por definição , $p_{\ell^{a_\ell}} \in T_2^a$, ou

(b) $\ell_i \ell_j$ para algum $1 \leq j \leq i$ e então pelo Lema 4.3.4(1) em que o conjunto dos n é um subgrupo de $\mathbb{Z}$ em que por definição , $p_{\ell_j \ell_k} \in T_2^b$, ou

(c) $\ell \ell_j$ para algum $\ell \in L$ com $i \geq 1$ e então pelo Lema 4.3.4(1) em que o conjunto dos n é um subgrupo de $\mathbb{Z}$ em que por definição , $p_{\ell \ell_i} \in T_2^c$.

Podemos assim concluir que para todos os casos $S_n \not\subseteq S$

□

4.3.7 Densidade Natural

A densidade natural de um subconjunto $T \subseteq \mathcal{P}$ é definida por:

DEFINIÇÃO 4.3.14.

$$\lim_{X \to \infty} \frac{\#\{p \in T : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$$

caso o limite exista.

4.3.8 Construção dos ℓ_i

Para cada primo ℓ, define-se:

$$\mu_\ell = \sup_{X \in \mathbb{Z}_{\geq 2}} \frac{\#\{p \in S_\ell : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$$

em que o supremum é atingido para algum $X \leq \max S_\ell$, logo μ_ℓ é computável para cada ℓ.

Temos o seguinte [Poo03b, Lema 7.1]

LEMA 4.3.15. *Para qualquer* $\varepsilon > 0$ *a densidade natural de* $\{\ell : \mu_\ell > \varepsilon\}$ *é* 0

Definimos ℓ_i indutivamente. Dada a sequência $\ell_1, ..., \ell_{i-1}$, seja ℓ_i o mais pequeno primo não pertencente a L, que satisfaça o conjunto das 5 condições seguintes:

(1) $\ell_i > \ell_j \qquad \forall j < i$

(2) $\mu_{\ell_i} \leq 2^{-i}$

(3) $p_{\ell_i l_j} > 2^i \qquad \forall j \leq i$

(4) $p_{\ell\ell_i} > 2^i \qquad \forall \ell \in L$

(5) $|y(\ell_i P) - i| \leq 1/(10i)$

FACTO 4.3.16. *A sequência $\ell_1, \ell_2, \ldots$ é bem definida e computável. [Poo03b, Proposição 7.2]*

4.3.9 Recursividade de T_1 e T_2

Um conjunto de números naturais é recursivo, se existir um algoritmo, que decida corretamente num tempo finito, se um dado número pertence ou não ao conjunto.
O conjunto $\{\ell_1, \ell_2, \ldots\}$ é recursivo, pois é estritamente uma sequência crescente em que os seus elementos podem ser computados ordenadamente.

PROPOSIÇÃO 4.3.17. *O conjunto T_1 é recursivo.*

Demonstração. O conjunto S_{bad} é um conjunto finito e todos os subconjuntos dos números naturais finitos são recursivos.
Precisamos então dum algoritmo para decidir quando um número primo p que não pertença a S_{bad} pertence a $\bigcup_{i\geq 1} S_{\ell_i}$. Tem-se que $p \in \bigcup_{i\geq 1} S_{\ell_i}$, se e só se, $p|d_{\ell_i}$ para algum i, o que advém diretamente da definição de S_{ℓ_i}.
Mas $p|d_{\ell_i}$, se e só se, se a ordem n_p da imagem do gerador P em $\mathbb{F}_p$ é divisor de ℓ_i, para algum i, mas ℓ_i é um número primo, logo temos que n_p pertence a $\{\ell_1, \ell_2, \ldots\}$, quando p pertence a S_{ℓ_i} . A ordem n_p pode ser computada.
Deste modo será suficiente verificar se $n_p \in \{\ell_1, \ell_2, \ldots\}$ □

Precisamos também do:

LEMA 4.3.18. *Se ℓ é um número primo, então $\ell | \#E(\mathbb{F}_{p_\ell})$*

Demonstração. Por definição de p_ℓ, o ponto $\ell^{a_\ell} P$ reduz para 0 em $E(\mathbb{F}_{p_\ell})$, mas para $\ell^{a_\ell - 1}$ isso não acontece. □

PROPOSIÇÃO 4.3.19. *O conjunto T_2^a é recursivo.*

Demonstração. Se p pertencer a T_2^a, por definição $p = p_\ell$ de algum $\ell \notin \{\ell_1, \ell_2, \ldots\}$ e consequentemente pelo [Poo03b, Lema 8.2], $p|\#E(\mathbb{F}_p)$.
O algoritmo para testarmos se um determinado número primo p pertence a T_2^a, verificamos que $p \notin S_{bad}$ e calculamos o cardinal de $E(\mathbb{F}_p)$, procedendo em seguida à decomposição em fatores primos do valor obtido para o cardinal. $p \in T_2^a$ se existir um fator primo $\ell \notin \{\ell_1, \ell_2, \ldots\}$ e em que $p_\ell = p$ □

PROPOSIÇÃO 4.3.20. *Os conjuntos T_2^b e T_2^c são recursivos.*

Demonstração. Se p pertencer a T_2^b, por definição $p = p_{\ell_i \ell_j}$.
Pela condição (3) na definição de ℓ_i, um número primo $p \in T_2^b$ tem de ser igual a $p_{\ell_i \ell_j}$ para algum $1 \leq j \leq i$ em que $2^i < p$.
Então para testarmos se um número primo p pertence a T_2^b é suficiente computar $p_{\ell_i \ell_j} < \log_2 p$ para $1 \leq j \leq i$.

Semelhantemente provamos e definimos o algoritmo para T_2^c Se p pertencer a T_2^c, por definição $p = p_{\ell\ell_i}$.
Pela condição (4) na definição de ℓ_i, um número primo $p \in T_2^c$ tem de ser igual a $p_{\ell\ell_i}$ para algum $\ell \in L$ em que $2^i < p$.
Então para testarmos se um número primo p pertence a T_2^c é suficiente computar $p_{\ell\ell_i} < \log_2 p$ para algum $\ell \in L$.

□

Os conjuntos T_1 e T_2 são recursivos.

4.3.10 Densidades de T_1 e T_2

Precisamos dos seguintes Factos:

FACTO 4.3.21. *O conjunto T_1 tem densidade natural 0. [Poo03b, Proposição 9.1]*

FACTO 4.3.22. *Os conjuntos T_2^b e T_2^c têm densidade natural 0. [Poo03b, Proposição 9.2]*

FACTO 4.3.23. *O conjunto T_2^a tem densidade natural 0. [Poo03b, Proposição 9.4]*

Os conjuntos T_1 e T_2 têm ambos densidade natural 0.

4.3.11 Prova do Teorema

Queremos demonstrar que para qualquer conjunto S contendo T_1 e disjunto de T_2, o conjunto dos números inteiros positivos com adição e multiplicação, $(\mathbb{Z}_{>0},+,\cdot)$ admite um modelo diofantino sobre $\mathbb{Z}[S^{-1}]$

Demonstração. Por [Shl07, Teorema 4.2] o conjunto de elementos não nulos de $\mathbb{Z}[S^{-1}]$ é diofantino.
É possível representarmos os elementos de $\mathbb{Q}$ como frações de elementos de $\mathbb{Z}[S^{-1}]$ em que o denominador seja diferente de zero. Consequentemente equaçõs sobre $\mathbb{Q}$, podem ser reescritas como sistemas de equações sobre $\mathbb{Z}[S^{-1}]$ e não há qualquer inconveniente em usá-las. Particularizando, podemos usar o predicado $x \geq y$, que representamos por $(\exists z_1,z_2,z_3,z_4 \in \mathbb{Q})(x = y+z_1^2+z_2^2+z_3^2+z_4^2)$

Vamos construir um modelo diofantino A dos números inteiros positivos, $\mathbb{Z}_{>0}$, sobre $\mathbb{Z}[S^{-1}]$.

Para $i \in \mathbb{Z}_{>0}$, seja $y_i := y(\ell_i P)$. Seja então o nosso $A = \{y_1,y_2,...\}$. Então temos que A é diofantino sobre $\mathbb{Z}[S^{-1}]$ porque consiste no conjunto dos elementos não negativos do conjunto das coordenadas y de $E'(\mathbb{Z}[S^{-1}])$ menos um conjunto finito. Temos deste modo uma bijeção $\mathbb{Z}_{>0} \to A$ aplicando i em y_i.

□

Temos igualmente de mostrar que os grafos da multiplicação e adição em $\mathbb{Z}_{>0}$ correspondem a subconjuntos diofantinos de A^3. Sabemos, que pela condição 5, para qualquer elemento ℓ_i, temos $|y_i - i| \leq 1/(10i)$, logo podemos concluir que no máximo temos $|y_i - i| \leq 1/10$

LEMA 4.3.24. *Seja $m,n,q \in \mathbb{Z}_{>0}$. Então:*

(1) *$m+n = q$ se e só se $|y_m + y_n - y_q| \leq 3/10$*

(2) *$m^2 = n$ se e só se $|y_m^2 - y_n| \leq 4/10$*

Demonstração.

(1) Deste modo a quantidade $y_m + y_n - y_q$ e o inteiro $m+n-q$ diferem quando muito $1/10 + 1/10 + 1/10$

(2) A quantidade $y_m^2 - y_n$ e o inteiro $m^2 - n$ diferem quando muito

$$|y_m^2 - m^2| + |y_n - n| \leq |(m + \frac{1}{10m})^2 - m^2| + \frac{1}{10} < \frac{4}{10}$$

□

Este Lema mostra-nos que os predicados $m+n=q$ e $m^2=n$ em $\mathbb{Z}_{>0}$ correspondem aos predicados diofantinos em A^3.

Está concluída a prova que A apresenta um modelo diofantino sobre $\mathbb{Z}_{>0}$.

Por aplicação do Teorema de Matijasevicc, conclui-se:

O 10^{o} problema de Hilbert sobre $\mathbb{Z}[S^{-1}]$ *tem uma resposta negativa*.

Capítulo 5

Teorema Poonen - Estudo Prático

5.1 Introdução

Foi escolhida a curva elíptica $E : Y^2 = X^3 + X + 1$ como base para o estudo prático e tendo como critério, o facto de haver correspondência com os requisitos considerados por Poonen B., na demonstração do 10 ºproblema de Hilbert sobre $\mathbb{Z}[S^{-1}]$.
Para começar, caracteriza-se a curva elíptica nomeadamente sobre o rank esperado (manteve-se o termo inglês para maior clareza), não singularidade, inexistência de pontos de ordem finita, ponto P gerador de $E(\mathbb{Q})$.
Em seguida, calculam-se: coordenadas em x e y, S_n, d_n, a_ℓ de vários pontos obtidos a partir do ponto gerador P.
Prossegue-se com o cálculo de $S_{(\ell,m)}$ e $S_{(\ell m)}$ para exemplificação do [Poo03a, Corolário 3.3] e do [Poo03a, Lema 3.4]. Por fim, constróiem-se: os conjuntos S_{bad}, L e determinam-se elementos dos conjuntos infinitos T_1 , T_2 e S, verificando as condições do Teorema de Poonen B.

5.2 Software Utilizado

Após análise ao software disponível, na área da matemática, "free open-source", tendo em consideração a área de incidência do estudo, selecionaram-se os seguintes produtos:

(1) Sage - Free Open Source Mathematics Software
`http://www.sagemath.org`

(2) GeoGebra
`http://www.geogebra.org`

GeoGebra

5.2.1 Sage - Free Open Source Mathematics Software

Sage foi a ferramenta principal utilizada no estudo prático. SageMathCloud e a linguagem Sage foram as opções escolhidas.
Foram utilizados 2 tipos de soluções: comandos e pequenas implementações de programação (transportadas para o texto quando utilizadas).

(1) Apresenta-se uma breve descrição de alguns comandos Sage utilizados no estudo.

```
# Comandos
E = EllipticCurve([1,1])          #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos
E                                 #visualiza as caracteristicas da curva elíptica E
P                                 #visualiza as coordenadas X,Y,Z do ponto P na curva elíptica E

P + P                             #Soma o Ponto P com P na curva elíptica E
n * P                             #Soma o Ponto P n vezes na curva elíptica E

E.rank()                          #visualiza o rank da curva elíptica E
E.has_cm()                        #visualiza (True/False) para a multiplicação complexa em E
E.cremonalabel()                  #visualiza a label da curva elíptica E na BD de Cremona
E.has_good_reduction_S[]          #visualiza (True/False) se a curva elíptica E tem boa redução
                                  #               fora da lista de números primos definida em S
E.gens()                          #visualiza pontos geradores de E
E.gens_certain()                  #visualiza (True/False) se há prova de que os geradores de E
                                  #                                          são corretos
E.S_integral_points(S=[2], mw_base=[P], verbose=false);  # visualiza os pontos da curva elipitica cujos denominadores
                                                         #          são fatorizados nos numeros primos definidos em S
```

Programa 5.1: Comandos Sage Utilizados

(2) Os programas em linguagem Sage estão estruturados de acordo com o seguinte standard:

```
# Definicoes

# Parametros                      #EDITAVEIS

# Variáveis

# Calculos iniciais

# Ciclo

# Calculos Finais

# OUTPUT
```

Programa 5.2: Estrutura Programas Sage

Todos os programas são diretamente executáveis numa célula Sage de SageMathCloud. Assinalaram-se como editáveis os campos que devem ser atualizados consoante os resultados que se pretendam obter. Todos os outros campos, instruções de programa, não devem ser objeto de qualquer alteração. As mensagens Sage são em inglês. As mensagens do programador são em português.

Alguns processamentos, por serem extremamente morosos (largas horas de processamento), foram calculados executando sucessivamente o mesmo programa, em que se alteravam sucessivamente os parâmetros de acordo com os resultados obtidos anteriormente.

5.2.2 GeoGebra - Geometry, Algebra

O GeoGebra foi utilizado para geração dos gráficos que ilustram o texto.

5.3 Caracterização da Curva Elíptica

Seja a curva elíptica E a seguinte:

$$E : Y^2 = X^3 + X + 1$$

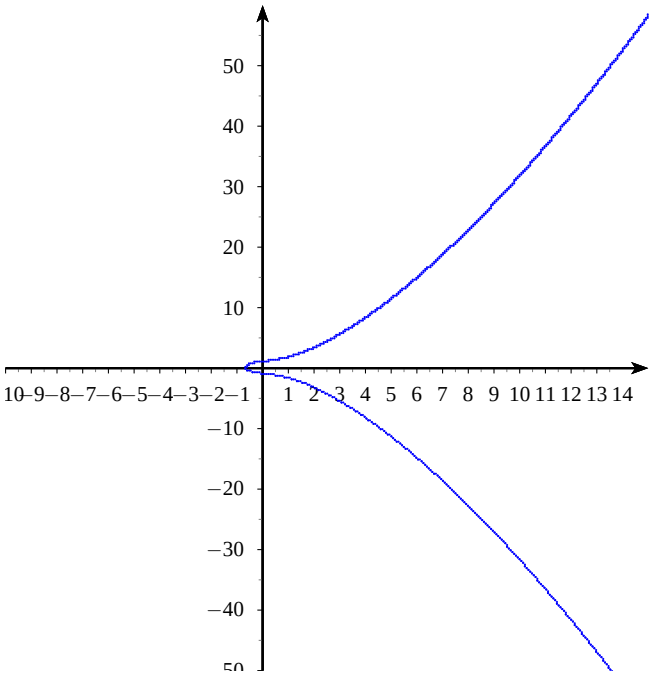

Figura 5.1: Curva Elíptica $y^2 = x^3 + x + 1$ de Rank 1

(1) E é uma curva de rank 1, sem multiplicação complexa.
Construímos a curva elíptica recorrendo ao software Sage, que nos garante que a curva escolhida tem rank 1, como pretendido:

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)

# Comando
rank =E.rank()
print ' O rank da curva elíptica E é:', rank

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
 O rank da curva elíptica E é: 1
```

Programa 5.3: Determinar Rank da Curva Elíptica E

Confirmamos que a curva elíptica não tem multiplicação complexa com igual recurso ao software Sage (em que cm: sinifica multiplicação complexa)

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)

# Comando
if    E.has_cm() == False:
      print ' A curva elíptica E não tem multiplicação complexa'
else:
      print ' A curva elíptica E tem multiplicação complexa'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
 A curva elíptica E não tem multiplicação complexa
```

Programa 5.4: Determinar Multiplicação Complexa na Curva Eliptica E

A identificação desta curva elíptica na base de dados Cremona, recorrendo ao software sage:

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)

# Comando
label = E.cremona_label()
print ' A label Cremona da curva elíptica E é:', label

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
 A label Cremona da curva elíptica E é: 496a1
```

Programa 5.5: Label Cremona da Curva Eliptica E

(2) E é uma curva não singular
Seja $\triangle_E$ o discriminante da equação de Weierstrass $Y^2 = X^3 + X + 1$

$$\triangle_E = -16(4A^3 + 27B^2)$$

Sendo $A = 1$ e $B = 1$ temos que o discriminante $\triangle_E = -2^4 \times 31 \neq 0$, logo a curva é não singular.

(3) Pontos de Ordem Finita em $E(\mathbb{Q})$
Pertencendo os coeficientes A e B a $\mathbb{Z}$ e de acordo com o Teorema de Nagell-Lutz para curvas elípticas do tipo em estudo, em que os coeficientes estão em $\mathbb{Z}$, temos que para qualquer $P(x,y) \in E_{tors}(\mathbb{Q})$, ambas as coordenadas x e y são inteiras e y igual a 0 ou y divide $\triangle_E$. Analisemos o caso em que $y = 0$ que implica $x^3 + x + 1 = 0$. Resolvendo esta equação, obtemos uma solução real e duas soluções complexas não inteiras.
Por outro lado, $\triangle_E = 496$ é divisivel pelos números primos $2, 31$. Atribuindo sucessivamente a y os inteiros que dividem o discriminante e resolvendo as equações resultantes da substituição de y em $x^3 + x + 1 = Y$, obtemos em todos os casos uma solução real e duas soluções complexas não inteiras.

Logo não existem soluções inteiras simultâneas para as coordenadas x, y de qualquer ponto P de $E(\mathbb{Q})$. Concluímos assim, que não existem pontos de ordem finita em $E(\mathbb{Q})$.

(4) Pontos de Ordem Infinita em $E(\mathbb{Q})$
Dado o rank da curva elíptica ser 1 e não existirem pontos de ordem finita $E(\mathbb{Q})$ como vimos anteriormente, isto significa que existe um ponto racional P, de ordem infinita, gerador de $E(\mathbb{Q})$.

Com recurso ao software sage, vamos identificar o ponto de ordem infinita P.

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1

# Comando
P = E.gens()
print ' O ponto',P, 'é o gerador de ordem infinita da curva elíptica E'
if   E.gens_certain() == True:
     print ' O ponto', P, 'é seguramente gerador de ordem infinita'
else:
     print ' O ponto', P, 'não é seguramente gerador de ordem infinita'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
 O ponto [(0 : 1 : 1)] é o gerador de ordem infinita da curva elíptica E
 O ponto [(0 : 1 : 1)] é seguramente gerador de ordem infinita
```

Programa 5.6: Ponto Gerador Ordem Infinita da Curva Eliptica E

O ponto P, em coordenadas afins, gerador de ordem infinita é $P(0,1)$. Tem-se então, pelo Teorema de Mordell-Weill:

$$E(\mathbb{Q}) \cong \mathbb{Z}$$

5.4 Cálculo de Pontos Racionais nP em E

Partindo do gerador $P(0,1)$, vamos calcular pontos racionais em $E(\mathbb{Q})$, com recurso à seguinte rotina em Sage:

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)

# Parametros
m = 7                             #número n                          EDITAVEIS

# Calculos iniciais

for k in range(1):
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        break

# Calcular nP e fatorizar  denominadores das coordenadas X,Y

    mP = m * P

    XmP = mP[0];
    DXmP = XmP.denominator();
    FDXmP = factor(DXmP);
    print 'A fatorização da coordenada X do número', m, 'P, é =', FDXmP;

    YmP = mP[1];
    DYmP = YmP.denominator();
    FDYmP = factor(DYmP);
    print 'A fatorização da coordenada Y do número', m, 'P, é =', FDYmP;

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
A fatorização da coordenada X do número 7P, é = 11^2 x 67^2 x 127^2
A fatorização da coordenada Y do número 7P, é = 11^3 x 67^3 x 127^3
```

Programa 5.7: Calcular nP e Fatorizar Denominadores das Coordenadas X,Y

Relembrando as fórmulas de cálculo, calculemos os valores:

$S_n =$ números primos do denominador da coordenada x não pertencentes a S_{bad}
$d_n =$ produto dos primos em S_n elevados às potências da fatorização
$F_{n_{xDen}} =$ Fatores Primos do denominador da coordenada x
$F_{n_{yDen}} =$ Fatores Primos do denominador da coordenada y
d_{ℓ^a}em que ℓ é um numero primo e a_ℓ o mais pequeno $a \in \mathbb{Z}_{>0}$ tal que $d_{\ell^a} > 1$
$nP = \underbrace{P \oplus P \oplus \ldots \oplus P}_{n}$ Cálculo de nP em que P é racional de E

Notas: alguns valores vêm assinalados como NC quando em virtude da sua dimensão não são incluíveis no texto, sem perda de clareza.

(1) Cálculo de $2P$
$2P = P + P$ onde P é o ponto em $E(\mathbb{Q})$ gerador de $E(\mathbb{Q})$

(a) $2P(\frac{1}{4}, \frac{-9}{8})$
$F_{2_{xDen}} = 2^2$
$F_{2_{yDen}} = 2^3$
$S_2 = \emptyset$
$d_2 = 1$
$\ell = 2, a_2 = 2, d_{2^2} > 1$

(2) Cálculo de $3P$
$3P = P + Q$ onde $Q = 2P$ que provém do cálculo anterior

(a) $3P(72,611)$
$F_{3_{xDen}} = 1$
$F_{3_{yDen}} = 1$

$S_3 = \emptyset$
$d_3 = 1$
$\ell = 3, a_3 = 2, d_{3^2} > 1$

(3) Cálculo de $4P$
$4P = P + Q$ onde $Q = 3P$ que provém do cálculo anterior

(a) $4P(\frac{-287}{1296}, \frac{40879}{46656})$
$F_{4_{xDen}} = 2^4 \times 3^4$

$F_{4_{yDen}} = 2^6 \times 3^6$
$S_4 = \{3\}$
$d_4 = 3^4$
$\ell = 4,$ não é número primo. Não aplicável $a_4, d_{4^{a_4}}$

(4) Cálculo de $5P$
$5P = P + Q$ onde $Q = 4P$ que provém do cálculo anterior

(a) $5P(\frac{43992}{82369}, \frac{-30699397}{23639903})$
$F_{5_{xDen}} = 7^2 \times 41^2$

$F_{5_{yDen}} = 7^3 \times 41^3$
$S_5 = \{7, 41\}$
$d_5 = 7^2 \times 41^2$
$\ell = 5, a_5 = 1, d_{5^1} > 1$

(5) Cálculo de $6P$
$6P = P + Q$ onde $Q = 5P$ que provém do cálculo anterior

(a) $6P(\frac{26862913}{1493284}, \frac{139455877527}{1824793048})$
$F_{6_{xDen}} = 2^2 \times 13^2 \times 47^2$

$F_{6_{yDen}} = 2^3 \times 13^3 \times 47^3$
$S_6 = \{13, 47\}$
$d_6 = 13^2 \times 47^2$
$\ell = 6,$ não é número primo. Não aplicável $a_6, d_{6^{a_6}}$

(6) Cálculo de $7P$
$7P = P + Q$ onde $Q = 6P$ que provém do cálculo anterior

(a) $7P(\frac{-3596697936}{8760772801}, \frac{591456591665497}{819999573400799})$
$F_{7_{xDen}} = 11^2 \times 67^2 \times 127^2$

$F_{7_{yDen}} = 11^3 \times 67^3 \times 127^3$
$S_7 = \{11, 67, 127\}$
$d_7 = 11^2 \times 67^2 \times 127^2$
$\ell = 7, a_7 = 1, d_{7^1} > 1$

(7) Cálculo de $8P$
$8P = P + Q$ onde $Q = 7P$ que provém do cálculo anterior

(a) $8P(\frac{7549090222465}{8662944250944}, \frac{-405816637344074398 07}{254975398584724638 72})$
$F_{8_{xDen}} = 2^6 \times 3^4 \times 40879^2$

$F_{8_{yDen}} = 2^9 \times 3^6 \times 40879^3$
$S_8 = \{3, 40879\}$
$d_8 = 3^4 \times 40879^2$
$\ell = 8,$ não é número primo. Não aplicável $a_8, d_{8^{a_8}}$

(8) Cálculo de $9P$
$9P = P + Q$ onde $Q = 8P$ que provém do cálculo anterior

(a) $9P(\frac{51865013741670864}{6504992707996225}, \frac{11915777535633550432194263}{52465065704911831290 5375})$
$F_{9_{xDen}} = 5^2 \times 503^2 \times 32069^2$

$F_{9_{yDen}} = 5^3 \times 503^3 \times 32069^3$
$S_9 = \{5, 503, 32069\}$
$d_9 = 5^2 \times 503^2 \times 32069^2$
$\ell = 9,$ não é número primo. Não aplicável $a_9, d_{9^{a_9}}$

(9) Cálculo de $10P$
$10P = P + Q$ onde $Q = 9P$ que provém do cálculo anterior

(a) $10P(\frac{-173161424238594532415}{3105156367744812388 84}, \frac{54717369718193218383707061921 52}{28375100785435021837118308918 17})$
$F_{10_{xDen}} = 2^2 \times 7^2 \times 41^2 \times 30699397^2$

$F_{10_{yDen}} = 2^3 \times 7^3 \times 41^3 \times 30699397^3$
$S_{10} = \{7, 41, 30699397\}$
$d_{10} = 7^2 \times 41^2 \times 30699397^2$
$\ell = 10,$ não é número primo. Não aplicável $a_{10}, d_{10^{a_{10}}}$

(10) Cálculo de $11P$
$11P = P + Q$ onde $Q = 10P$ que provém do cálculo anterior

(a) $11P(\frac{600592302706906735608 1464}{4609517672092049172106561}, \frac{-21028397961497210092713450922431210037}{98965364029292703333025012426400998 09})$
$F_{11_{xDen}} = 419^2 \times 5124054251^2$

$F_{11_{yDen}} = 419^3 \times 5124054251^3$
$S_{11} = \{419, 5124054251\}$
$d_{11} = 419^2 \times 5124054251^2$
$\ell = 11, a_{11} = 1, d_{11^1} > 1$

(11) Cálculo de $12P$
$12P = P + Q$ onde $Q = 11P$ que provém do cálculo anterior

(a) $12P(\frac{516800901506579137034097949153}{1161652011530610982610237761 44}, \frac{38284499813337506892512075721659350884149473 7}{39592604638617085219380314122331748004030272})$
$F_{12_{xDen}} = 2^4 \times 3^6 \times 13^2 \times 29^2 \times 37^2 \times 47^2 \times 1721^2 \times 2797^2$

$F_{12_{yDen}} = 2^6 \times 3^9 \times 13^3 \times 29^3 \times 37^3 \times 47^3 \times 1721^3 \times 2797^3$
$S_{12} = \{3, 13, 29, 37, 47, 1721, 2797\}$
$d_{12} = 3^6 \times 13^2 \times 29^2 \times 37^2 \times 47^2 \times 1721^2 \times 2797^2$
$\ell = 12,$ não é número primo. Não aplicável $a_{12}, d_{12^{a_{12}}}$

(12) Cálculo de $13P$
$13P = P + Q$ onde $Q = 12P$ que provém do cálculo anterior

(a) $13P(\frac{-37736542366253475570818485617967128}{5794167433504947998768876827435 2769}, \frac{37543086291884747339175213680482424476128042698485 57}{13947184989042686635273853115866192596128419723708353})$
$F_{13_{xDen}} = 240710769046691137^2$

$F_{13_{yDen}} = 240710769046691137^3$
$S_{13} = \{240710769046691137\}$
$d_{13} = 240710769046691137^2$
$\ell = 13, a_{13} = 1, d_{13^1} > 1$

(13) Cálculo de $14P$
$14P = P + Q$ onde $Q = 13P$ que provém do cálculo anterior

(a) $14P(\frac{23419679382776533016338728874246651427713}{1225880569761762989368984702749518724883 6}, \frac{-42669615440899599235171408697613606401015056927434810991862 49}{13572887276793561355003160038054885146409318349704420420794 16})$

$F_{14_{xDen}} = 2^2 \times 11^2 \times 67^2 \times 127^2 \times 591456591665497^2$

$F_{14_{yDen}} = 2^3 \times 11^3 \times 67^3 \times 127^3 \times 591456591665497^3$
$S_{14} = \{11, 67, 127, 591456591665497\}$
$d_{14} = 11^2 \times 67^2 \times 127^2 \times 591456591665497^2$
$\ell = 14,$ não é número primo. Não aplicável $a_{14}, d_{14^{a_{14}}}$

(14) Cálculo de $15P$
$15P = P + Q$ onde $Q = 14P$ que provém do cálculo anterior

(a) $15P(\frac{26449452347718826171173662182327682047670541792}{946609480458638576231250966183730296135455040 1},$
$\frac{46606458136711217650255902676473006722529458735865410777113893945637 91}{9209928837349924627451415221112259088619760982194656165856924 5395649})$

$F_{15_{xDen}} = 7^2 \times 41^2 \times 36097^2 \times 79588361^2 \times 118000231^2$

$F_{15_{yDen}} = 7^3 \times 41^3 \times 36097^3 \times 79588361^3 \times 118000231^3$
$S_{15} = \{7, 41, 36097, 79588361, 118000231\}$
$d_{15} = 7^2 \times 41^2 \times 36097^2 \times 79588361^2 \times 118000231^2$
$\ell = 15,$ não é número primo. Não aplicável $a_{15}, d_{15^{a_{15}}}$

(15) Cálculo de $16P$
$16P = P + Q$ onde $Q = 15P$ que provém do cálculo anterior

(a) $16P(\frac{-389367048138495499968455411084106749368804088837145 59}{57067021596582095418792572856507183982778272745812224},$
$\frac{-11555135799560864467265052203092115153188617871843133706685139551808316060288 1}{13632572314749827052343190259228808419404589327309380914502004773796813685690368})$

$F_{16_{xDen}} = 2^8 \times 3^4 \times 7937^2 \times 40879^2 \times 5112972626232511^2$

$F_{16_{yDen}} = 2^{12} \times 3^6 \times 7937^3 \times 40879^3 \times 5112972626232511^3$
$S_{16} = \{3, 7937, 40879, 5112972626232511\}$
$d_{16} = 3^4 \times 7937^2 \times 40879^2 \times 5112972626232511^2$
$\ell = 16,$ não é número primo. Não aplicável $a_{16}, d_{16^{a_{16}}}$

(16) Cálculo de $17P$
$17P = P + Q$ onde $Q = 16P$ que provém do cálculo anterior

(a) $17P(\frac{45916459412056517587433993766785476404391103664725829768912 0}{16015759540315507981058922649735677225876502666521790971820 81},$
$\frac{-33569710015123825744650243724044111391688692120988273919130607180461348556517452282992968 1}{64094580522113759156198705195016223582644524244776037761659451423230368974380292474542271})$

$F_{17_{xDen}} = 4091^2 \times 82837^2 \times 3017341^2 \times 391377210997453^2$

$F_{17_{yDen}} = 4091^3 \times 82837^3 \times 3017341^3 \times 391377210997453^3$
$S_{17} = \{4091, 82837, 3017341, 391377210997453\}$
$d_{17} = 4091^2 \times 82837^2 \times 3017341^2 \times 391377210997453^2$
$\ell = 17, a_{17} = 1, d_{17^1} > 1$

(17) Cálculo de $18P$
$18P = P + Q$ onde $Q = 17P$ que provém do cálculo anterior

(a) $18P(\frac{68959182620311388778577311972229580726760149557647155768372919686 41}{36944651848893877713225624066917526092297582124077894864378991481 00},$
$\frac{21736496197915859010987602922106689703210698856809890238130753199198514730291506877535280890728096311}{71011284323853676450640847863735161270204834339794395336946362455747271476655801197441561533566210 00})$

$F_{18_{xDen}} = 2^2 \times 5^2 \times 13^2 \times 17^4 \times 47^2 \times 503^2 \times 18397^2 \times 32069^2 \times 11165779^2 \times 328509019^2$

$F_{18_{yDen}} = 2^3 \times 5^3 \times 13^3 \times 17^6 \times 47^3 \times 503^3 \times 18397^3 \times 32069^3 \times 11165779^3 \times 328509019^3$
$S_{18} = \{5, 13, 17, 47, 503, 18397, 32069, 11165779, 328509019\}$
$d_{18} = 5^2 \times 13^2 \times 17^4 \times 47^2 \times 503^2 \times 18397^2 \times 32069^2 \times 11165779^2 \times 328509019^2$
$\ell = 18,$ não é número primo. Não aplicável $a_{18}, d_{18^{a_{18}}}$

(18) Cálculo de $19P$
$19P = P + Q$ onde $Q = 18P$ que provém do cálculo anterior

(a) $19P(\frac{-19221396395849680820602265032469014282778956866659318660297908 1002044415320}{29691809843242540752577327698559664154833604461531476154166114523121734 0801},$
$\frac{-14591699049790836092310699877248493069596857022893742468160697733610682142448060174546620541373741443498017102 59}{51162882649472895616919501911148616373782637016506809619752832842681444643648133790878492944764402652816724699 51})$

$F_{19_{xDen}} = 1409^2 \times 14401^2 \times 84920920247437898904712836223 9^2$

$F_{19_{yDen}} = 1409^3 \times 14401^3 \times 84920920247437898904712836223 9^3$
$S_{19} = \{1409, 14401, 84920920247437898904712836223 9\}$
$d_{19} = 1409^2 \times 14401^2 \times 84920920247437898904712836223 9^2$
$\ell = 19, a_{19} = 2, d_{19^1} > 1$

(19) Cálculo de $21P$
$21P = P + Q$ onde $Q = 20P$

(a) $21P(\frac{NC}{NC}, \frac{NC}{NC})$
$F_{21_{xDen}} = 11^2 \times 19^2 \times 67^2 \times 101^2 \times 107^2 \times 127^2 \times 139^2 \times 996888466040485726666015170445207^2$

$F_{21_{yDen}} = 11^3 \times 19^3 \times 67^3 \times 101^3 \times 107^3 \times 127^3 \times 139^3 \times 996888466040485726666015170445207^3$
$S_{21} = \{11, 19, 67, 101, 107, 127, 139, 996888466040485726666015170445207\}$
$d_{21} = 11^2 \times 19^2 \times 67^2 \times 101^2 \times 107^2 \times 127^2 \times 139^2 \times 996888466040485726666015170445207^2$
$\ell = 21,$ não é número primo. Não aplicável $a_{21}, d_{21^{a_{21}}}$

(20) Cálculo de $22P$
$22P = P + Q$ onde $Q = 21P$ que provém do cálculo anterior

(a) $_{22P(\frac{NC}{NC}, \frac{NC}{NC})}$
$F_{22_{xDen}} = 2^2 \times 283^2 \times 419^2 \times 9337^2 \times 50461^2 \times 5124054251^2 \times 157709022629640714785102227^2$

$F_{22_{yDen}} = NC$
$S_{22} = \{283, 419, 9337, 50461, 5124054251, 157709022629640714785102227\}$
$d_{22} = 283^2 \times 419^2 \times 9337^2 \times 50461^2 \times 5124054251^2 \times 157709022629640714785102227^2$
$\ell = 22,$ não é número primo. Não aplicável $a_{22}, d_{22^{a_{22}}}$

(21) Cálculo de $23P$
$23P = P + Q$ onde $Q = 22P$ que provém do cálculo anterior

(a) $_{23P(\frac{NC}{NC}, \frac{NC}{NC})}$
$F_{23_{xDen}} = 5824307669^2 \times 2892229819630428638773690149186718411931792932$

$F_{23_{yDen}} = NC$
$S_{23} = \{5824307669, 2892229819630428638773690149186718411931792932\}$
$d_{23} = 5824307669^2 \times 2892229819630428638773690149186718411931792932$
$\ell = 23, a_{23} = 1, d_{23^1} > 1$

(22) Cálculo de $25P$
$25P = P + Q$ onde $Q = 24P$

(a) $_{25P(\frac{NC}{NC}, \frac{NC}{NC})}$
$F_{25_{xDen}} = 7^2 \times 41^2 \times 33224363^2 \times 18875330097677^2 \times 190728312224797197710686021440375760978097^2$

$F_{25_{yDen}} = 7^3 \times 41^3 \times 33224363^3 \times 18875330097677^3 \times 190728312224797197710686021440375760978097^3$
$S_{25} = \{7, 41, 33224363, 18875330097677, 190728312224797197710686021440375760978097\}$
$d_{25} = 7^2 \times 41^2 \times 33224363^2 \times 18875330097677^2 \times 190728312224797197710686021440375760978097^2$
$\ell = 25,$ não é número primo. Não aplicável $a_{25}, d_{25^{a_{25}}}$

(23) Cálculo de $27P$
$27P = P + Q$ onde $Q = 26P$

(a) $_{27P(\frac{NC}{NC}, \frac{NC}{NC})}$
$F_{27_{xDen}} = 5^2 \times 359^2 \times 503^2 \times 32069^2 \times 17540359^2 \times$

$3582736852718269213724587157244438482974165253388434762383^2$
$F_{27_{yDen}} = NC$
$S_{27} = \{5, 359, 503, 32069, 17540359, 3582736852718269213724587157244438482974165253388434762383\}$
$d_{27} = 5^2 \times 359^2 \times 503^2 \times 32069^2 \times 17540359^2 \times 3582736852718269213724587157244438482974165253388434762383^2$
$\ell = 27,$ não é número primo. Não aplicável $a_{27}, d_{27^{a_{27}}}$

(24) Cálculo de $33P$
$33P = P + Q$ onde $Q = 32P$

(a) $_{33P(\frac{NC}{NC}, \frac{NC}{NC})}$
$F_{33_{xDen}} = 419^2 \times 5124054251^2 \times 112495827113005309^2 \times$

$1343091216935291951449340782628927934643571115593362464068505478175175485161466647792$
$F_{33_{yDen}} = NC$
$S_{33} = \{419, 5124054251, 112495827113005309$
$, 1343091216935291951449340782628927934643571115593362464068505478175175485161466647792\}$
$d_{33} = 419^2 \times 5124054251^2 \times 112495827113005309^2 \times$
$1343091216935291951449340782628927934643571115593362464068505478175175485161466647792^2$
$\ell = 33$ não é número primo. Não aplicável $a_{33}, d_{33^{a_{33}}}$

5.5 Height $H(X(nP))$

Para a curva elíptica em estudo e para o ponto $P(0,1)$, construímos a tabela de $H(X(nP))$ para os primeiros 22 pontos. Note-se o crescimento do número de dígitos à medida que n aumenta [Sil08, Lema 4.1 (b)].

```
# Definicoes
E = EllipticCurve([1,1])          #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]);                     #definir P como o ponto infinito gerador de E(Q)

# Parametros
m = 1                             #número m inicial                EDITAVEIS
ntimes = 22                       #número vezes do ciclo           EDITAVEIS


# Calcular Height

for i in range(ntimes):
    mP = m * P
    XmP = mP[0] ;
    A=XmP.height()
    if   m < 10:
         print m,'   ', A
    else:
         print m,'  ', A
    m=m+1
# OUTPUT
1P     1
2P     4
3P     72
4P     1296
5P     82369
6P     26862913
7P     8760772801
8P     8662944250944
9P     51865013741670864
10P    310515636774481238884
11P    60059230270690673560814 64
12P    516800901506579137034097949153
13P    57941674335049479987688768274352769
14P    234196793827765330163387288742466514277 13
15P    26449452347718826171173662182327682047670541792
16P    57067021596582095418792572856507183982778272745812224
17P    459164594120565175874339937667854764043911036647258297689120
18P    689591826203113887785773119722295807267601495576471557683729196864 1
19P    29691809843242540752577327698559664154833604461531476154166114523121734080 1
20P    458891135067460617559393347748448737372098017478945476634496877965445549330329886 41
21P    902969479206730073819627845037793082263297789593664611266477126082385875207979572674179391 2
22P    815319539494848460179709256156552339863652654432705447903035660624694065590758094666477190882328803616 82
```

Programa 5.8: Height $H(X(nP))$ de Pontos da Curva Eliptica E referidos a P

5.6 Cálculo de $S_{(\ell,m)}$

Relembramos que representamos o máximo divisor comum dos inteiros ℓ e m por (l,m).
Vamos calcular os valores de S_ℓ referente ao ponto ℓP, de S_m referente ao ponto mP , de $S_{(\ell,m)}$ referente ao ponto $(\ell,m)P$ para exemplificação do [Poo03a, Corolário 3.3]:

$$S_{(\ell,m)} = S_\ell \cap S_m \text{ em que, se mdc } (\ell,m) = 1 \text{ então } S_\ell \cap S_m = \emptyset$$

De acordo com os cálculos anteriores (Ver em 5.4 na página 37):

(1) Seja $\ell = 4$ e $m = 8$.

$S_4 = \{3\}$
$S_8 = \{3, 40879\}$
$S_{(4,8)} = S_4 = S_4 \cap S_8 = \{3\}$

(2) Seja $\ell = 4$ e $m = 12$.

$S_4 = \{3\}$
$S_{12} = \{3, 13, 29, 37, 47, 1721, 2797\}$
$S_{(4,12)} = S_4 = S_4 \cap S_{12} = \{3\}$

(3) Seja $\ell = 9$ e $m = 18$.

$S_9 = \{5, 503, 32069\}$
$S_{18} = \{5, 13, 17, 47, 503, 18397, 32069, 11165779, 328509019\}$
$S_{(9,18)} = S_9 = S_9 \cap S_{18} = \{5, 503, 32069\}$

(4) Seja $\ell = 9$ e $m = 27$.

$S_9 = \{5, 503, 32069\}$
$S_{27} = \{5, 359, 503, 32069,$
$17540359, 358273685271826921372458715724443848297416525338843476238 3\}$
$S_{(9,27)} = S_9 = S_9 \cap S_{27} = \{5, 503, 32069\}$

(5) Seja $\ell = 14$ e $m = 21$.

$S_{14} = \{11, 67, 127, 591456591665497\}$
$S_{21} = \{11, 19, 67, 101, 107, 127, 139, 996888466040485726666015170445207\}$
$S_{(14,21)} = S_7 = S_{14} \cap S_{21} = \{11, 67, 127\}$

(6) Seja $\ell = 14$ e $m = 15$.

$S_{14} = \{11, 67, 127, 591456591665497\}$
$S_{15} = \{7, 41, 36097, 79588361, 118000231\}$
$S_{14,15} = S_1 = S_{14} \cap S_{15} = \emptyset$

5.7 Cálculo de $S_{\ell m}$

Vamos calcular os valores de S_ℓ referente ao ponto ℓP, de S_m referente ao ponto mP , de $S_{\ell m}$ referente ao ponto ℓmP para exemplificação do [Poo03a, Lema 3.4]:

$$S_{\ell m} - (S_\ell \cup S_m) \neq \emptyset \qquad \text{em que } \ell \text{ é suficientemente grande}$$

De acordo com os cálculos anteriores (Ver em 5.4 na página 37):

(1) Seja $\ell = 2$, $m = 3$ e $\ell m = 6$

$S_2 = \emptyset$
$S_3 = \emptyset$
$S_{2\times 3} = S_6 = \{13, 47\}$
$S_6 - (S_2 \cup S_3) = \{13, 47\}$

(2) Seja $\ell = 2$, $m = 5$ e $\ell m = 10$

$S_2 = \emptyset$
$S_5 = \{7, 41\}$
$S_{2\times 5} = S_{10} = \{7, 41, 30699397\}$
$S_{10} - (S_2 \cup S_5) = \{30699397\}$

(3) Seja $\ell = 2$, $m = 7$ e $\ell m = 14$

$S_2 = \emptyset$
$S_7 = \{11, 67, 127\}$
$S_{2\times 7} = S_{14} = \{11, 67, 127, 591456591665497\}$
$S_{14} - (S_2 \cup S_7) = \{591456591665497\}$

(4) Seja $\ell = 3$, $m = 5$ e $\ell m = 15$

$S_3 = \emptyset$
$S_5 = \{7, 41\}$
$S_{3\times 5} = S_{15} = \{7, 41, 36097, 79588361, 118000231\}$
$S_{15} - (S_3 \cup S_5) = \{36097, 79588361, 118000231\}$

(5) Seja $\ell = 5$, $m = 5$ e $\ell m = 25$

$S_5 = \{7, 41\}$
$S_{5\times 5} = S_{25} = \{7, 41, 33224363, 18875330097677,$
$190728312224797197710686021440375760978097\}$
$S_{25} - (S_5 \cup S_5) = \{33224363, 18875330097677,$
$190728312224797197710686021440375760978097\}$

5.8 Conjunto S_{bad}

Para determinar o conjunto finito de primos S_{bad}, temos a considerar 2 originadores de elementos:

(1) Número Primo 2

Por definição o número primo 2 pertence a S_{bad}.

(2) Números primos da fatorização do denominador da coordenada *x* de *P*

Os primos da fatorização do denominador da coordenada *x* do gerador *P* pertecem a S_{bad}. Neste caso, dado a coordenada *x* ser um número inteiro, não há elementos a incluir em S_{bad}.

Podemos então definir:

$$S_{bad} = \{2\}$$

5.9 Conjunto L

Por definição o conjunto L, é o conjunto finito dos números primos ℓ em que $a_\ell > 1$, sendo que $a \in \mathbb{Z}_{>0}$ tal que $d_{\ell^a} > 1$
Vamos então pesquisar os numeros primos ℓ, em que $d_\ell = 1$, tomando em consideração os números primos que estão em S_{bad}. Para o conseguirmos recorremos aos comandos Sage.

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)

# Comandos

L = E.S_integral_points(S=[2], mw_base=[P], verbose=false);

print 'os pontos que pertencem ao conjunto L, são:', L

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
os pontos que pertencem ao conjunto L, são: [(0 : 1 : 1), (1/4 : 9/8 : 1), (72 : 611 : 1)]
```

Programa 5.9: Determinar Elementos do Conjunto L

São então 3 os pontos: P, $2P$ e $3P$. Então:

$$L = \{2,3\}$$

5.10 Conjunto de Números Primos ℓ_i

Relembremos o conjunto de condições a que os números primos ℓ, que representaremos por ℓ_i têm de cumprir para integrarem este conjunto:

(1) Condição 1: $\ell_i > \ell_j \qquad \forall j < i$

(2) Condição 2: $\mu_{\ell_i} \leq 2^{-i}$

(3) Condição 3: $p_{\ell_i \ell_j} > 2^i \qquad \forall j \leq i$

(4) Condição 4: $p_{\ell \ell_i} > 2^i \qquad \forall \ell \in L$

(5) Condição 5: $|y(\ell_i P) - i| \leq 1/(10i)$

(6) Condição 6: $\ell_i \notin L$

Tenhamos igualmente presente as definições:

$$p_\ell = \max\, S_{\ell^a} \qquad \text{onde } a = a_\ell$$

$$\mu_{\ell_i} = \sup_{X \in \mathbb{Z}_{\geq 2}} \frac{\#\{p \in S_{\ell_i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$$

5.10.1 Estratégia para Construção e Validação de Elementos ℓ_i

Constatamos que:

(1) Sejam mP os pontos da curva elíptica em estudo, em que P é o ponto gerador de ordem infinita.
À medida que m cresce, as coordenadas x e y dos pontos mP, apresentam denominadores com um grande número de dígitos. Para valores de m elevados não foi possível calcular as coordenadas de mP.

(2) Com os meios tecnológicos atuais os algoritmos de fatorização em números primos não resolvem em tempo útil o problema para números com elevado número de dígitos.

Dadas as dificuldades resultantes dos factos referidos, seguiram-se as seguintes estratégias na implementação do estudo prático deste conjunto.

- 1^o - Construção do Conjunto de Candidatos ao conjunto ℓ_i
- 2^o - Validação das 6 condições requeridas para aceitação do candidato como elemento do conjunto ℓ_i

(1) Construção do Conjunto de Candidatos ao conjunto ℓ_i

   (a) Pesquisa sequencial do conjunto dos números primos, a partir de 2.
   (b) Para cada número primo, definir se é ou não candidato, em função da condição 5

(2) Validação das 6 condições requeridas para aceitação do candidato como elemento do conjunto ℓ_i

Analisemos as estratégias para validação dos candidatos ao conjunto ℓ_i, condição a condição:

(a) Condição 1: $\ell_i > \ell_j \qquad \forall j < i$

A construção de candidatos ao conjunto ℓ_i, por ordem crescente, garante esta condição.

(b) Condição 2: $\mu_{\ell_i} \leq 2^{-i}$, em que:

$$\mu_{\ell_i} = \sup_{X \in \mathbb{Z}_{\geq 2}} \frac{\#\{p \in S_{\ell_i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$$

A impossibilidade de fatorizar totalmente ou até fatorizar parcialmente o denominador da coordenada X dos pontos $\ell_i P$, levaram à escolha duma solução alternativa, com utilização de 2 métodos cumulativos.

Seja:
Nf = raiz denominador coordenada $X(\ell_i P)$ ou o resultado das divisões por f_i
Ndnf = número de dígitos de Nf
f_i = fatores primos resultantes da fatorização
Nfi = total de fatores primos efetivos resultantes da fatorização

Método 1:
Sempre que resultante da fatorização, tenham sido identificados fatores primos. Permite o cálculo efetivo de $\frac{\#\{p \in S_{\ell i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$, para cada fator primo de Nf encontrado.

- Fatorizar Nf por pesquisa sequencial crescente de números primos a partir de 2.
- Por cada fator primo encontrado, calcular $\frac{\#\{p \in S_{\ell i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$ em que $X =$ fator primo e recalcular Nf.

Método 2:

Permite-nos calcular o número primo X a partir do qual, $\frac{\#\{p \in S_{\ell i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$ é sempre $\leq 2^{-i}$. Para conseguir este objetivo, estabelecemos a regra que os sucessores imediatos de X são fatores de Nf e em número estimado pela fórmula de Número Máximo Fatores

(Max) que nos permite calcular o Maior Valor Estimado (MVE). Neste método, trabalhamos em resultados hipotéticos máximos, em que pretendemos garantir a verdade da condição 2 e não calcular valores efetivos.
Comecemos por definir

DEFINIÇÃO 5.10.1. *Menor Número Primo, que designamos por Lim, é o primeiro número primo estimado, que na pesquisa sequencial crescente de números primos para fatorizar um dado número, torna verdadeira a expressão:*

$$\frac{\#\{p \in S_{\ell i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}} \leq 2^{-i} \qquad \forall X \geq Lim$$

Designamos por Ndmfp o número de dígitos de Lim

DEFINIÇÃO 5.10.2. *Número Máximo Fatores, que designaremos por Max, é a quantidade máxima estimada de fatores primos que poderão existir num dado número, baseados na quantidade de dígitos resultantes num produto.*

$$Max = \frac{Nfi + Ndnf}{Ndmfp - 1}$$

DEFINIÇÃO 5.10.3. *Maior Valor Estimado, que designamos por MVE, é o maior valor possível obtido a partir de Lim e de Max.*

$$MVE = \frac{\#\{p \in S_{\ell i} : p \leq Lim\} + Max}{\#\{p \in \mathcal{P} :\leq p \leq Lim\} + Max}$$

- Fatorizar Nf por pesquisa sequencial crescente de números primos a partir de 2, até encontrar Lim.
- Para cada número primo X de pesquisa, candidato a Lim , calculemos:
  - $\frac{\#\{p \in S_{\ell i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$
  - $Max = \frac{Nfi + Ndnf}{Ndmfp - 1}$
  - $MVE = \frac{\#\{p \in S_{\ell_i} : p \leq X\} + Max}{\#\{p \in \mathcal{P} : p \leq X\} + Max}$
  - Se $MVE \leq 2^{-i}$, então o nosso número primo X é o Lim que pesquisávamos e a partir do qual se verifica sempre a condição:

$$\frac{\#\{p \in S_{\ell i} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}} \leq 2^{-i} \qquad \forall X \geq Lim$$

(c) Condição 3: $p_{\ell_i \ell_j} > 2^i \qquad \forall j \leq i$

Dependendo da possibilidade de efetuar cálculos, utilizaremos alternativamente um dos seguites métodos:

Método 1:
Sabemos que:

- $p_{\ell_i \ell_j} = \max(S_{\ell_i \ell_j} - (S_{\ell_i} \cup S_{\ell_j}))$ (por definição)

- $(S_{\ell_i\ell_j} - (S_{\ell_i} \cup S_{\ell_j})) \neq \emptyset$ [Poo03a, Lema 3.4], se $\max\{\ell_i, \ell_j\}$ suficientemente grande

Nos casos em estudo, não se indo além da fatorização parcial (pelas razões já expostas), sabendo que $S_{\ell_i} = S_{\ell_i\ell_j} \cap S_{\ell_i}$ e $S_{\ell_j} = S_{\ell_i\ell_j} \cap S_{\ell_j}$ [Poo03a, Corolário 3.3], optou-se pela seguinte metodologia, para retirar as conclusões:

i. Cálculo de $\ell_i\ell_j P$, isolando o denominador da coordenada x e extraindo a raiz quadrada.

ii. Cálculo de $\ell_i P$, isolando o denominador da coordenada x e extraindo a raiz quadrada.

iii. Cálculo de $\ell_j P$, isolando o denominador da coordenada x e extraindo a raiz quadrada.

iv. Sendo ℓ_i e ℓ_j números primos, e $\ell_i \neq \ell_j$ os conjuntos S_{ℓ_i}, S_{ℓ_j} são disjuntos. Neste caso, dividimos o valor calculado em i) por o valor calculado em ii), sendo o resto $= 0$. Utilizamos o resultado da operação anterior (o quociente) para nova divisão pelo valor calculado em iii), sendo o resto $= 0$
Sendo ℓ_i e ℓ_j números primos, e $\ell_i = \ell_j$ os conjuntos S_{ℓ_i}, S_{ℓ_j} são iguais. Neste caso, dividimos o valor calculado em i) pelo valor calculado em ii) ou iii), sendo o resto $= 0$, só utilizando uma operação de divisão.

O resultado final obtido, um número não fatorizado se maior que 1 é o valor originador de $(S_{\ell_i\ell_j} - (S_{\ell_i} \cup S_{\ell_j}))$. Procedemos então a uma fatorização parcial do resultado final até ao fator que permita decidir sobre a condição 3.

Método 2:
Utilizaremos este método sempre que não nos seja possíve calcular $\ell_i\ell_j P$.
De acordo com a definição, temos que:

$$p_{\ell_i\ell_j} = \max(S_{\ell_i\ell_j} - (S_{\ell_i} \cup S_{\ell_j}))$$

e que pelo [Poo03a, Lema 3.4], $(S_{\ell_i\ell_j} - (S_{\ell_i} \cup S_{\ell_j})) \neq \emptyset$, se o $\max\{\ell_i, \ell_j\}$ é suficientemente grande.
Pelo método de redução módulo p, e na impossibilidade de calcular os fatores, asseguraremos que os numeros que tornam falsa a expressão $p_{\ell m} > 2^i$ ou não são fatores de $S_{\ell m}$, ou são igualmente fatores de $(S_\ell \cup S_m)$.

(d) Condição 4: $p_{\ell\ell_i} > 2^i \qquad \forall \ell \in L$

Sabemos que:

- $p_{\ell\ell_i} = \max(S_{\ell\ell_i} - (S_\ell \cup S_{\ell_i}))$ (por definição)
- $(S_{\ell\ell_i} - (S_\ell \cup S_{\ell_i})) \neq \emptyset$ [Poo03a, Lema 3.4], se $\max\{\ell, \ell_i\}$ for suficientemente grande

Nos casos em estudo, não se indo além da fatorização parcial (pelas razões já expostas), sabendo que $S_\ell = S_{\ell\ell_i} \cap S_\ell$ e $S_{\ell_i} = S_{\ell\ell_i} \cap S_{\ell_i}$ [Poo03a, Corolário 3.3], optou-se pela seguinte metodologia, para retirar as conclusões:

i. Cálculo de $\ell\ell_i P$, isolando o denominador da coordenada x e extraindo a raiz quadrada.

ii. Cálculo de ℓP, isolando o denominador da coordenada x e extraindo a raiz quadrada.

iii. Cálculo de $\ell_i P$, isolando o denominador da coordenada x e extraindo a raiz quadrada.

iv. Sendo ℓ e ℓ_i números primos, e $\ell \neq \ell_i$ os conjuntos S_ℓ, S_{ℓ_i} são disjuntos. Neste caso, dividimos o valor calculado em i) pelo valor calculado em ii), sendo o resto $= 0$. Utilizamos o resultado da operação anterior (o quociente) para nova divisão pelo valor calculado em iii), sendo o resto $= 0$

O resultado final obtido, um número não fatorizado se maior que 1 é o valor originador de $(S_{\ell\ell_i} - (S_\ell \cup S_{\ell_i}))$. Procedemos então a uma fatorização parcial do resultado final até ao fator que permita decidir sobre a condição 4.

(e) Condição 5: $|y(\ell_i P) - i| \leq 1/(10i)$
Determinar a coordenada Y de $\ell_i P$ e verificar que a condição se verifica de acordo com a expressão matemática.

(f) Condição 6: $\ell_i \notin L$
Verificar que $\ell_i \notin L$, sendo que L tem de ser previamente construído.

5.10.2 Construção do Conjunto de Candidatos ao Conjunto ℓ_i

De acordo com a metodologia definida, efetuou-se a identificação dos candidatos a elementos de ℓ_i.
Recorreu-se ao Sage, para os cálculos de ℓP, iniciando o cálculo em $\ell = 2$ calculando sucessivamente para todos os números primos seguintes, o valor de ℓP.
A pesquisa a elementos ℓ_i revelou-se cheia de dificuldades. As capacidades de processamento disponíveis, a enorme dimensão dos campos resultado, os meios disponíveis de programação, limitaram os resultados que se pretendiam obter.
O cálculo de mP, para valores baixos de m, foi suficientemente rápido. À medida que m crescia, os tempos de processamento degradavam-se acentuadamente. Baseado nas propriedades da adição dos pontos racionais em E, optou-se por cálculos parciais de pontos que foram posteriormente somados, tendo em consideração que: $mP = m_1P + m_2P + ... + m_nP$ sendo que $m = m_1 + m_2 + ... + m_n$. Isto permitiu levar a análise dos números primos bem mais longe.
A rotina em Sage é a seguinte:

Programa 5.10: Construção do Conjunto ℓ_i

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
nextPrime = 2                    #primeiro numero primo a calcular   EDITAVEIS
I = [];                           #conjunto l1 < l2 <    < li-1 ....  EDITAVEIS
Iindx = 1;                        #proximo i para li                  EDITAVEIS
ntimes =100                      #numero vezes do ciclo de calculo   EDITAVEIS

# Variáveis
ListIO = []                       #Intervalo dos números Primos Pesquisados

# Calculos iniciais
```

```
Initial_nextPrime = nextPrime;
Oindx = Iindx

for i in range(1):
   if  is_prime(nextPrime) == False:
       print ' O número ', nextPrime, 'não é primo'
       break
   if  nextPrime == 2:
       lastPrime = nextPrime
   else:
       lastPrime = previous_prime(nextPrime);
   m1 = lastPrime//3 + (lastPrime%3);
   m2 = lastPrime//3;
   m3 = lastPrime//3
   if m1 > 0:
      m1P = m1 * P; mP = m1P
   if m2 > 0:
      m2P = m2 * P; mP = mP + m2P
   if m3 > 0:
      m3P = m3 * P; mP = mP + m3P

# Ciclo calculo de mP e teste condicao 5
   for i in range(ntimes):
       m4 =nextPrime - lastPrime
       m4P= m4 * P;
       FmP = mP + m4P
       YmP = FmP[1] - Iindx
       if abs(YmP) <=  (1/(10 * Iindx)):
          I.append(nextPrime);
          Iindx = Iindx +1
          print 'O número primo', nextPrime, 'satisfaz condição 5 em Ii'
       nextPrime = Pr.next(nextPrime)

# Calculos Finais
ListIO.append(Initial_nextPrime);
LastPrime = previous_prime(nextPrime);
ListIO.append(LastPrime);
if    Initial_nextPrime <> nextPrime:
     print 'A lista dos candidatos a Ii é:  ',I
     print'O próximo valor de i,  é:  ', Iindx
     print'O intervalo dos números primos testados é:  ', ListIO
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
O número primo 97 satisfaz condição 5 em Ii
A lista dos candidatos a Ii é:   [97]
O próximo valor de i,  é:   2
O intervalo dos números primos testados é:   [2, 541]
O próximo número primo é:   547
```

Foi analisado (por execução sucessiva da rotina) o subconjunto finito dos números primos $A = \{2,3,5,7,11,...,27031\}$.
Satisfizeram a condição 5, os seguintes elementos:

(1) $\ell_1 = 97$

(2) $\ell_2 = 3299$

(3) $\ell_3 = 14369$

(4) A pesquisa de ℓ_4 foi efetuada até ao número primo 27031 no universo do subconjunto finito A. Não foi encontrado um número primo que satisfizesse a condição 5, logo o elemento ℓ_4, bem como todos os sucessores ℓ_5, ℓ_6, ...não foram identificados e têm um valor superior ao número primo 27031.

5.10.3 Candidato a $\ell_1 = 97$

O ponto $97P$ tem coordenadas x e y sendo que a raiz quadrada do denominador da coordenada x tem 973 dígitos.
Procedeu-se à fatorização parcial da raiz quadrada do denominador da coordenada x, tendo sido pesquisados fatores primos no intervalo $[2, 2038074743]$. Foram identificados 3 fatores primos e um número não fatorizado.

Programa 5.11: Fatorizar Denominador Coordenada X para o Ponto $97P$

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
```

```
Pr =Primes(); Pr                    #definir conjunto primos

# Parametros
nextPrime = 2                       #primeiro numero primo fator        EDITAVEIS
m=97;                               #numero m para calculo de mP       EDITAVEIS
Iindx = 1;                          #proximo i para fatores             EDITAVEIS
ntimes =100000000                   #numero vezes do ciclo de calculo   EDITAVEIS

# Variáveis
I = [];                             #fatores de denominador x(mP)       EDITAVEIS
ListIO = []                         #Intervalo dos números Primos Pesquisados

# Calculos iniciais
Initial_nextPrime = nextPrime;
Oindx = Iindx
for i in range(1):
   if  is_prime(nextPrime) == False:
       print ' O número ', nextPrime, 'não é primo'
       break
   mP = m * P;  XmP = mP[0];
   DmP = XmP.denominator(); RmP = sqrt(DmP)

# Ciclo fatorização de mP
   for i in range(ntimes):
       Reminder = RmP % nextPrime
       if Reminder == 0:
          I.append(nextPrime)
          Iindx = Iindx +1
          print nextPrime,'é fator primo do denominador coordenada X de ', m, 'P'
       nextPrime = Pr.next(nextPrime)

# Calculos Finais
ListIO.append(Initial_nextPrime);
LastPrime = previous_prime(nextPrime);
ListIO.append(LastPrime);
if   Initial_nextPrime <> nextPrime:
     print 'A lista dos fatores primos do denominador coordenada X ,de ', m, 'P, é: ',I
     print'O intervalo dos números primos testados é:  ', ListIO
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
97 é fator primo do denominador coordenada X de  97P
520081343 é fator primo do denominador coordenada X de  97P
882754619 é fator primo do denominador coordenada X de  97P
A lista dos fatores primos do denominador coordenada X ,de  97P, é:  [97, 520081343, 882754619]
O intervalo dos números primos testados é:   [2, 2038074743]
O próximo número primo é:   2038074751
```

$F_{97_{xDen}} = 97^2 \times 520081343^2 \times 882754619^2 \times$
1733014345954819445388584038922739103307671632740461608511345321914740
2730392490492979962377997499829302653147415554945070409544160585199754
3924196212928661185288522887783302190129959186949388917864266391119723
6612435401953281342209065426978934235612227399416603629636376745472496
6259774159353844369080466533269924422566340762113162521850018727875075
0736340831162141706445447127111401591257434367727266809222287722486618
5565638029330829966183250051954149004792886971083936937949258599730414
4510458610799545088952877853731962011028924619195047430299851269388271
2567514131574898336974206828190552172491547729225171824186821248869201
7712860447434413718246341839104182508038287743198437898842294056027355
4558223135695521190711680984331964617475403437302723834493540151790551
7483290620330162443405898834605905903766169488641527966092276906493123
4448763617548429386485904740053903574963981719847622267464033801779176
7347399422645686921475866398095378025549889 9^2$

Analisemos as 5 condições necessárias a ℓ_1

(1) Condição 1: $97 > \ell_j \qquad \forall j < 1$

A condição 1 não é aplicável ao primeiro elemento ℓ_1

(2) Condição 2: $\mu_{97} \leq 2^{-1}$

$$\mu_{97} = \sup_{X \in \mathbb{Z}_{\geq 2}} \frac{\#\{p \in S_{97} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$$

Já verificámos, que existem 3 fatores primos. Vamos então aplicar o Método 1, para esses fatores. Calculemos valores $\frac{\#\{p \in S_{97} : p \leq X\}}{\#\{p \in \mathcal{P} : p \leq X\}}$ para diferentes valores de X:

(a) $\frac{\#\{p \in S_{97} : p \leq 89\}}{\#\{p \in \mathcal{P} : p \leq 89\}} = 0$

(b) $\frac{\#\{p\in S_{97}:p\leq 97\}}{\#\{p\in \mathcal{P}:p\leq 97\}}=\frac{1}{25}$

Programa 5.12: Cardinal do Intervalo de Números Primos $(2,97)$

```
# Definicoes
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
First_Prime = 2                   #primeiro n. primo do [] cont       EDITAVEIS
Last_Prime = 97                   #ultimo n. primo do [] cont         EDITAVEIS

# Variáveis
Count_Primes = 0                  #contador de números primos
ListIO = []                       #Intervalo dos números primos contados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
nextPrime = First_Prime

# Calculos iniciais
ListIO.append(First_Prime);
ListIO.append(Last_Prime);

for i in range(1):
   if  is_prime(First_Prime) == False:
       print ' O número inicial', First_Prime, 'não é primo'
       Error_Prime = 1
       break
   if  is_prime(Last_Prime) == False:
       print ' O número final', Last_Prime, 'não é primo'
       Error_Prime = 1
       break

# Ciclo contagem de números primos no intervalo definido
   while (nextPrime  <= Last_Prime):
         Count_Primes=Count_Primes+1
         nextPrime = Pr.next(nextPrime)

# Calculos Finais
if   Error_Prime == 0:
     print 'O cardinal do conjunto dos números primos, no intervalo ',ListIO, 'é: ', Count_Primes
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Set of all prime numbers: 2, 3, 5, 7, ...
O cardinal do conjunto dos números primos, no intervalo  [2, 97] é:  25
O próximo número primo é:   101
```

(c) $\frac{\#\{p\in S_{97}:p\leq 520081343\}}{\#\{p\in \mathcal{P}:p\leq 520081343\}}=\frac{2}{27356746}$

Programa 5.13: Cardinal do Intervalo de Números Primos $(2,520081343)$

```
# Definicoes
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
First_Prime = 2                   #primeiro n. primo do [] cont       EDITAVEIS
Last_Prime = 520081343            #ultimo n. primo do [] cont         EDITAVEIS

# Variáveis
Count_Primes = 0                  #contador de números primos
ListIO = []                       #Intervalo dos números primos contados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
nextPrime = First_Prime

# Calculos iniciais
ListIO.append(First_Prime);
ListIO.append(Last_Prime);

for i in range(1):
   if  is_prime(First_Prime) == False:
       print ' O número inicial', First_Prime, 'não é primo'
       Error_Prime = 1
       break
   if  is_prime(Last_Prime) == False:
       print ' O número final', Last_Prime, 'não é primo'
       Error_Prime = 1
       break

# Ciclo contagem de números primos no intervalo definido
   while (nextPrime  <= Last_Prime):
         Count_Primes=Count_Primes+1
         nextPrime = Pr.next(nextPrime)

# Calculos Finais
if   Error_Prime == 0:
     print 'O cardinal do conjunto dos números primos, no intervalo ',ListIO, 'é: ', Count_Primes
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Set of all prime numbers: 2, 3, 5, 7, ...
O cardinal do conjunto dos números primos, no intervalo  [2, 520081343] é:  27356746
O próximo número primo é:   520081409
```

(d) $\frac{\#\{p\in S_{97}: p\leq 882754619\}}{\#\{p\in \mathcal{P}: p\leq 882754619\}} = \frac{3}{45173021}$

Programa 5.14: Cardinal do Intervalo de Números Primos $(2, 882754619)$

```
# Definicoes
Pr =Primes(); Pr                    #definir conjunto primos

# Parametros
First_Prime = 2                     #primeiro n. primo do [] cont       EDITAVEIS
Last_Prime = 882754619              #ultimo n. primo do [] cont         EDITAVEIS

# Variáveis
Count_Primes = 0                    #contador de números primos
ListIO = []                         #Intervalo dos números primos contados
Error_Prime = 0                     #Primeiro e Ultimo n. primo válido se 0
nextPrime = First_Prime

# Calculos iniciais
ListIO.append(First_Prime);
ListIO.append(Last_Prime);

for i in range(1):
   if  is_prime(First_Prime) == False:
       print ' O número inicial', First_Prime, 'não é primo'
       Error_Prime = 1
       break
   if  is_prime(Last_Prime) == False:
       print ' O número final', Last_Prime, 'não é primo'
       Error_Prime = 1
       break

# Ciclo contagem de números primos no intervalo definido
   while (nextPrime  <= Last_Prime):
         Count_Primes=Count_Primes+1
         nextPrime = Pr.next(nextPrime)

# Calculos Finais
if    Error_Prime == 0:
     print 'O cardinal do conjunto dos números primos, no intervalo ',ListIO, 'é: ', Count_Primes
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Set of all prime numbers: 2, 3, 5, 7, ...
O cardinal do conjunto dos números primos, no intervalo  [2, 882754619] é:  45173021
O próximo número primo é:   882754627
```

Vamos agora aplicar o Método 2 para cálculo do número primo Lim:

Programa 5.15: Cálculo de Lim para o Ponto $97P$

```
# Definicoes
E = EllipticCurve([1,1]) ; E        #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                     #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                    #definir conjunto primos

# Parametros
nextPrime = 2                       #primeiro numero primo fator        EDITAVEIS
m=97;                               #numero m para calculo de mP        EDITAVEIS
Iindx = 1;                          #i de li                            EDITAVEIS
ntimes =10000                       #numero vezes do ciclo de calculo   EDITAVEIS
I = [];                             #fatores de denominador x(mP)       EDITAVEIS
Ifact =[]                           #( # Sli / # num primos ) p/ fator  EDITAVEIS
numerador = 0                       # acumulado de fatores              EDITAVEIS
denominador = 0                     # Acumulado de numeros primos       EDITAVEIS

# Variáveis
limFound = 0
FirstnextPrime = nextPrime

# Calculos iniciais
for i in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número ', nextPrime, 'não é primo'
        limFound = 2
        break

    mP = m * P
    XmP = mP[0]
    DmP = XmP.denominator()
    RmP = sqrt(DmP)
    LengthRmP = len(str( RmP ));
    originalLen = LengthRmP

# Ciclo fatorização de mP
    for i in range(ntimes):

        Reminder = RmP % nextPrime

        if Reminder == 0:
           I.append(nextPrime)
           denominador = denominador + 1
```

```
            numerador = numerador + 1
            Ifact.append (numerador/denominador)
            RmP = RmP // nextPrime
            LengthRmP = len(str( RmP ));
         else:
            denominador = denominador + 1

#       Estimar limite

         LengthnextPrime = len( str (nextPrime) );

         if    LengthnextPrime > 1 :
               LengthnextPrime = LengthnextPrime -1;

         Aditional = LengthRmP // LengthnextPrime;
         numeradorVirtual = numerador
         denominadorVirtual = denominador
         numeradorVirtual = numeradorVirtual + Aditional;
         denominadorVirtual = denominadorVirtual + Aditional

         lim = numeradorVirtual / denominadorVirtual

         if   lim < 2^(-Iindx):
              limFound = 1
              break;

         nextPrime = Pr.next(nextPrime)
# Cálculos Finais
if      limFound == 0:
        print 'Aumente o contador ntimes, ou faça calculos parciais, para encontrar Lim'
if      limFound <= 1:
        print 'para ', m, 'P, temos:';
        print '            foi efetuada a fatorização de',FirstnextPrime, 'até ', nextPrime
        print '            foram encontrados os fatores: ', I
        print '            para os fatores( # Sli / # num primos ) são: ', Ifact
        print '            o Menor Número Primo ( Lim ) é:', nextPrime
        print '            Em Lim (# li / # num primos) é:', numerador, '/', denominador
        print '            o valor de MVE é', lim
        print '            o número de dígitos da raiz do denominador é', originalLen

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
para  97P, temos:
            foi efetuada a fatorização de 2 até  2161
            foram encontrados os fatores:  [97]
            para os fatores( # Sli / # num primos ) são:  [1/25]
            o Menor Número Primo ( Lim ) é: 2161
            Em Lim (# li / # num primos) é: 1 / 326
            o valor de MVE é 324/649
            o número de dígitos da raiz do denominador é 973
```

Estimámos deste modo:

- Menor Número Primo (Lim) = 2161
- Maior Valor Estimado (MVE)= $\frac{324}{649}$

Podemos então concluir:
$\frac{\#\{p\in S_{97}:p\leq 2161\}}{\#\{p\in\mathcal{P}:p\leq 2161\}} = \frac{1}{326} \leq 2^{-1}$

$\frac{\#\{p\in S_{97}:p\geq 2161\}}{\#\{p\in\mathcal{P}:p\geq 2161\}} \leq 2^{-1}$

Pela análise aos resultados obtidos em ambos os métodos utilizados , temos que até ao numero primo 2161, obtivemos os valores $\{0, 1/25, ..., 1/326\}$ todos satisfazendo a condição $\leq 2^{-1}$. Verificámos que para numeros primos ≥ 2161 os valores satisfazem a condição $\leq 2^{-1}$. Sendo μ_{97} o supremum destes valores, concluímos:

$$\mu_{97} \leq 2^{-1}$$

A condição 2 verifica-se para $\ell_1 = 97$

(3) Condição 3: $p_{97\ell_j} > 2^1 \qquad \forall j \leq 1$

Para $\ell_j = \ell_1$, temos que calcular $p_{\ell_i\ell_1} = p_{97\times 97} = p_{9409}$

(a) Cálculo de F_{9409} (raiz quadrada do denominador coordenada x do ponto $9409P$).

(b) Cálculo de F_{97} (raiz quadrada do denominador coordenada x do ponto $97P$).

(c) Cálculo de F_{9409}/F_{97} (Divisão de a) por b)).

(d) Pesquisar em F_{9409}/F_{97}, fatores primos.

Validando previamente que $F_{9409}/F_{97} > 1$, fez-se a fatorização parcial de F_{9409}/F_{97}, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{9409} = \max(S_{9409} - (S_{97} \cup S_{97}))) > 2$$

É-nos suficiente garantir que o primeiro numero primo da fatorização de $(F_{9409}/F_{97}) > 2$. Essa condição verifica-se, como é demonstrada na rotina desenvolvida em Sage.

Programa 5.16: Verificar que p_{9409} é Maior que 2

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 97                            #número l                          EDITAVEIS
m = 97                            #número m                          EDITAVEIS
i = 1;                            #valor de i para o numero li       EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo de lP e teste condicao 5

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad
    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if   Q1lmP % RmP == 0:
         Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

# Ciclo calculo de lP e teste condicao 5
    while   nextPrime <= Number_2i:
```

```
            Reminder = Q2lmP % nextPrime
            if   Reminder == 0:
                 I.append(nextPrime)
            nextPrime = Pr.next(nextPrime)


# Calculos Finais
Len_I = len( I );
if   Error_Prime == 0:
     if   Len_I == 0 and Q2lmP > 1:
          print 'l =', l, ', m =', m
          print 'O conjunto Slm - (Sl - Sm ) não é vazio'
          print 'p de ', l,'x', m, ' é maior que 2^', i
          print 'a pesquisa de fatorização terminou no numero primo', nextPrime
     else:
          print 'p de ', l,'x', m, ' é menor que 2^', i
if   Error_Prime == 2:
     print'Erro não previsto'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 97 , m = 97
O conjunto Slm - (Sl - Sm ) não é vazio
p de  9409  é maior que 2^ 1
a pesquisa de fatorização terminou no numero primo 3
```

Assim,

$p_{9409} > 2^1$

A condição 3 verifica-se para $\ell_j = 97$ onde $j = 1$

(4) Condição 4: $p_{\ell 97} > 2^1 \qquad \forall \ell \in L$

Já calculámos anteriormente $L = \{2,3\}$ e $\ell_1 = 97$

(a) Para $\ell = 2$, temos que calcular $p_{\ell\ell_1} = p_{2\times 97} = p_{194}$

i. Cálculo de F_{194} (raiz quadrada do denominador coordenada x do ponto $194P$).
ii. Cálculo de F_2 (raiz quadrada do denominador coordenada x do ponto $2P$).
iii. Cálculo de F_{97} (raiz quadrada do denominador coordenada x do ponto $97P$).
iv. Cálculo de F_{194}/F_2 (Divisão de i) por ii)).
v. Cálculo de $(F_{194}/F_2)/F_{97}$ (Divisão de iv) por iii)).
vi. Pesquisar em $(F_{194}/F_2)/F_{97}$, fatores primos.

Validando previamente que $(F_{194}/F_2)/F_{97} > 1$, fez-se a fatorização parcial de $(F_{194}/F_2)/F_{97}$, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{194} = \max(S_{194} - (S_2 \cup S_{97}))) > 2$$

É-nos suficiente garantir que o primeiro numero primo da fatorização de $(F_{194}/F_2)/F_{97} > 2$. Essa condição verifica-se, como é demonstrada na rotina desenvolvida em Sage.

Programa 5.17: Verificar que p_{194} é Maior que 2

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 2                             #número l                          EDITAVEIS
m = 97                            #número m                          EDITAVEIS
i = 1;                            #valor de i para o numero li       EDITAVEIS
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0
```

```
# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo de lP, mP e lmP

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad

    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if     Q1lmP % RmP == 0:
           Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

#
    while   nextPrime <= Number_2i:
            Reminder = Q2lmP % nextPrime
            if   Reminder == 0:
                 I.append(nextPrime)
            nextPrime = Pr.next(nextPrime)


# Calculos Finais
Len_I = len( I );
if   Error_Prime == 0:
     if   Len_I == 0 and Q2lmP > 1:
          print 'l =', l, ', m =', m
          print 'O conjunto Slm - (Sl - Sm ) não é vazio'
          print 'p de ',lm, ' é maior que 2^', i
          print 'a pesquisa de fatorização terminou no numero primo', nextPrime
     else:
          print 'p de ', lm, ' é menor que 2^', i
if   Error_Prime == 2:
     print'Erro não previsto'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 2 , m = 97
O conjunto Slm - (Sl - Sm ) não é vazio
p de  194  é maior que 2^ 1
a pesquisa de fatorização terminou no numero primo 3
```

Assim,

$p_{194} > 2^1$

Concluímos que a condição 4 é verificada para $\ell = 2$:

(b) Para $\ell = 3$, temos que calcular $p_{\ell\ell_1} = p_{3\times 97} = p_{291}$

i. Cálculo de F_{291} (raiz quadrada do denominador coordenada x do ponto $291P$).

ii. Cálculo de F_3 (raiz quadrada do denominador coordenada x do ponto $3P$).

iii. Cálculo de F_{97} (raiz quadrada do denominador coordenada x do ponto $97P$).

iv. Cálculo de F_{291}/F_3 (Divisão de i) por ii)).

v. Cálculo de $(F_{291}/F_3)/F_{97}$ (Divisão de iv) por iii)).

vi. Pesquisar em $(F_{291}/F_3)/F_{97}$, fatores primos.

Validando previamente que $(F_{291}/F_3)/F_{97} > 1$, fez-se a fatorização parcial de $(F_{291}/F_3)/F_{97}$, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{291} = \max(S_{291} - (S_3 \cup S_{97}))) > 2$$

É suficiente garantir que o primeiro numero primo da fatorização de $(F_{291}/F_3)/F_{97} > 2$. Essa condição verifica-se, como é demonstrada na rotina desenvolvida em Sage.

Programa 5.18: Verificar que p_{291} é Maior que 2

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 3                             #número l                          EDITAVEIS
m = 97                            #número m                          EDITAVEIS
i = 1;                            #valor de i para o numero li       EDITAVEIS
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0

# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo de lP, mP e lmP

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad

    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if     Q1lmP % RmP == 0:
           Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

#
    while   nextPrime <= Number_2i:
            Reminder = Q2lmP % nextPrime
            if   Reminder == 0:
                 I.append(nextPrime)
```

```
            nextPrime = Pr.next(nextPrime)


# Calculos Finais
Len_I = len( I );
if    Error_Prime == 0:
      if    Len_I == 0 and Q2lmP > 1:
            print 'l =', l, ', m =', m
            print 'O conjunto Slm - (Sl - Sm ) não é vazio'
            print 'p de ', lm, ' é maior que 2^', i
            print 'a pesquisa de fatorização terminou no numero primo', nextPrime
      else:
            print 'p de ', lm, ' é menor que 2^', i
if    Error_Prime == 2:
      print'Erro não previsto'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 3 , m = 97
O conjunto Slm - (Sl - Sm ) não é vazio
p de  291  é maior que 2^ 1
a pesquisa de fatorização terminou no numero primo 3
```

Assim,

$p_{291} > 2^1$

Concluímos que a condição 4 é verificada para $\ell = 3$:

A condição 4 verifica-se para $\ell_1 = 97$

(5) Condição 5: $|y(\ell_1 P) - 1| \leq 1/(10 \times 1)$

A condição 5 verifica-se para $\ell_1 = 97$, com recurso a uma rotina desenvolvida utilizando o software Sage.

Programa 5.19: Verificar que $97P$ Satisfaz a Condição 5 de ℓ_i

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
m = 97                            #número m                          EDITAVEIS
Iindx = 1;                        #valor i para li                   EDITAVEIS


# Calculos iniciais

for k in range(1):
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        break

# Verificar se mP satisfaz condicao 5

    mP = m * P
    YmP = mP[1] - Iindx;
    if    abs(YmP) <=  (1/(10 * Iindx)):
          print 'O número primo', m, 'satisfaz condição 5 em li, para o valor de i=', Iindx
    else:
          print 'O número primo', m, 'não satisfaz condição 5 em li, para o valor de i=', Iindx

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
O número primo 97 satisfaz condição 5 em li, para o valor de i= 1
```

5.10.4 Candidato a $\ell_2 = 3299$

O ponto $3299P$ tem coordenadas x e y sendo que a raiz quadrada do denominador da coordenada x tem 1125459 dígitos.

Procedeu-se à fatorização parcial da raiz quadrada do denominador da coordenada x, tendo sido pesquisados fatores primos no intervalo $[2, 18200410997]$. Neste intervalo não existe nenhum número primo que seja fator do denominador da coordenada x, o que significa que todos os fatores primos têm um mínimo de 11 dígitos e um valor superior a 18200410997.

Programa 5.20: Fatorizar Denominador Coordenada X para o Ponto 3299P

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
nextPrime = 18100000013           #primeiro numero primo fator        EDITAVEIS
m=3299;                           #numero m para calculo de mP        EDITAVEIS
Iindx = 1;                        #proximo i para fatores             EDITAVEIS
ntimes =4249389                   #numero vezes do ciclo de calculo   EDITAVEIS
I = [];                           #fatores de denominador x(mP)       EDITAVEIS

# Variáveis
ListIO = []                       #Intervalo dos números Primos Pesquisados

# Calculos iniciais
Initial_nextPrime = nextPrime;
Oindx = Iindx
for i in range(1):
   if  is_prime(nextPrime) == False:
       print ' O número ', nextPrime, 'não é primo'
       break
   mP = m * P;  XmP = mP[0];
   DmP = XmP.denominator(); RmP = sqrt(DmP)

# Ciclo fatorização de mP
   for i in range(ntimes):
       Reminder = RmP % nextPrime;
       if Reminder == 0:
          I.append(nextPrime)
          Iindx = Iindx +1
          print nextPrime,'é fator primo do denominador coordenada X de ', m, 'P'
       nextPrime = Pr.next(nextPrime)

# Calculos Finais
ListIO.append(Initial_nextPrime);
LastPrime = previous_prime(nextPrime);
ListIO.append(LastPrime);
if   Initial_nextPrime <> nextPrime:
     print 'A lista dos fatores primos do denominador coordenada X ,de ', m, 'P, é: ',I
     print'O intervalo dos números primos testados é:  ', ListIO
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
A lista dos fatores primos do denominador coordenada X ,de  3299P, é:  []
O intervalo dos números primos testados é:   [18100000013, 18200410997]
O próximo número primo é:   18200411029
```

$F_{3299_{xDen}} = Numero\ nao\ fatorizado\ em\ primos\ com\ (1125459\ digitos)^2$

Analisemos as 5 condições necessária a ℓ_2

(1) Condição 1: $3299 > \ell_j \qquad \forall j < 2$
Temos que:
$3299 > 97$ para $j = 1$
A condição 1 verifica-se para $\ell_2 = 3299$

(2) Condição 2: $\mu_{3299} \leq 2^{-2}$

Não tendo sido encontrados quaisquer fatores primos, vamos utilizar o Método 2 para cálculo de Lim.

Programa 5.21: Cálculo de Lim para o Ponto 3299P

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
nextPrime = 8300041               #primeiro numero primo fator        EDITAVEIS
m=3299;                           #numero m para calculo de mP        EDITAVEIS
Iindx = 2;                        #i de li                            EDITAVEIS
ntimes = 10000                    #numero vezes do ciclo de calculo   EDITAVEIS
I = [];                           #fatores de denominador x(mP)       EDITAVEIS
Ifact =[]                         #( # Sli / # num primos ) p/ fator  EDITAVEIS
numerador = 0                     # acumulado de fatores               EDITAVEIS
denominador = 558598              # Acumulado de numeros primos        EDITAVEIS

# Variáveis

limFound = 0
FirstnextPrime = nextPrime
```

```
# Calculos iniciais
for i in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número ', nextPrime, 'não é primo'
        limFound = 2
        break

    mP = m * P
    XmP = mP[0]
    DmP = XmP.denominator()
    RmP = sqrt(DmP)
    LengthRmP = len(str( RmP ));
    originalLen = LengthRmP


# Ciclo fatorização de mP
    for i in range(ntimes):

        Reminder = RmP % nextPrime

        if Reminder == 0:
           I.append(nextPrime)
           denominador = denominador + 1
           numerador = numerador + 1
           Ifact.append (numerador/denominador)
           RmP = RmP // nextPrime
           LengthRmP = len(str( RmP ));
        else:
           denominador = denominador + 1

#      Estimar limite

        LengthnextPrime = len( str (nextPrime) );

        if    LengthnextPrime > 1 :
              LengthnextPrime = LengthnextPrime -1;

        Aditional = LengthRmP // LengthnextPrime;
        numeradorVirtual = numerador
        denominadorVirtual = denominador
        numeradorVirtual = numeradorVirtual + Aditional;
        denominadorVirtual = denominadorVirtual + Aditional

        lim = numeradorVirtual / denominadorVirtual

        if   lim < 2^(-Iindx):
             limFound = 1
             break;

        nextPrime = Pr.next(nextPrime)
# Cálculos Finais
if     limFound == 0:
       print 'Aumente o contador ntimes, ou faça calculos parciais, para encontrar Lim'
if     limFound <= 1:
       print 'para ', m, 'P, temos:';
       print '            foi efetuada a fatorização de',FirstnextPrime, 'até ', nextPrime
       print '            foram encontrados os fatores: ', I
       print '            para os fatores( # Sli / # num primos ) são: ', Ifact
       print '            o Menor Número Primo ( Lim ) é:', nextPrime
       print '            Em Lim (# li / # num primos) é:', numerador, '/', denominador
       print '            o valor de MVE é', lim
       print '            o número de dígitos da raiz do denominador é', originalLen

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
para  3299P, temos:
            foi efetuada a fatorização de 8300041 até  8365783
            foram encontrados os fatores:  []
            para os fatores( # Sli / # num primos ) são:  []
            o Menor Número Primo ( Lim ) é: 8365783
            Em Lim (# li / # num primos) é: 0 / 562729
            o valor de MVE é 187576/750305
            o número de dígitos da raiz do denominador é 1125459
```

Estimámos deste modo:

- Menor Número Primo (Lim) = 8365783
- Maior Valor Estimado (MVE)= $\frac{187576}{750305}$

Podemos então concluir:

$$\frac{\#\{p \in S_{3299} : p \leq 8365783\}}{\#\{p \in \mathcal{P} : p \leq 8365783\}} = \frac{0}{562729} \leq 2^{-2}$$

$$\frac{\#\{p \in S_{3299} : p \geq 8365783\}}{\#\{p \in \mathcal{P} : p \geq 8365783\}} \leq 2^{-2}$$

Logo todos os valores satisfazem a condição $\leq 2^{-2}$. Sendo μ_{3299} o supremum destes valores, concluímos:

$$\mu_{3299} \leq 2^{-2}$$

A condição 2 verifica-se para $\ell_2 = 3299$

(3) Condição 3: $p_{3299\ell_j} > 2^2 \qquad \forall j \leq 2$

(a) Para $\ell_j = \ell_1$, temos que calcular $p_{\ell_i\ell_1} = p_{3299\times 97} = p_{320003}$

De acordo com a definição, temos que:

$$p_{320003} = \max(S_{320003} - (S_{97} \cup S_{3299}))$$

e que pelo [Poo03a, Lema 3.4], $(S_{320003} - (S_{97} \cup S_{3299}))$ não é um conjunto vazio, se o $\max\{97, 3299\}$ é suficientemente grande. Na impossibilidade de calcular $320003P$ vamos utilizar o método 2 para provar a condição 3 com recurso à redução mod p.

Programa 5.22: Geração $\mathbb{F}_p$ onde p menor 107 para o Ponto $320003P$

```
# Parametros
nextPrime= 3                  #número p                          EDITAVEIS
Last_Prime = 107              #Fim de geração de p               EDITAVEIS
li=320003                     #numero primo a fatorizar          EDITAVEIS
pontoSingular=31              #numero singular                   EDITAVEIS

# Definicoes
Pr =Primes(); Pr                  #definir conjunto primos

# Variáveis
Error_Prime = 0
FirstnextPrime = nextPrime
l1=[]

# Calculos iniciais

for k in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número nextPrime =', nextPrime, ', não é primo'
        Error_Prime = 1
        break;

# Ciclo calculo da ordem de P


    while (nextPrime  <= Last_Prime):
       F = Zmod(nextPrime);
       E = EllipticCurve(F,[1, 1]);
       P = E([0,1]);
       mP=li*P;
       Y = mP[2];

       if    Y == 0:

             l1.append(nextPrime);

       nextPrime = Pr.next(nextPrime);
       if    nextPrime == pontoSingular:
             nextPrime = Pr.next(nextPrime);
if     Error_Prime == 0:
       print 'para ', li, 'P, temos:';
       print '            foi efetuada redução mod p no intervalo de ',FirstnextPrime, 'até ', Last_Prime
       print '            foram encontrados os fatores : ', l1

#Output
Set of all prime numbers: 2, 3, 5, 7, ...
para  320003P, temos:
            foi efetuada redução mod p no intervalo de  3 até  107
            foram encontrados os fatores :  [97]
```

Verificamos que o numero 3 não é fator do ponto $320003P$ e sendo $(S_{320003} - (S_{97} \cup S_{3299})) \neq \emptyset$, então podemos concluir que:

$$p_{320003} > 2^2 \qquad \text{para } \ell_j = 97 \text{ onde } j = 1$$

(b) Para $\ell_j = \ell_2$, temos que calcular $p_{\ell_i \ell_2} = p_{3299\times 3299} = p_{10883401}$

De acordo com a definição, temos que:

$$p_{10883401} = \max(S_{10883401} - (S_{3299} \cup S_{3299}))$$

e que pelo [Poo03a, Lema 3.4], $(S_{10883401} - (S_{3299} \cup S_{3299}))$ não é um conjunto vazio, se o $\max\{3299, 3299\}$ é suficientemente grande. Na impossibilidade de calcular $10883401P$ vamos utilizar o método 2 para provar a condição 3 com recurso à redução mod p.

Programa 5.23: Geração $\mathbb{F}_p$ onde p menor 107 para o Ponto $10883401P$

```
# Parametros
nextPrime= 3                  #número p                          EDITAVEIS
Last_Prime = 107              #Fim de geração de p               EDITAVEIS
li=10883401                   #numero primo a fatorizar          EDITAVEIS
pontoSingular=31              #numero singular                   EDITAVEIS

# Definicoes
Pr =Primes(); Pr                  #definir conjunto primos

# Variáveis
Error_Prime = 0
FirstnextPrime = nextPrime
l1=[]

# Calculos iniciais

for k in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número nextPrime =', nextPrime, ', não é primo'
        Error_Prime = 1
        break;

# Ciclo calculo da ordem de P


    while (nextPrime  <= Last_Prime):
       F = Zmod(nextPrime);
       E = EllipticCurve(F,[1, 1]);
       P = E([0,1]);
       mP=li*P;
       Y = mP[2];

       if    Y == 0:

             l1.append(nextPrime);

       nextPrime = Pr.next(nextPrime);
       if    nextPrime == pontoSingular:
             nextPrime = Pr.next(nextPrime);
if     Error_Prime == 0:
       print 'para ', li, 'P, temos:';
       print '             foi efetuada redução mod p no intervalo de ',FirstnextPrime, 'até ', Last_Prime
       print '             foram encontrados os fatores: ', l1

#Output
Set of all prime numbers: 2, 3, 5, 7, ...
para  10883401P, temos:
            foi efetuada redução mod p no intervalo de  3 até  107
            foram encontrados os fatores:  []
```

Verificamos que o numero 3 não é fator do ponto $10883401P$ e sendo $(S_{10883401} - (S_{3299} \cup S_{3299})) \neq \emptyset$, então podemos concluir que:

$$p_{10883401} > 2^2 \qquad \text{para } \ell_j = 3299 \text{ onde } j = 2$$

Assim, a condição 3 verifica-se para $\ell_2 = 3299$

(4) Condição 4: $p_{\ell 3299} > 2^2 \qquad \forall \ell \in L$

Já calculámos anteriormente $L = \{2, 3\}$ e $\ell_2 = 3299$

(a) Para $\ell = 2$, temos que calcular $p_{\ell \ell_2} = p_{2\times 3299} = p_{6598}$

i. Cálculo de F_{6598} (raiz quadrada do denominador coordenada x do ponto $6598P$).
ii. Cálculo de F_2 (raiz quadrada do denominador coordenada x do ponto $2P$).

iii. Cálculo de F_{3299} (raiz quadrada do denominador coordenada x do ponto $3299P$).

iv. Cálculo de F_{6598}/F_2 (Divisão de i) por ii)).

v. Cálculo de $(F_{6598}/F_2)/F_{3299}$ (Divisão de iv) por iii)).

vi. Pesquisar em $(F_{6598}/F_2)/F_{3299}$, fatores primos.

Validando previamente que $(F_{6598}/F_2)/F_{3299} > 1$, fez-se a fatorização parcial de $(F_{6598}/F_2)/F_{3299}$, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{6598} = \max(S_{6598} - (S_2 \cup S_{3299}))) > 2^2$$

É suficiente garantir que o primeiro numero primo da fatorização de $(F_{6598}/F_2)/F_{3299} > 2$. Essa condição verifica-se, como é demonstrada na rotina desenvolvida em Sage.

Programa 5.24: Verificar que p_{6598} é Maior que 2^2

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 2                             #número l                          EDITAVEIS
m = 3299                          #número m                          EDITAVEIS
i = 2;                            #valor de i para o numero li       EDITAVEIS
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0

# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo de lP, mP e lmP

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad

    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if     Q1lmP % RmP == 0:
           Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

#
    while   nextPrime <= Number_2i:
            Reminder = Q2lmP % nextPrime
```

```
            if   Reminder == 0:
                 I.append(nextPrime)
            nextPrime = Pr.next(nextPrime)


# Calculos Finais
Len_I = len( I );
if   Error_Prime == 0:
     if   Len_I == 0 and Q2lmP > 1:
          print 'l =', l, ', m =', m
          print 'O conjunto Slm - (Sl - Sm ) não é vazio'
          print 'p de ', lm, ' é maior que 2^', i
          print 'a pesquisa de fatorização terminou no numero primo', nextPrime
     else:
          print 'p de ', lm, ' é menor que 2^', i
if   Error_Prime == 2:
     print'Erro não previsto'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 2 , m = 3299
O conjunto Slm - (Sl - Sm ) não é vazio
p de  6598  é maior que 2^ 2
a pesquisa de fatorização terminou no numero primo 5
```

Assim,

$p_{6598} > 2^2$

Concluímos que a condição 4 é verificada para $\ell = 2$:

(b) Para $\ell = 3$, temos que calcular $p_{\ell\ell_1} = p_{3\times 3299} = p_{9897}$

i. Cálculo de F_{9897} (raiz quadrada do denominador coordenada x do ponto $9897P$).
ii. Cálculo de F_3 (raiz quadrada do denominador coordenada x do ponto $3P$).
iii. Cálculo de F_{3299} (raiz quadrada do denominador coordenada x do ponto $3299P$).
iv. Cálculo de F_{9897}/F_3 (Divisão de i) por ii)).
v. Cálculo de $(F_{9897}/F_3)/F_{3299}$ (Divisão de iv) por iii)).
vi. Pesquisar em $(F_{9897}/F_3)/F_{3299}$, fatores primos.

Validando previamente que $(F_{9897}/F_3)/F_{3299} > 1$, fez-se a fatorização parcial de $(F_{9897}/F_3)/F_{3299}$, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{9897} = \max(S_{9897} - (S_3 \cup S_{3299}))) > 2$$

É suficiente garantir que o primeiro numero primo da fatorização de $(F_{9897}/F_3)/F_{3299} > 2$. Essa condição verifica-se, como é demonstrada na rotina desenvolvida em Sage.

Programa 5.25: Verificar que p_{9897} é Maior que 2^2

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 3                             #número l                          EDITAVEIS
m = 3299                          #número m                          EDITAVEIS
i = 2;                            #valor de i para o numero li       EDITAVEIS
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0

# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
```

```
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo de lP, mP e lmP

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad

    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if     Q1lmP % RmP == 0:
           Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

#
    while   nextPrime <= Number_2i:
            Reminder = Q2lmP % nextPrime
            if   Reminder == 0:
                 I.append(nextPrime)
            nextPrime = Pr.next(nextPrime)


# Calculos Finais
Len_I = len( I );
if   Error_Prime == 0:
     if   Len_I == 0 and Q2lmP > 1:
          print 'l =', l, ', m =', m
          print 'O conjunto Slm - (Sl - Sm ) não é vazio'
          print 'p de ', lm, ' é maior que 2^', i
          print 'a pesquisa de fatorização terminou no numero primo', nextPrime
     else:
          print 'p de ', lm, ' é menor que 2^', i
if   Error_Prime == 2:
     print'Erro não previsto'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 3 , m = 3299
O conjunto Slm - (Sl - Sm ) não é vazio
p de  9897  é maior que 2^ 2
a pesquisa de fatorização terminou no numero primo 5
```

Assim,

$p_{9897} > 2^2$

Concluí-se que a condição 4 é verificada para $\ell = 3$:

A condição 4 verifica-se para $\ell_2 = 3299$

(5) Condição 5: $|y(\ell_2 P) - 2| \leq 1/(10 \times 2)$
A condição 5 verifica-se para $\ell_2 = 3299$, com recurso a uma rotina desenvolvida utilizando o software Sage.

Programa 5.26: Verificar que $3299P$ Satisfaz a Condição 5 de ℓ_i

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos
```

```
# Parametros
m = 3299                          #número m                          EDITAVEIS
Iindx = 2;                        #valor i para li                   EDITAVEIS

# Calculos iniciais

for k in range(1):
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        break

# Verificar se mP satisfaz condicao 5

    mP = m * P
    YmP = mP[1] - Iindx;
    if   abs(YmP) <=  (1/(10 * Iindx)):
         print 'O número primo', m, 'satisfaz condição 5 em li, para o valor de i=', Iindx
    else:
         print 'O número primo', m, 'não satisfaz condição 5 em li, para o valor de i=', Iindx

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
O número primo 3299 satisfaz condição 5 em li, para o valor de i= 2
```

Concluímos deste modo, confirmadas as 5 condições, que o candidato a ℓ_2, o número primo 3299 é o valor de ℓ_2 para a curva elíptica em análise.

5.10.5 Candidato a $\ell_3 = 14369$

O ponto 14369P tem coordenadas x e y sendo que a raiz quadrada do denominador da coordenada x tem 21350983 dígitos.
Procedeu-se à fatorização parcial da raiz quadrada do denominador da coordenada x, tendo sido pesquisados fatores primos no intervalo $[2, 369625447]$. Neste intervalo não existe nenhum número primo que seja fator do denominador da coordenada x, o que significa que todos os fatores primos têm um mínimo de 8 dígitos e um valor superior a 369625447.

Programa 5.27: Fatorizar Denominador Coordenada X para o Ponto 14369P

```
#LaTex: caption = Fatorizar denominador coordenada X para o ponto 14369P
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
nextPrime = 359765669             #primeiro numero primo fator        EDITAVEIS
m=14369                           #numero m para calculo de mP        EDITAVEIS
Iindx = 1;                        #proximo i para fatores             EDITAVEIS
ntimes =500000                    #numero vezes do ciclo de calculo   EDITAVEIS

# Variáveis
I = [];                           #fatores de denominador x(mP)       EDITAVEIS
ListIO = []                       #Intervalo dos números Primos Pesquisados

# Calculos iniciais
Initial_nextPrime = nextPrime;
Oindx = Iindx
for i in range(1):
   if  is_prime(nextPrime) == False:
       print ' O número ', nextPrime, 'não é primo'
       break
   mP = m * P;  XmP = mP[0];
   DmP = XmP.denominator(); RmP = sqrt(DmP)

# Ciclo fatorização de mP
   for i in range(ntimes):
       Reminder = RmP % nextPrime;
       if Reminder == 0:
          I.append(nextPrime)
          Iindx = Iindx +1
          print nextPrime,'é fator primo do denominador coordenada X de ', m, 'P'
       nextPrime = Pr.next(nextPrime)

# Calculos Finais
ListIO.append(Initial_nextPrime);
LastPrime = previous_prime(nextPrime);
ListIO.append(LastPrime);
if   Initial_nextPrime <> nextPrime:
     print 'A lista dos fatores primos do denominador coordenada X ,de ', m, 'P, é: ',I
     print'O intervalo dos números primos testados é:  ', ListIO
     print'O próximo número primo é:  ', nextPrime

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
```

```
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
A lista dos fatores primos do denominador coordenada X ,de  14369P, é:  []
O intervalo dos números primos testados é:   [359765669, 369625447]
O próximo número primo é:   369625507
```

$F_{14369_{xDen}} = Nmero\ no\ fatorizado\ em\ primos\ com\ (21350983\ dgitos)^2$

Analisemos as 5 condições necessária a ℓ_3

(1) Condição 1: $14369 > \ell_j \qquad \forall j < 1$
Temos que:
$14369 > 97$ para $j = 1$
$14369 > 3299$ para $j = 2$
A condição 1 verifica-se para $\ell_3 = 14369$

(2) Condição 2: $\mu_{14369} \leq 2^{-3}$

Não tendo sido encontrados quaisquer fatores primos, vamos utilizar o Método 2 para cálculo de Lim

Programa 5.28: Cálculo de Lim para o Ponto 14369*P*

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
nextPrime = 347612339             #primeiro numero primo fator        EDITAVEIS
m=14369;                          #numero m para calculo de mP        EDITAVEIS
Iindx = 3;                        #i de li                            EDITAVEIS
ntimes =10000                     #numero vezes do ciclo de calculo   EDITAVEIS
I = [];                           #fatores de denominador x(mP)       EDITAVEIS
Ifact =[]                         #( # Sli / # num primos ) p/ fator  EDITAVEIS
numerador = 0                     # acumulado de fatores              EDITAVEIS
denominador = 18682076            # Acumulado de numeros primos       EDITAVEIS

# Variáveis
limFound = 0
FirstnextPrime = nextPrime

# Calculos iniciais
for i in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número ', nextPrime, 'não é primo'
        limFound = 2
        break

    mP = m * P
    XmP = mP[0]
    DmP = XmP.denominator()
    RmP = sqrt(DmP)
    LengthRmP = len(str( RmP ));
    originalLen = LengthRmP


# Ciclo fatorização de mP
    for i in range(ntimes):

        Reminder = RmP % nextPrime

        if Reminder == 0:
           I.append(nextPrime)
           denominador = denominador + 1
           numerador = numerador + 1
           Ifact.append (numerador/denominador)
           RmP = RmP // nextPrime
           LengthRmP = len(str( RmP ));
        else:
           denominador = denominador + 1

#      Estimar limite

        LengthnextPrime = len( str (nextPrime) );

        if    LengthnextPrime > 1 :
              LengthnextPrime = LengthnextPrime -1;

        Aditional = LengthRmP // LengthnextPrime;
        numeradorVirtual = numerador
        denominadorVirtual = denominador
        numeradorVirtual = numeradorVirtual + Aditional;
        denominadorVirtual = denominadorVirtual + Aditional

        lim = numeradorVirtual / denominadorVirtual
```

```
        if   lim < 2^(-Iindx):
             limFound = 1
             break;

        nextPrime = Pr.next(nextPrime)
# Cálculos Finais
if     limFound == 0:
       print 'Aumente o contador ntimes, ou faça calculos parciais, para encontrar Lim'
if     limFound <= 1:
       print 'para ', m, 'P, temos:';
       print '            foi efetuada a fatorização de',FirstnextPrime, 'até ', nextPrime
       print '            foram encontrados os fatores: ', I
       print '            para os fatores( # Sli / # num primos ) são: ', Ifact
       print '            o Menor Número Primo ( Lim ) é:', nextPrime
       print '            Em Lim (# li / # num primos) é:', numerador, '/', denominador
       print '            o valor de MVE é', lim
       print '            o número de dígitos da raiz do denominador é', originalLen

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
para  14369P, temos:
            foi efetuada a fatorização de 347612339 até  347613029
            foram encontrados os fatores:  []
            para os fatores( # Sli / # num primos ) são:  []
            o Menor Número Primo ( Lim ) é: 347613029
            Em Lim (# li / # num primos) é: 0 / 18682105
            o valor de MVE é 2668872/21350977
            o número de dígitos da raiz do denominador é 21350983
```

Estimámos deste modo:

- Menor Número Primo (Lim) = 347613029
- Maior Valor Estimado (MVE)= $\frac{2668872}{21350977}$

Podemos então concluir:
$\frac{\#\{p\in S_{14369}:p\leq 347613029\}}{\#\{p\in\mathcal{P}:p\leq 347613029\}} = \frac{0}{18682105} \leq 2^{-3}$

$\frac{\#\{p\in S_{14369}:p\geq 347613029\}}{\#\{p\in\mathcal{P}:p\geq 347613029\}} \leq 2^{-3}$

Logo todos os valores satisfazem a condição $\leq 2^{-3}$. Sendo μ_{14369} o supremum destes valores, concluímos:

$$\mu_{14369} \leq 2^{-3}$$

A condição 2 verifica-se para $\ell_3 = 14369$

(3) Condição 3: $p_{14369\ell_j} > 2^3 \qquad \forall j \leq 3$

(a) Para $\ell_j = \ell_1$, temos que calcular $p_{\ell_i\ell_1} = p_{14369\times 97} = p_{1393793}$

De acordo com a definição, temos que:

$$p_{1393793} = \max(S_{1393793} - (S_{97} \cup S_{14369}))$$

e que pelo [Poo03a, Lema 3.4], $(S_{1393793} - (S_{97} \cup S_{14369}))$ não é um conjunto vazio, se o $\max\{97, 14369\}$ é suficientemente grande. Na impossibilidade de calcular $1393793P$ vamos utilizar o método 2 para provar a condição 3 com recurso à redução mod p.

Programa 5.29: Geração $\mathbb{F}_p$ onde p menor 107 para o Ponto $1393793P$

```
# Parametros
nextPrime= 3                #número p                          EDITAVEIS
Last_Prime = 107            #Fim de geração de p               EDITAVEIS
li=1393793                  #numero primo a fatorizar          EDITAVEIS
```

```
pontoSingular=31             #numero singular                    EDITAVEIS

# Definicoes
Pr =Primes(); Pr                 #definir conjunto primos

# Variáveis
Error_Prime = 0
FirstnextPrime = nextPrime
l1=[]

# Calculos iniciais

for k in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número nextPrime =', nextPrime, ', não é primo'
        Error_Prime = 1
        break;

# Ciclo calculo da ordem de P


    while (nextPrime  <= Last_Prime):
       F = Zmod(nextPrime);
       E = EllipticCurve(F,[1, 1]);
       P = E([0,1]);
       mP=li*P;
       Y = mP[2];

       if    Y == 0:

             l1.append(nextPrime);

       nextPrime = Pr.next(nextPrime);
       if    nextPrime == pontoSingular:
             nextPrime = Pr.next(nextPrime);
if     Error_Prime == 0:
       print 'para ', li, 'P, temos:';
       print '             foi efetuada redução mod p no intervalo de ',FirstnextPrime, 'até ', Last_Prime
       print '             foram encontrados os fatores: ', l1

#Output
Set of all prime numbers: 2, 3, 5, 7, ...
para  1393793P, temos:
            foi efetuada redução mod p no intervalo de  3 até  107
            foram encontrados os fatores:  [97]
```

Verificamos que os numeros $3,5,7$ não são fatores do ponto $1393793P$ e sendo $(S_{1393793} - (S_{97} \cup S_{14369})) \neq \emptyset$, então podemos concluir que:

$$p_{1393793} > 2^3 \qquad \text{para } \ell_j = 97 \text{ onde } j = 1$$

(b) Para $\ell_j = \ell_2$, temos que calcular $p_{\ell_i \ell_2} = p_{14369 \times 3299} = p_{47403331}$

De acordo com a definição, temos que:

$$p_{47403331} = \max(S_{47403331} - (S_{3299} \cup S_{14369}))$$

e que pelo [Poo03a, Lema 3.4], $(S_{47403331} - (S_{3299} \cup S_{14369}))$ não é um conjunto vazio, se o $\max\{3299, 14369\}$ é suficientemente grande. Na impossibilidade de calcular $47403331P$ vamos utilizar o método 2 para provar a condição 3 com recurso à redução mod p.

Programa 5.30: Geração $\mathbb{F}_p$ onde p menor 107 para o Ponto $47403331P$

```
# Parametros
nextPrime= 3                 #número p                           EDITAVEIS
Last_Prime = 107             #Fim de geração de p                EDITAVEIS
li=47403331                  #numero primo a fatorizar           EDITAVEIS
pontoSingular=31             #numero singular                    EDITAVEIS

# Definicoes
Pr =Primes(); Pr                 #definir conjunto primos

# Variáveis
Error_Prime = 0
FirstnextPrime = nextPrime
l1=[]

# Calculos iniciais

for k in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número nextPrime =', nextPrime, ', não é primo'
        Error_Prime = 1
        break;

# Ciclo calculo da ordem de P
```

```
    while (nextPrime  <= Last_Prime):
       F = Zmod(nextPrime);
       E = EllipticCurve(F,[1, 1]);
       P = E([0,1]);
       mP=li*P;
       Y = mP[2];

       if    Y == 0:

             l1.append(nextPrime);

       nextPrime = Pr.next(nextPrime);
       if    nextPrime == pontoSingular:
             nextPrime = Pr.next(nextPrime);
if     Error_Prime == 0:
       print 'para ', li, 'P, temos:';
       print '            foi efetuada redução mod p no intervalo de ',FirstnextPrime, 'até ', Last_Prime
       print '            foram encontrados os fatores: ', l1

#Output
Set of all prime numbers: 2, 3, 5, 7, ...
para  47403331P, temos:
           foi efetuada redução mod p no intervalo de  3 até  107
           foram encontrados os fatores:  []
```

Verificamos que os numeros $3,5,7$ não são fatores do ponto $47403331P$ e sendo $(S_{47403331} - (S_{3299} \cup S_{14369})) \neq \emptyset$, então podemos concluir que:

$$p_{47403331} > 2^3 \qquad \text{para } \ell_j = 3299 \text{ onde } j = 2$$

(c) Para $\ell_j = \ell_3$, temos que calcular $p_{\ell_i \ell_3} = p_{14369 \times 14369} = p_{206468161}$

De acordo com a definição, temos que:

$$p_{206468161} = \max(S_{206468161} - (S_{14369} \cup S_{14369}))$$

e que pelo [Poo03a, Lema 3.4], $(S_{206468161} - (S_{14369} \cup S_{14369}))$ não é um conjunto vazio, se o $\max\{14369, 14369\}$ é suficientemente grande. Na impossibilidade de calcular $206468161P$ vamos utilizar o método 2 para provar a condição 3 com recurso à redução mod p.

Programa 5.31: Geração $\mathbb{F}_p$ onde p menor 107 para o Ponto $206468161P$

```
# Parametros
nextPrime= 3                  #número p                          EDITAVEIS
Last_Prime = 107              #Fim de geração de p               EDITAVEIS
li=206468161                  #numero primo a fatorizar          EDITAVEIS
pontoSingular=31              #numero singular                   EDITAVEIS

# Definicoes
Pr =Primes(); Pr                  #definir conjunto primos

# Variáveis
Error_Prime = 0
FirstnextPrime = nextPrime
l1=[]

# Calculos iniciais

for k in range(1):
    if  is_prime(nextPrime) == False:
        print ' O número nextPrime =', nextPrime, ', não é primo'
        Error_Prime = 1
        break;

# Ciclo calculo da ordem de P


    while (nextPrime  <= Last_Prime):
       F = Zmod(nextPrime);
       E = EllipticCurve(F,[1, 1]);
       P = E([0,1]);
       mP=li*P;
       Y = mP[2];

       if    Y == 0:

             l1.append(nextPrime);

       nextPrime = Pr.next(nextPrime);
       if    nextPrime == pontoSingular:
             nextPrime = Pr.next(nextPrime);
if     Error_Prime == 0:
       print 'para ', li, 'P, temos:';
       print '            foi efetuada redução mod p no intervalo de ',FirstnextPrime, 'até ', Last_Prime
```

```
      print '           foram encontrados os fatores: ', l1

#Output
Set of all prime numbers: 2, 3, 5, 7, ...
para  206468161P, temos:
           foi efetuada redução mod p no intervalo de  3 até  107
           foram encontrados os fatores:  []
```

Verificamos que os numeros $3,5,7$ não são fatores do ponto $206468161P$ e sendo $(S_{206468161} - (S_{14369} \cup S_{14369})) \neq \emptyset$, então podemos concluir que:

$$p_{206468161} > 2^3 \qquad \text{para } \ell_j = 14369 \text{ onde } j = 3$$

Assim, a condição 3 verifica-se para $\ell_3 = 14369$

(4) $p_{\ell 14369} > 2^3 \qquad \forall \ell \in L$

Já calculámos anteriormente $L = \{2,3\}$ e $\ell_3 = 14369$

(a) Para $\ell = 2$, temos que calcular $p_{\ell\ell_2} = p_{2\times 14369} = p_{28738}$

i. Cálculo de F_{28738} (raiz quadrada do denominador coordenada x do ponto $28738P$).
ii. Cálculo de F_2 (raiz quadrada do denominador coordenada x do ponto $2P$).
iii. Cálculo de F_{14369} (raiz quadrada do denominador coordenada x do ponto $14369P$).
iv. Cálculo de F_{28738}/F_2 (Divisão de i) por ii)).
v. Cálculo de $(F_{28738}/F_2)/F_{14369}$ (Divisão de iv) por iii)).
vi. Pesquisar em $(F_{28738}/F_2)/F_{14369}$, fatores primos.

Validando previamente que $(F_{28738}/F_2)/F_{14369} > 1$, fez-se a fatorização parcial de $(F_{28738}/F_2)/F_{14369}$, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{28738} = \max(S_{28738} - (S_2 \cup S_{14369}))) > 2^3$$

É suficiente garantir que o primeiro numero primo da fatorização de $(F_{28738}/F_2)/F_{14369} > 2$. Essa condição verifica-se, como é demonstrada na rotina desenvolvida em Sage (igual à desenvolvida para condição 4 do ponto $97P$ - não reproduzida)

Programa 5.32: Verificar que p_{28738} é Maior que 2^3

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 2                             #número l                          EDITAVEIS
m = 14369                         #número m                          EDITAVEIS
i = 3;                            #valor de i para o numero li       EDITAVEIS
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0

# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break
```

```
# Ciclo calculo de lP, mP e lmP

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad

    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if     Q1lmP % RmP == 0:
           Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

# Ciclo calculo de l*P e teste condicao 5
    while   nextPrime <= Number_2i:
            Reminder = Q2lmP % nextPrime
            if   Reminder == 0:
                 I.append(nextPrime)
            nextPrime = Pr.next(nextPrime)


# Calculos Finais
Len_I = len( I );
if   Error_Prime == 0:
     if   Len_I == 0 and Q2lmP > 1:
          print 'l =', l, ', m =', m
          print 'O conjunto Slm - (Sl - Sm ) não é vazio'
          print 'p de ', lm, ' é maior que 2^', i
          print 'a pesquisa de fatorização terminou no numero primo', nextPrime
     else:
          print 'p de ', lm, ' é menor que 2^', i
if   Error_Prime == 2:
     print'Erro não previsto'

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 2 , m = 14369
O conjunto Slm - (Sl - Sm ) não é vazio
p de  28738  é maior que 2^ 3
a pesquisa de fatorização terminou no numero primo 11
```

Assim,

$p_{28738} > 2^3$

Concluímos que a condição 4 é verificada para $\ell = 2$:

(b) Para $\ell = 3$, temos que calcular $p_{\ell\ell_3} = p_{3\times 14369} = p_{43107}$

i. Cálculo de F_{43107} (raiz quadrada do denominador coordenada x do ponto $43107P$).
ii. Cálculo de F_3 (raiz quadrada do denominador coordenada x do ponto $3P$).
iii. Cálculo de F_{14369} (raiz quadrada do denominador coordenada x do ponto $14369P$).
iv. Cálculo de F_{43107}/F_3 (Divisão de i) por ii)).
v. Cálculo de $(F_{43107}/F_3)/F_{14369}$ (Divisão de iv) por iii)).
vi. Pesquisar em $(F_{43107}/F_3)/F_{14369}$, fatores primos.

Validando previamente que $(F_{43107}/F_3)/F_{14369} > 1$, fez-se a fatorização parcial de $(F_{43107}/F_3)/F_{14369}$, por pesquisa sequencial dos números primos, a partir de 2. Dado que neste caso, queremos apenas garantir, sem determinar que:

$$(p_{43107} = \max(S_{43107} - (S_3 \cup S_{14369}))) > 2^3$$

É suficiente garantir que o primeiro numero primo da fatorização de $(F_{28738}/F_2)/F_{3299} > 2$.

Programa 5.33: Verificar que p_{43107} é Maior que 2^3

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
l = 3                             #número l                          EDITAVEIS
m = 14369                         #número m                          EDITAVEIS
i = 3;                            #valor de i para o numero li       EDITAVEIS
nextPrime = 2                     #primeiro n. primo do [] cont      EDITAVEIS

# Variáveis
I = []                            #Intervalo dos números Primos Pesquisados
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
Q1lmp = 0
Q2lmp = 0

# Calculos iniciais
Number_2i = 2^i
lm = l * m

for k in range(1):
    if  is_prime(l) == False:
        print ' O número l =', l, ', não é primo'
        Error_Prime = 1
        break
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo de lP, mP e lmP

    lP = l * P
    XlP = lP[0];
    DlP = XlP.denominator();
    RlPbad = sqrt(DlP)
    while   RlPbad % 2 == 0:
            RlPbad = RlPbad / 2
    RlP = RlPbad

    mP = m * P
    XmP = mP[0];
    DmP = XmP.denominator();
    RmPbad = sqrt(DmP)
    while   RmPbad % 2 == 0:
            RmPbad = RmPbad / 2
    RmP = RmPbad

    lmP = lm * P
    XlmP = lmP[0];
    DlmP = XlmP.denominator();
    RlmPbad = sqrt(DlmP)
    while   RlmPbad % 2 == 0:
            RlmPbad = RlmPbad / 2
    RlmP = RlmPbad

    if   RlmP % RlP == 0:
         Q1lmP = RlmP / RlP
    if     Q1lmP % RmP == 0:
           Q2lmP = Q1lmP / RmP
    else:
           if   m == l:
                Q2lmP = Q1lmP
           else:
                Error_Prime =2
                break

#
    while   nextPrime <= Number_2i:
            Reminder = Q2lmP % nextPrime
            if   Reminder == 0:
                 I.append(nextPrime)
            nextPrime = Pr.next(nextPrime)

# Calculos Finais
Len_I = len( I );
if   Error_Prime == 0:
     if   Len_I == 0 and Q2lmP > 1:
          print 'l =', l, ', m =', m
          print 'O conjunto Slm - (Sl - Sm ) não é vazio'
          print 'p de ', lm, ' é maior que 2^', i
          print 'a pesquisa de fatorização terminou no numero primo', nextPrime
     else:
          print 'p de ', lm, ' é menor que 2^', i
if   Error_Prime == 2:
     print'Erro não previsto'

# OUTPUT
```

```
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
l = 3 , m = 14369
O conjunto Slm - (Sl - Sm ) não é vazio
p de  43107  é maior que 2^ 3
a pesquisa de fatorização terminou no numero primo 11
```

Assim,

$p_{43107} > 2^3$

Concluímos que a condição 4 é verificada para $\ell = 3$:

A condição 4 verifica-se para $\ell_3 = 14369$

(5) Condição 5: $|y(\ell_3 P) - 3| \leq 1/(10 \times 3)$
A condição 5 verifica-se para $\ell_3 = 14369$, com recurso a uma rotina desenvolvida utilizando o software Sage.

Programa 5.34: Verificar que $14369P$ Satisfaz a Condição 5 de ℓ_i

```
# Definicoes
E = EllipticCurve([1,1]) ; E      #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]); P                   #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Parametros
m = 14369                         #número m                          EDITAVEIS
Iindx = 3;                        #valor i para li                   EDITAVEIS


# Calculos iniciais

for k in range(1):
    if  is_prime(m) == False:
        print ' O número m =', m, ', não é primo'
        break

# Verificar se mP satisfaz condicao 5

    mP = m * P
    YmP = mP[1] - Iindx;
    if   abs(YmP) <=  (1/(10 * Iindx)):
         print 'O número primo', m, 'satisfaz condição 5 em li, para o valor de i=', Iindx
    else:
         print 'O número primo', m, 'não satisfaz condição 5 em li, para o valor de i=', Iindx

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Rational Field
(0 : 1 : 1)
Set of all prime numbers: 2, 3, 5, 7, ...
O número primo 14369 satisfaz condição 5 em li, para o valor de i= 3
```

Concluímos deste modo, confirmadas as 5 condições, que o candidato a ℓ_3, o número primo 14369 é o valor de ℓ_3 para a curva elíptica em análise.

5.10.6 Candidato a $\ell_4 > 27031$

Foi efetuada uma pesquisa, utilizando o software Sage,aos numeros primos do intervalo, $]14369, 27031]$, para encontrar um candidato que satisfizesse a condição 5:

$$|y(\ell_4 P) - 4| \leq 1/(10 \times 4)$$

Nenhum dos numeros primos do intervalo, satisfez a condição.

5.11 Conjuntos T_1, T_2

5.11.1 Conjunto T_1

Como definido previamente,

$$T_1 = S_{bad} \cup \bigcup_{i \geq 1} S_{\ell_i}$$

Construímos já o conjunto $S_{bad} = \{2\}$
Identificámos 3 elementos do conjunto ℓ_i : $97, 3299, 14369$.
Construídos anteriormente, temos:

(1) $S_{97} = \{97, 520081343, 882754619, ...\}$

(2) $S_{3299} = \{..., ...\}$ não foi obtido qualquer fator primo

(3) $S_{14369} = \{..., ...\}$ não foi obtido qualquer fator primo

Teremos então identificado alguns elementos do conjunto T_1:

$$T_1 = \{2, 97, 520081343, 882754619, ...\} \cup (S_{97} - \{97, 520081343, 882754619\}) \cup S_{3299} \cup S_{14369} \cup \cdots$$

5.11.2 Recursividade em T_1 - Algoritmo

S_{bad} por ser um conjunto finito, é recursivo. Vamos implementar um algoritmo que dado um número primo $p \notin S_{bad}$ permita decidir se $p \in \bigcup_{i \geq 1} S_{\ell_i}$.

- Cálculo da ordem de P em $\mathbb{F}_p$.
- Verificar se ordem de P pertence ao conjunto dos ℓ_i.
- Concluir se p pertence ou não a S_{ℓ_i}.

O algoritmo do cálculo da ordem de P, foi desenvolvido em Sage:

Programa 5.35: Cálculo da Ordem de P para E módulo p

```
# Parametros
p = 520081343                    #número p                          EDITAVEIS
ntimes =1000                     #numero vezes do ciclo de calculo  EDITAVEIS
li=[97,3299,14369]               #lista de elementos de li          EDITAVEIS

# Definicoes
F = Zmod(p)
E = EllipticCurve(F, [1, 1]); E   #geração da curva elíptica y^2=x^3+x+1
P = E([0,1]);                    #definir P como o ponto infinito gerador de E(Q)
Pr =Primes(); Pr                  #definir conjunto primos

# Variáveis
m = 2                            #numero para calculo mP
Error_Prime = 0                  #Primeiro e Ultimo n. primo válido se 0

# Calculos iniciais
G=E.gens()
Cardinal= E.order();

for k in range(1):
    if  is_prime(p) == False:
        print ' O número p =', p, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo da ordem de P

    for i in range(ntimes):
        mP=m*P;
        Y = mP[2]

        if  Y == 0:
            Error_Prime = 1
            print 'Fp redução mod', p
            print 'Coordenadas do ponto P', P
            print 'A ordem do ponto P é', m
            if   m in li:
                 print m, 'pertence ao conjunto li',li
                 print 'logo', p, ' Pertence ao conjunto Sli'
            else:
                 print p, 'não é fator primo de', m,'P'
            break
        m=m+1
    if  Error_Prime == 0:
        print 'aumente o número de vezes do ciclo. É insuficiente'
```

```
#Output
Elliptic Curve defined by y^2 = x^3 + x + 1 over Ring of integers modulo 520081343
Set of all prime numbers: 2, 3, 5, 7, ...
Fp redução mod 520081343
Coordenadas do ponto P (0 : 1 : 1)
A ordem do ponto P é 97
97 pertence ao conjunto li [97, 3299, 14369]
logo 520081343  Pertence ao conjunto Sli
```

Neste caso, dado que 97 pertence ao conjunto dos ℓ_i (com $97 = \ell_1$), então podemos concluir que $520081343 \in \bigcup_{i\geq 1} S_{\ell_i}$.

5.11.3 Conjunto T_2

Como vimos anteriormente,

$$T_2 = T_2^a \cup T_2^b \cup T_2^c$$

Vamos limitar o nosso estudo ao conjunto T_2^a em virtude de não nos ser exequível o cálculo de elementos de T_2^b e T_2^c dado que para calcular o max dum conjunto, teríamos de proceder à fatorização total do número.

Conjunto T_2^a

Recordemos que:

$$T_2^a = \{p_\ell : \ell \notin \{\ell_1, \ell_2, \ell_3, ...\}\}$$

Sendo $\ell_1 = 97,\ \ell_2 = 3299,\ \ell_3 = 14369$, temos:

$$T_2^a = \{p_\ell : \ell \notin \{97, 3299, 14369, ...\}\}$$

Calculemos então os números ℓ que servirão de base ao cálculo de p_ℓ:

(1) Conjunto dos numeros primos ℓ, no intervalo $[5, 97[$

[5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89]

Total elementos= 22

(2) Conjunto dos numeros primos ℓ, no intervalo $[101, 3299[$

[101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, ..., 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169, 3181, 3187, 3191, 3203, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271]

Total elementos= 437

(3) Conjunto dos numeros primos ℓ, no intervalo $[3301, 14369[$

[3301, 3307, 3313, 3319, 3323, 3329, 3331, 3343, 3347, 3359, 3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463, 3467, ..., 14083, 14087, 14107, 14143, 14149, 14153, 14159, 14173, 14177, 14197, 14207, 14221, 14243, 14249, 14251, 14281, 14293, 14303, 14321, 14323, 14327, 14341, 14347]

Total elementos= 1220

Vamos calcular alguns elementos de T_2^a.

(1) $F_{5_{xDen}} = 7^2 \times 41^2$
$S_5 = \{7, 41\}$
$p_5 = \max\ S_5 = \{41\}$

(2) $F_{7_{xDen}} = 11^2 \times 67^2 \times 127^2$
$S_7 = \{11, 67, 127\}$
$p_7 = \max\ S_7 = \{127\}$

(3) $F_{11_{xDen}} = 419^2 \times 5124054251^2$
$S_{11} = \{419, 5124054251\}$
$p_{11} = \max\ S_{11} = \{5124054251\}$

(4) $F_{13_{xDen}} = 240710769046691137^2$
$S_{13} = \{240710769046691137\}$
$p_{13} = \max\ S_{13} = \{240710769046691137\}$

(5) $F_{17_{xDen}} = 4091^2 \times 82837^2 \times 3017341^2 \times 391377210997453^2$
$S_{17} = \{4091, 82837, 3017341, 391377210997453\}$
$p_{17} = \max\ S_{17} = \{391377210997453\}$

(6) $F_{19_{xDen}} = 1409^2 \times 14401^2 \times 84920920247437898904712836223 9^2$
$S_{19} = \{1409, 14401, 8492092024743789890471283622 39\}$
$p_{19} = \max\ S_{19} = \{8492092024743789890471283622 39\}$

(7) $F_{23_{xDen}} = 5824307669^2 \times 2892229819630428638773690149186718411931792 93^2$
$S_{23} = \{5824307669, 2892229819630428638773690149186718411931792 93\}$
$p_{23} = \max\ S_{23} = \{2892229819630428638773690149186718411931792 93\}$

(8) $F_{29_{xDen}} = 53^2 \times 14629^2 \times 135221^2 \times 91006007^2 \times 228030767^2 \times 14205867207973897^2 \times$
$331781320755651251616786292499255804307725 9^2$
$S_{29} = \{53, 14629, 135221, 91006007, 228030767, 14205867207973897,$
$331781320755651251616786292499255804307725 9\}$
$p_{29} = \max\ S_{29} = \{331781320755651251616786292499255804307725 9\}$

(9) $F_{31_{xDen}} = 17351993270217672 \times 10512112416476383^2 \times$
$102955436647782269011144467834885700347239364872940347399950304413847^2$
$S_{31} = \{17351993270217672, 10512112416476383,$
$102955436647782269011144467834885700347239364872940347399950304413847\}$
$p_{31} = \max\ S_{31} = \{102955436647782269011144467834885700347239364872940347399950304413847\}$

Teremos então identificado alguns elementos do conjunto T_2^a:

$T_2^a = \{41,\ 127,\ 5124054251,\ 391377210997453,\ 240710769046691137,\ 84920920247437898904712836223 9,$
$331781320755651251616786292499255804307725 9,\ 2892229819630428638773690149186718411931792 93,$
$102955436647782269011144467834885700347239364872940347399950304413847,\ ...\}$

Logo, estão identificados alguns elementos do conjunto T_2:

$T_2 = \{41,\ 127,\ 5124054251,\ 391377210997453,\ 240710769046691137,\ 84920920247437898904712836223 9,$
$331781320755651251616786292499255804307725 9,\ 2892229819630428638773690149186718411931792 93,$
$102955436647782269011144467834885700347239364872940347399950304413847,\ ...\}$

5.11.4 Recursividade em T_2^a - Algoritmo

Vamos implementar um algoritmo que dado um número primo $p \notin S_{bad}$ permita decidir se $p \in T_2^a$.
Dado p, o algoritmo tem os seguintes passos:
1) Cálculo do cardinal de $\mathbb{F}_p$.
2) Fatorizar o cardinal de $\mathbb{F}_p$.
3) Verificar se $p = p_\ell$ para algum ℓ que seja fator do cardinal.

O algoritmo para cálculo do cardinal de $\mathbb{F}_p$ e fatorização foi desenvolvido em Sage:

Programa 5.36: Cálculo Cardinal $\mathbb{F}_p$

```
# Parametros
p = 41                              #número p                          EDITAVEIS
ntimes =10                          #numero vezes do ciclo de calculo  EDITAVEIS
LnotLi=[5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53]        #EDITAVEIS

# Definicoes
F = Zmod(p)
E = EllipticCurve(F, [1, 1]); E   #geração da curva elíptica y^2=x^3+x+1
Pr =Primes(); Pr                  #definir conjunto primos

# Variáveis
Error_Prime = 0                   #Primeiro e Ultimo n. primo válido se 0
nextPrime = 2
Pl=[]

# Calculos iniciais
G=E.gens()
Cardinal= E.order();

for k in range(1):
    if  is_prime(p) == False:
        print ' O número p =', p, ', não é primo'
        Error_Prime = 1
        break

# Ciclo calculo da ordem de P
    print 'Coordenadas do ponto gerador G', G
    print 'O cardinal de Fp é: ', Cardinal

    for i in range(ntimes):
        Reminder =Cardinal % nextPrime
        if  Reminder == 0:
            if  nextPrime in LnotLi:
                Error_Prime = 1
                Pl.append(nextPrime)
                print nextPrime,'é fator primo do cardinal', Cardinal
                print nextPrime,'existe no conjunto dos l que não estão em li', LnotLi
        nextPrime= Pr.next(nextPrime)
    if  Error_Prime == 0:
        print 'aumente o número de vezes do ciclo. É insuficiente'
    if  Error_Prime == 1:
        print 'Verique para qual dos números primos',Pl,'se tem pl=',p

# OUTPUT
Elliptic Curve defined by y^2 = x^3 + x + 1 over Ring of integers modulo 41
Set of all prime numbers: 2, 3, 5, 7, ...
Coordenadas do ponto gerador G [(9 : 40 : 1)]
O cardinal de Fp é:  35
5 é fator primo do cardinal 35
5 existe no conjunto dos l que não estão em li [5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53]
7 é fator primo do cardinal 35
7 existe no conjunto dos l que não estão em li [5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53]
Verique para qual dos números primos [5, 7] se tem pl= 41
```

Neste caso, dado que $p_5 = \max\ S_5 = 41 = p$, como já calculado anteriormente, podemos concluir que $41 \in T_2^a$.

5.12 Conjunto S

Sabemos pela [Poo03a, Proposição 5.2], que $E'(\mathbb{Z}[S^{-1}])$ é união de $\{\pm \ell_i P : i \geq 1\}$ e algum subconjunto dum conjunto finito $\{rP : \ r | \prod_{\ell \in L} \ell^{a_\ell - 1}\}$.

Vamos calcular os valores de r que satisfazem na curva elíptica em estudo:

$$\{rP : \ r | \prod_{\ell \in L} \ell^{a_\ell - 1}\}$$

Determinámos anteriormente o conjunto L e os valores de a_ℓ para os elementos do conjunto L.

$L = \{2,3\}$
$a_2 = 2$
$a_3 = 2$

Assim, $\prod_{\ell \in L} \ell^{a_\ell - 1} = 2^{2-1} \times 3^{2-1} = 6$.Temos então determinado, os valore possíveis para r: $r \in \{2,3,6\}$
Já calculámos anteriormente, os valores de S_n, para os pontos $2P$, $3P$ e $6P$:

$S_2 = \emptyset$
$S_3 = \emptyset$
$S_6 = \{13, 47\}$

Sendo $\mathcal{P}$ o conjunto dos números primos, o conjunto S é qualquer subconjunto de $\mathcal{P}$ que contém o conjunto T_1 e é disjunto de T_2.

Desta forma, S_{bad} contribui com o número primo 2, o conjunto finito $\{rP\}$ com os números primos $13, 47$ e o conjunto $\{\pm\ell_i P\}$ com os restantes números primos.

Enumeremos, para a curva elíptica em estudo, alguns elementos de S:

$$S = \{2, 13, 47, 97, 520081343, 882754619, \} \cup (S_{97} - \{97, 520081343, 882754619\}) \cup S_{3299} \cup S_{14369} \cup \cdots$$

5.13 Considerações Finais

Os objetivos propostos no início deste estudo prático foram inteiramente alcançados.
Desenvolveram-se os algoritmos necessários à obtenção dos dados e utilizaram-se os comandos disponibilizados pelo Sage, por forma a obter a informação necessária.
Confirmou-se que a curva elíptica selecionada obedecia às condições definidas por Poonen. Verificou-se para diversos casos concretos a correcção do [Poo03a, Corolário 3.3] e [Poo03a, Lema 3.4]. Utilizando os algoritmos desenvolvidos, obtiveram-se exemplos para os elementos iniciais dos conjuntos T_1, T_2 e S estudados no artigo [Poo03a].
Os dois maiores obstáculos encontrados durante o cálculo foram o vasto número de digitos quer do numerador quer do denominador das coordenadas de nP e a complexidade algorítmica da fatorização em números primos do denominador da coordenada x.
Uma vez que o primeiro obstáculo é um facto incontornável da natureza do problema, a solução encontrada foi um investimento de mais de 1000 horas nos cálculos realizados. Os recursos de computador disponíveis para este estudo foram inferiores aos desejados o que se refletiu na quantidade de valores obtidos. Mais velocidade de processamento, mais capacidade de memória, um processamento multi-thread, levar-nos-ia a construir subconjuntos mais populados.
Relativamente à fatorização em primos de números com muitos dígitos, o facto do algoritmo de fatorização Sage não permitir fatorização parcial, levou à utilização do algoritmo geralmente denominado por "fatorização por procura direta". Embora a programação do mesmo tenha levado a uma solução com complexidade sub-óptima, assegurou a flexibilidade necessária à realização do trabalho. Mesmo quando não foi possível realizar a fatorização de forma a validar as condições dos candidatos $\ell_i P$ desenvolveram-se estratégias para atingir este objetivo sem ser necessário recorrer a esta fatorização.
Apesar dos condicionalismos encontrados na implementação dos algoritmos, foi possível construir um subconjunto do conjunto S, onde para o maior ℓ_i obtido, o denominador da coordenada x tem 10^7 digitos.
Não é de excluir a possibilidade de existirem resultados mais avançados da teoria dos números que permitam obter algoritmos mais eficazes para os cálculos que implementamos. No entanto tais resultados saem fora do âmbito desta tese.

Bibliografia

[Coo04] S. Barry Cooper. *Computability Theory*. Chapman & Hall, 2004.

[DPR61] Martin Davis, Hilary Putman, and Julia Robinson. The decision problem for exponencial diophantine equations. *Ann. of Math.*, 74(2):425–436, 1961.

[Hil02] David Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8(10):437–479, 1902.

[Mat70] Yuri V. Matijasevic. The diopnantiness of enumerable sets. *Dokl. Akad Nauk SSSR*, 191:279–282, 1970.

[Poo03a] Bjorn Poonen. Hilbert ' s tenth problem and mazur´s conjecture for large subrings of $\mathbb{Q}$. *Journal of the American Mathematical Society*, 16(4):981–990, 2003.

[Poo03b] Bjorn Poonen. Hilberts tenth problem over rings of number-theoretic interest. 2003. Notes for the lectures at the Arizona Winter School on Number theory and logic 2003 `http://www-math.mit.edu/~poonen/`.

[Poo06] Bjorn Poonen. Hilbert's tenth problem. 2006. Slides for the talk at the MSRI introductory workshop on rational and integral points on higher-dimensional varieties 2006 `http://www-math.mit.edu/~poonen/`.

[Row14] Todd Rowland. Transcendence degree, 2014. MathWorld–Wolfram Web Source `http://mathworld.wolfram.com/TranscendenceDegree.html`.

[Shl07] Alexandra Shljapentokh. *Hilbert's Tenth Problem - Diophantine Classes and Extensions to Global Fields*. Cambridge - University Press, 2007.

[Sil08] Joseph H. Silverman. *The Arithmetic of Elliptic Curves, Second Edition, Graduate Texts in Mathematics*. Springer Verlag, 2008.

[ST10] Joseph H. Silverman and John Tate. *Rational Points on Elliptic Curves Undergraduate Texts in Mathematics*. Springer Verlag, 2010.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry . 2nd ed*. Berkeley, CA: University of California Press, 1951.

Lista de Figuras

Lista de Programas