Universität Zürich

Master's Thesis

Cayley Graphs with Large Girth and Constructions of LDPC Codes

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Mathematics

Author:
Luana Kurmann
19-745-231

Supervisor:
Prof. Dr. Joachim Rosenthal

Universität Zürich[UZH]

Institute of Mathematics
Universität Zürich
3rd February 2025

Abstract

In this thesis, we investigate a particular family of graphs with large girth and look at their applications to low-density parity-check (LDPC) codes. Specifically, we study a family of Cayley graphs with large girth over finite fields of prime order, constructed by Margulis in 1982. We analyse Margulis' lower bound for the girth of these graphs and compute their exact girths for several prime numbers. Our results show that the actual girth exceeds the bound by an average of about 87% in all cases studied. We show an improvement to Margulis' bound which leads to an average error rate of about 57%. Additionally, we present further observations which suggest that an even better bound may still exist. Furthermore, we show how LDPC codes can be constructed based on these graphs and simulate their error-correction performance over an additive white Gaussian noise channel.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Joachim Rosenthal for introducing me to the fascinating areas of coding theory and cryptography and for giving me the opportunity to write my Master's thesis in coding theory. He always had time for my questions and I'm extremely thankful for his invaluable support, not only regarding my Master's thesis, but also on a personal level.
Secondly, I would also like to thank Dr. Jessica Bariffi for supporting me during this time and especially for proofreading my thesis. I'm extremely grateful for her feedback, her advice and that she always was available when I needed help.
I would also like to thank Michael Schaller for checking my Python code and Abigail Sutton for proofreading my Master's thesis. Moreover, I would like to thank Prof. Pascal O. Vontobel and Ganeshaanand (Rishi) Balasubramanian for helping me with the simulation of the LDPC codes.

I also want to thank my fellow students who made these years to an unforgettable experience. Especially, I want to thank Vanessa and Elia for being at my side from day one of my studies. It was a pleasure to attend the lectures together with you and I enjoyed the time we spent together. Thank you for always supporting me and I feel extremely honoured to call you my friends.
Last but not least, I want to thank my family for their support during this time and that they always believed in me.

Contents

1 Introduction **1**

2 Introduction to Graph Theory **3**

2.1 The Adjacency Matrix and the Spectral Gap 4

2.2 Ramanujan Graphs 7

2.3 Bounds on Girth 7

2.4 Cayley Graphs 10

3 Introduction to Coding Theory **13**

3.1 Linear Codes and the Hamming Distance 13

3.2 Bounds on Codes 15

3.3 Low-Density Parity-Check Codes 17

3.4 Sum-Product Algorithm 18

3.4.1 Probability-Domain Sum-Product Algorithm 20

3.4.2 Log-Domain Sum-Product Algorithm 22

4 Constructions of Cayley Graphs with Large Girth **25**

4.1 Construction by Margulis 25

4.1.1 Large Girth Property 26

4.1.2 Other Matrices 29

4.1.3 Calculation of Exact Girth 32

4.2 Construction by Lubotzky, Phillips and Sarnak 42

4.2.1 Quaternions and Sums of Four Squares 42

4.2.2 Construction of the Ramanujan Graphs $X_{p,q}$ 44

5 Construction and Simulation of Margulis' LDPC Codes **49**

5.1 Margulis' Construction of LDPC Codes 49

5.2 Simulation 51

6 Conclusion **55**

7 Appendix **56**

7.1 The Python Code for Calculating the Exact Girth 56

7.2 Results Girth Calculation 62

7.3 Matlab and Sage Codes used for the Simulation of LDPC Codes 63

7.3.1 Computing a Parity-Check Matrix 63

7.3.2 Simulation of LDPC Codes 63

1 Introduction

Graphs with large girth as well as expander graphs play an important role, not only in graph theory, but also in other areas such as for example in network theory or coding theory. Indeed, we can consider a graph $X = (V, E)$ to be a network of vertices where information is passed along the edges. For example, this network could be a group of people who write letters to each other to distribute information. The people of this group correspond to the vertices of the network and there is an edge between two people if and only if they know each other's address. Assume that there is a person in this group who wants to share some news with all the other people. This person writes a letter to all the people they know the address of. The recipients of this letter then forward the message to all the other people they know the address of and so on. This network has a large girth if it takes a long time until someone who already received the letter receives it again. It is additionally a "good" expander if the time until everyone receives the letter is as short as possible, i.e. most people know many addresses and hence can distribute the letter fast. Thus, these properties ensure that the information is spread as efficiently as possible.
More precisely, consider any connected network consisting of vertices and edges and imagine that there is a vertex v_0 from which new information is spread to all the other vertices. In order for such a network to be "good", the two mentioned properties are particularly desirable. The graph X should not have small cycles, respectively its girth should be as large as possible, and it should be a good expander. The latter means that for any subset of vertices $S \subseteq V$ with $|S| \leq \frac{|V|}{2}$, the number of edges connecting S to $V \setminus S$ is large with respect to the number of vertices in S. In other words, assume that all vertices in a set S send a new message to their neighbouring vertices in $V \setminus S$. Then, the number of vertices in $V \setminus S$ which receive this message should ideally be larger than the number of vertices in S. We say that such a graph (i.e. a graph which is a "good" expander) is highly connected and therefore, information is passed along quickly.
However, these two properties are somehow contradictory. A large girth implies that the corresponding graph is only sparsely connected whereas expander graphs are highly connected.

As already mentioned, graphs with large girths and expander graphs are also important in coding theory, especially when talking about low-density parity-check (LDPC) codes. LDPC codes were introduced by Robert G. Gallager in 1962 [6]. These codes are binary linear block codes whose parity-check matrices consist only of a few nonzero entries. However, for about 35 years after Gallager proposed them, LDPC codes were only sparsely researched. One exception is R. Michael Tanner who presented a graphical representation of LDPC codes (called Tanner graphs) in 1981 [18]. A Tanner graph is a (bipartite) graph whose nodes are divided into two parts, the variable nodes (which represent the code symbols) and the check nodes (which represent the code constraints). One year later, G. A. Margulis presented a construction of LDPC codes based on a family of graphs with large girth [11].

The representation of LDPC codes as Tanner graphs is especially useful for iterative decoding algorithms such as the sum-product algorithm (SPA). For this, let us think of a (simplified) situation where we want to send a message y to somebody (for example, we communicate via the Internet or we try to speak to this person on the phone). This message is then converted to a codeword c which contains several redundant bits. During its transmission through a (communication) channel, the codeword might be changed as errors might occur. More precisely, some bits of the codeword might be changed which results in receiving a different, i.e. "wrong", word (for example, we speak to somebody on the phone but have a bad connection such that the other person is not able to understand everything correctly). The receiver then tries to retrieve the original codeword from the received message. Finally, the redundancy is removed to obtain the originally sent message (for more details see for example [17]).
Retrieving the original codeword can be done by an iterative decoder. Such a decoder for LDPC codes sends messages, which contain probabilistic information about the values of the bit nodes, along the edges between the variable and check nodes. Consequently, the corresponding Tanner graph should not have small cycles. Otherwise, the decoder gets trapped in these cycles which makes it difficult to find the correct codeword (this will be explained in more detail in Section 3.4). In addition, it is desirable that the graph is a good expander. This ensures that many messages are sent along the edges which increases the probability that the decoder finds the original codeword.

In Chapter 2, we give a short introduction to graph theory. We introduce the most important definitions and theorems and discuss some bounds on the girth. Furthermore, we introduce Cayley graphs which will be essential when it comes to the explicit construction of families of graphs with large girth.

In Chapter 3, we give a short overview of coding theory. In particular, we consider LDPC codes and their graphical representation and introduce the Probability-Domain SPA Decoder as well as the Log-Domain SPA Decoder.

In Chapter 4, we describe two different constructions of families of graphs with large girth where we lay the focus on the construction by G. A. Margulis [11]. We discuss Margulis' lower bound on the girth of these graphs and present a comparison of the bound with the exact girth for some of these graphs. Following this, we show an improvement of this bound and present new observations which indicate that a further improvement might be possible.

Finally, in Chapter 5, we describe a construction of LDPC codes based on Margulis' family of graphs with large girth. In addition, we present results of simulations of their error-correction performance over an additive white Gaussian noise channel using the sum-product algorithm.

2 Introduction to Graph Theory

We start by introducing some important definitions and results from graph theory which will be needed later. This chapter is mainly based on [3, Overview and Chapter 1].

A *graph* X consists of vertices (also called nodes) and edges between these vertices. A distinction is made between directed and undirected graphs. A *directed* graph is a graph where the edges have a direction, i.e. an edge going from a vertex v to a vertex $\tilde{v}$ is different from an edge going from $\tilde{v}$ to v. The edges of an *undirected* graph do not have directions.
Throughout this thesis, we will only consider undirected graphs whose formal definition is the following.

Definition 2.1. [3, Overview] An *undirected graph* $X = (V, E)$ consists of

(i) V, a set of vertices (possibly infinite),

(ii) E, a set of edges where $E \subseteq \{\{v, \tilde{v}\} \mid v, \tilde{v} \in V\}$.

Note that E may contain elements of the form $\{v, v\}$, $v \in V$; such edges are called *loops*. However, we will only consider finite graphs (meaning V is finite) without loops that are *simple* (i.e. there is at most one edge joining two vertices).

Example 2.2. Figure 1 shows an example of an undirected graph X with vertex set $V = \{1, 2, 3, 4, 5\}$ and edges $E = \{\{1, 2\}, \{2, 4\}, \{2, 5\}, \{4, 5\}, \{3, 4\}\}$. Clearly, X is finite, simple and has no loop.

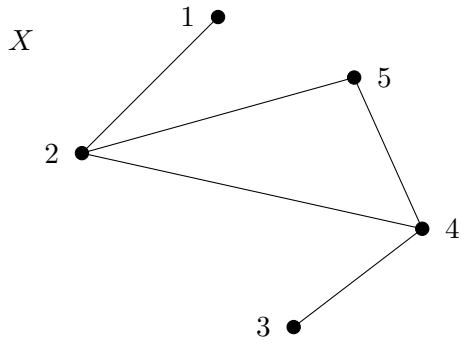

Figure 1: Example of an undirected graph.

Definition 2.3. [3, Overview] Let X be a graph. Then, we define

(i) A *path* of length $k \in \mathbb{N}$ in X is a sequence $v_0 v_1 \dots v_k$ of vertices where, for all $0 \leq i < k$, v_i is adjacent to v_{i+1} (i.e. $\{v_i, v_{i+1}\}$ is an edge of X).

(ii) X is said to be *connected* if every two vertices can be joined by a path.

Definition 2.4. [3, Section 1.1] Let $V = \{v_0, v_1, \dots\}$ be the set of vertices of a graph X. Then, the *adjacency matrix* of the graph X is a matrix A indexed by the vertices in V. More precisely, $A = (A_{ij})$ where

$$A_{ij} := \begin{cases} 1, & \text{if } v_i \text{ and } v_j \text{ are joined by an edge} \\ 0, & \text{otherwise.} \end{cases}$$

Note that A completely determines X and that A is, by definition, a square matrix. Moreover, A is symmetric because X is undirected. It also holds that X has no loops if and only if $A_{ii} = 0$ for every $v_i \in V$.

Example 2.5. The adjacency matrix of the graph X in Figure 1 is given by

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

We see, in Figure 1, that for example vertex 2 is adjacent to the vertices $1, 4$ and 5 and indeed $A_{21} = A_{24} = A_{25} = 1$.

Definition 2.6. [3, Definition 1.1.1] Let $k \geq 2$ be an integer. We say that a graph X is *k-regular* if for every $v_i \in V$ it holds that $\sum_{v_j \in V} A_{ij} = k$.

Note that if X is k-regular and has no loops, then every vertex has exactly k adjacent vertices. The *degree* of a vertex v (denoted by $\deg(v)$) is the number of edges that are incident to v. Therefore, each vertex of a k-regular graph has degree k. If X has no loops, the degree also indicates the number of adjacent vertices of v. The sum of the degrees of all vertices in a graph is always even, as Lemma 2.7 shows.

Lemma 2.7. [13, p. 12] *Let $X = (V, E)$ be any graph. Then,*

$$\sum_{v \in V} \deg(v) = 2 \cdot |E|.$$

Proof. This follows directly from the definition of degree. □

2.1 The Adjacency Matrix and the Spectral Gap

We assume in this section that X is a finite and simple graph without loops on $n := |V|$ vertices. Consequently, the adjacency matrix A is a symmetric $(n \times n)$-matrix which, thus, has n real eigenvalues, counted with multiplicities. The set of all eigenvalues of A is called the *spectrum* of X. Let

$$\mu_0 \geq \mu_1 \geq \cdots \geq \mu_{n-1}$$

denote these eigenvalues in decreasing order. In case it is not clear from the context, we write $\mu_i(X)$ to denote the eigenvalues corresponding to the graph X. Proposition 2.8 describes some spectral properties of A (its proof can be found in [3, p. 9-10]).

Proposition 2.8. [3, Proposition 1.1.2] *Let X be a finite k-regular graph with n vertices. Then, the following properties hold.*

(i) $\mu_0 = k$,

(ii) $|\mu_i| \leq k$ *for* $1 \leq i \leq n-1$,

(iii) μ_0 *has multiplicity* 1 *if and only if* X *is connected.*

Under some conditions, we can also exactly determine the smallest eigenvalue μ_{n-1}. This holds if X is bipartite which is defined in the following way.

Definition 2.9. [3, Definition 1.1.3] A graph $X = (V, E)$ is *bipartite* or *bicolourable* if there exists a partition $V = V_+ \sqcup V_-$ of the vertices such that for any two vertices x, y with $A_{xy} \neq 0$, if $x \in V_+$ (respectively V_-), then $y \in V_-$ (respectively V_+).

The term *bicolourable* comes from the fact that if X is bipartite, then its vertices can be painted with two different colours such that no two adjacent vertices have the same colour.

Proposition 2.10. [3, Proposition 1.1.4] *Let X be a connected, k-regular graph on n vertices. Then, the following are equivalent.*

(i) X *is bipartite,*

(ii) the spectrum of X *is symmetric about* 0,

(iii) $\mu_{n-1} = -k$.

The proof of Proposition 2.10 can be found in [3, p. 11].
We already saw that the largest positive eigenvalue of any connected, finite, k-regular graph X is $\mu_0 = k$. If (and only if) in addition X is bipartite, then the most negative eigenvalue is $\mu_{n-1} = -k$. These two eigenvalues (if X is not bipartite, only μ_0 is meant) are therefore called the *trivial* eigenvalues of X.

Definition 2.11. [3, p. 11] Let X be a connected, k-regular graph. The difference $k - \mu_1 = \mu_0 - \mu_1$ is called the *spectral gap* of X.

Definition 2.12. [3, Section 1.2 and Definition 1.2.1] Given a graph $X = (V, E)$.

(i) For a subset $F \subseteq V$, we define the *boundary* ∂F of F as the set of edges of X connecting F to $V \setminus F$.

(ii) The *isoperimetric constant* or *expanding constant* of X is

$$h(X) := \inf \left\{ \frac{|\partial F|}{\min\{|F|, |V \setminus F|\}} \,\middle|\, F \subseteq V,\, 0 < |F| < +\infty \right\}.$$

Note that $\partial F = \partial(V \setminus F)$ and therefore, as we assume $|V| = n$, it holds that

$$h(X) = \min\left\{ \frac{|\partial F|}{|F|} \,\middle|\, F \subseteq V,\, 0 < |F| \leq \frac{n}{2} \right\}.$$

We can also view X as a network where in one unit of time information is sent by one vertex to all its neighbouring vertices. In the next unit of time, all these vertices forward the information to their adjacent vertices and so on. In this situation, we want that the information spreads as fast as possible, i.e. that X is a "good" network. The expanding constant $h(X)$ measures how fast information is spread, so it is a measure for the connectivity of a network.

Definition 2.13. [3, Definition 1.2.2] Let $(X_m)_{m\geq 1}$ be a family of finite, connected, k-regular graphs with $|V_m| \overset{m\to\infty}{\longrightarrow} \infty$. We say that $(X_m)_{m\geq 1}$ is a *family of expanders* if there exists an $\varepsilon > 0$ such that $h(X_m) \geq \varepsilon$ for all $m \geq 1$.

Theorem 2.14 shows the connection between the spectral gap and the expanding constant (its proof can be found in [3, p. 13-17]). In addition, it gives a necessary and sufficient condition for a family of graphs to be a family of expanders.

Theorem 2.14. [3, Theorem 1.2.3] *Let X be a finite, connected, k-regular graph without loops. Then*

$$\frac{k - \mu_1}{2} \leq h(X) \leq \sqrt{2k(k - \mu_1)}. \tag{1}$$

Theorem 2.14 allows us to characterise a family of expanders via their spectral gap.

Corollary 2.15. [3, Corollary 1.2.4] *Let $(X_m)_{m\geq 1}$ be a family of finite, connected, k-regular graphs without loops such that $|V_m| \overset{m\to\infty}{\longrightarrow} \infty$. The family $(X_m)_{m\geq 1}$ is a family of expanders if and only if there exists $\varepsilon > 0$ such that $k - \mu_1(X_m) \geq \varepsilon$ for every $m \geq 1$.*

Proof. Let us first assume that $(X_m)_{m\geq 1}$ is a family of expanders. Hence, there exists $\tilde{\varepsilon} > 0$ such that $h(X_m) \geq \tilde{\varepsilon}$ for every $m \geq 1$. Then, choose

$$\varepsilon := \frac{\tilde{\varepsilon}^2}{2k} > 0$$

and use the right part of Inequality (1) to obtain, for every $m \geq 1$,

$$k - \mu_1(X_m) \geq \frac{h(X_m)^2}{2k} \geq \frac{\tilde{\varepsilon}^2}{2k} = \varepsilon.$$

For the other direction, we assume that there exists $\varepsilon > 0$ such that $k - \mu_1(X_m) \geq \varepsilon$ for every $m \geq 1$. We then choose $\tilde{\varepsilon} := \varepsilon/2 > 0$ and again use inequality (1) to obtain, for every $m \geq 1$,

$$h(X_m) \geq \frac{k - \mu_1(X_m)}{2} \geq \frac{\varepsilon}{2} = \tilde{\varepsilon},$$

which implies that $(X_m)_{m\geq 1}$ is a family of expanders. □

2.2 Ramanujan Graphs

We saw just above in Theorem 2.14 and Corollary 2.15 that the spectral gap should be as large as possible to ensure that a family of graphs is a "good" family of expanders. However, this spectral gap cannot be arbitrarily large.

Theorem 2.16. [8, Section 1 and 4] *Let $(X_m)_{m\geq 1}$ be a family of connected, k-regular, finite graphs with $|V_m| \overset{m\to\infty}{\longrightarrow} +\infty$. Denote for every m by $\mu(X_m)$ the absolute value of the (in absolute value) largest nontrivial eigenvalue of X. Then,*

$$\liminf_{m\to\infty} \mu(X_m) \geq 2\sqrt{k-1}.$$

Remark 2.17. In [3, Section 1.3] Theorem 2.16 is split into two parts where the asymptotic upper and lower bound of the nontrivial eigenvalues are treated separately. However, for our purposes, Theorem 2.16 is sufficient (its proof can be found in [8, p. 269]).

In order for the spectral gap to be large, μ_1 should be as small as possible. According to Theorem 2.16, we can define a type of graphs which achieves the biggest possible spectral gap and which are therefore the best possible expander graphs.

Definition 2.18. [3, Definition 1.3.4] A finite, connected, k-regular graph X is a *Ramanujan graph* if for every nontrivial eigenvalue μ of X, we have $|\mu| \leq 2\sqrt{k-1}$.

2.3 Bounds on Girth

A *cycle* (or *circuit*) in a graph $X = (V, E)$ is a path $v_1 v_2 \dots v_k$ where v_1 and v_k coincide. We are not only interested in graphs with large spectral gaps but also in those which additionally do not have short cycles. This motivates the next definitions.

Definition 2.19. [3, Definition 1.3.2] The *girth* of a connected graph X, denoted by $g(X)$, is the length of the shortest cycle in X. We say that $g(X) = \infty$ if X has no cycles. In this case, X is called a *tree*.

Definition 2.20. [3, Definition 0.12.12] Let $(X_m)_{m\geq 1}$ be a family of connected, k-regular, finite graphs with $|V_m| \overset{m\to\infty}{\longrightarrow} \infty$. This family has *large girth* if

$$g(X_m) \geq (C + o(1)) \cdot \log_{k-1}(|V_m|)$$

for some constant $C > 0$ and for all $m \geq 1$. The term $o(1)$ represents a quantity which goes to 0 for $m \to \infty$.

Lemma 2.21 shows that the constant C in Definition 2.20 cannot be arbitrarily large as $C \leq 2$.

Lemma 2.21. [20, Section 4.3.4] *Let X be a connected, k-regular graph. Then, it holds that $0 < g(X) \leq (2 + o(1)) \log_{k-1}(|V|)$.*

Proof. For simplicity, let $g := g(X)$. Note that the girth is trivially greater than 0. We say that a path of length $l \in \mathbb{N}$ without backtracking is a path $v_0 v_1 \dots v_l$ in X such that $v_{i-1} \neq v_{i+1}$ for all i. The upper bound now follows from a short counting argument. For this, assume that we have two paths without backtracking of length smaller than $\left\lfloor \frac{g-1}{2} \right\rfloor$ which both start at the same vertex. Clearly, they cannot end at the same vertex as then there would exist a cycle of length smaller than $2 \cdot \left\lfloor \frac{g-1}{2} \right\rfloor \leq g - 1 < g$.

Hence, all paths of length at most $\left\lfloor \frac{g-1}{2} \right\rfloor$ which start at the same vertex must end at different vertices. In other words, the number of all these paths is at most $|V|$. Furthermore, we can count the number of such paths by recalling that X is k-regular. Hence, starting a path at a vertex v_0, there are k possibilities for the second vertex v_1 of this path. As backtracking is not allowed, there are then $k-1$ possibilities for each further vertex v_i with $i > 1$. This leads to the following inequality

$$k(k-1)^{\lfloor \frac{g-1}{2} \rfloor - 1} \leq |V|,$$

which can be simplified to

$$(k-1)^{\lfloor \frac{g-1}{2} \rfloor} \leq |V|.$$

Finally, this leads to

$$g - 2 \leq 2 \log_{k-1}(|V|)$$

and hence

$$g \leq 2 \log_{k-1}(|V|) + 2 = \left(2 + \frac{2}{\log_{k-1}(|V|)}\right) \cdot \log_{k-1}(|V|) = (2 + o(1)) \log_{k-1}(|V|)$$

which concludes the proof. □

Another well-known bound is the Erdős-Sachs bound. In 1963, Erdős and Sachs [4] found an upper and lower bound for the following quantity

$$f(k, g) := \min\{|V| \mid X = (V, E) \text{ is } k\text{-regular and } g(X) \geq g\}.$$

These bounds are presented in Proposition 2.22.

Proposition 2.22. [4, Satz 1] *For all $k \geq 2$ and $g \geq 3$, it holds*

$$1 + k \sum_{j=0}^{\lfloor \frac{g-3}{2} \rfloor} (k-1)^j \leq f(k, g) \leq 4 \sum_{j=1}^{g-2} (k-1)^j. \tag{2}$$

We give here an outline of the proof of the right inequality, following [4, p. 251-253]. For this, note that for $g = 3$, it holds $f(k, 3) = k + 1$. So, the bound is fulfilled in this case. To show that the bound also holds for $g > 3$, we first prove the following, slightly stronger result.

Proposition 2.23. [4, p. 252] *Let $k \geq 2$, $g \geq 4$ and $m \geq 2\sum_{j=1}^{g-2}(k-1)^j$. Then, there exists a k-regular graph $X^{(k)}$ with $2m$ vertices and $g(X^{(k)}) \geq g$.*

Proof. This statement is shown by induction on k. The case $k = 2$ is trivial ($X^{(2)}$ is a circle of length $2m$), so we assume that the statement holds for $k-1$. More precisely, we assume that there exists a $(k-1)$-regular graph $X^{(k-1)}$ with $2m$ vertices and girth $g(X^{(k-1)}) \geq g$. Let $X_{\max} = (V_{\max}, E_{\max})$ be a graph with $2m$ vertices such that

(i) $k - 1 \leq \deg(v) \leq k$ for all $v \in V_{\max}$,

(ii) $g(X_{\max}) \geq g$,

(iii) the number of edges of $X_{\max}$ is maximal among all graphs fulfilling (i) and (ii).

Such a graph $X_{\max}$ exists as $X^{(k-1)}$ fulfils (i) and (ii). The last step is now to show that $X_{\max}$ is a k-regular graph. This can be done by proving that there exists at most one vertex $v \in V_{\max}$ with $\deg(v) = k-1$ (see [4, p. 252-253]). Having shown this, we assume by contradiction that there exists a vertex $\tilde{v} \in V_{\max}$ of degree $k-1$.
Recall Lemma 2.7 and note that this implies that the number of vertices with odd degree must always be even. If k was even, we would have exactly one vertex, namely $\tilde{v}$, of odd degree which is not possible. If k was odd, then there would be exactly $2m-1$ vertices of odd degree k which is not possible either. Therefore, $X^{(k)} := X_{\max}$ is a k-regular graph which concludes the proof. □

Proof of Proposition 2.22. We note that Proposition 2.23 implies that $f(k,g) \leq 2m$ for any $m \geq 2\sum_{j=1}^{g-2}(k-1)^j$. In particular, this holds for $m = 2\sum_{j=1}^{g-2}(k-1)^j$ which finally leads to

$$f(k,g) \leq 2m = 4\sum_{j=1}^{g-2}(k-1)^j.$$

□

Note that by rearranging the terms of the right inequality of (2), we obtain that there exist families of graphs $(X_m)_{m\geq 1}$ with girth

$$g(X_m) \geq (1 + o(1)) \log_{k-1}(|V_m|)$$

(i.e. $C = 1$) for all $m \geq 1$. Indeed, let $k > 2$ and let $(X_m)_{m\geq 1}$ be a family of connected, k-regular graphs such that $|V_m| = f(k, g(X_{m-1}) + 1)$ (where $|V_1| = f(k,g)$ for an integer $g \geq 3$). Then, $|V_m| \overset{m\to\infty}{\longrightarrow} \infty$ and for $m \geq 2$

$$|V_m| \leq 4 \cdot \sum_{j=1}^{g(X_{m-1})-1} (k-1)^j = 4 \cdot \frac{(k-1)^{g(X_{m-1})} - 1}{(k-1) - 1}$$

which is equivalent to

$$\frac{|V_m|(k-2)}{4} + 1 \leq (k-1)^{g(X_{m-1})}.$$

This leads to

$$g(X_m) \geq g(X_{m-1}) \geq \log_{k-1}\left(|V_m| \cdot \frac{k-2}{4} + 1\right) \geq \log_{k-1}\left(|V_m| \cdot \frac{k-2}{4}\right)$$

which finally implies

$$g(X_m) \geq \left(1 + \underbrace{\frac{\log_{k-1}((k-2)/4)}{\log_{k-1}(|V_m|)}}_{\longrightarrow 0}\right) \cdot \log_{k-1}(|V_m|) = (1 + o(1)) \cdot \log_{k-1}(|V_m|).$$

However, even though this proves the existence of such families of graphs, it neither gives an explicit example of such a family nor shows how such a family could be constructed.

In 1982, Margulis [11] explicitly constructed a family of 4-regular graphs with large girth, i.e. with constant $C = \frac{2}{3} \cdot \frac{1}{\log_3(1+\sqrt{2})} \approx 0.831$. We will describe this construction in Section 4.1.
In the same paper [11], the author also gave an explicit construction of regular graphs with arbitrary even degree and large girth, namely $C = \frac{4}{9}$.
Then, in 1988, Lubotzky, Phillips and Sarnak [8] explicitly constructed a family of $(p+1)$-regular Ramanujan graphs (where p is a prime congruent to 1 modulo 4) with even larger girth, namely $C = \frac{4}{3}$ (a description of it can be found in Section 4.2). This girth is even better than the bound by Erdős and Sachs from above. Around the same time, Margulis [12] independently came to similar results. He also gave an explicit construction of families of $(p+1)$-regular Ramanujan graphs with the same large girth, where p could be any prime number. Moreover, these two families of graphs do not only have a large girth, they both are in addition good families of expanders.

2.4 Cayley Graphs

Another important type of graphs are the Cayley graphs. They are particularly interesting because calculating the girth of such graphs can be done by studying the properties of certain associated sets. Moreover, we will see later that they can also be used in coding theory.

Definition 2.24. [3, Section 4.1] Let $(G, *)$ be a group and $S \subseteq G$ a nonempty, finite subset. The *Cayley graph* $\mathcal{G}(G, S)$ is the graph with vertex set $V = G$ and edge set

$$E = \{(x, y) \mid x, y \in G \text{ such that } \exists s \in S \text{ with } y = x * s\}.$$

Remark 2.25. If the set S is symmetric, i.e. $S = S^{-1}$, then the respective Cayley graph is undirected. Indeed, if (x, y) is an edge, there exists an element $s \in S$ such that $y = x * s$ and consequently $y * s^{-1} = x$ with $s^{-1} \in S$. From now on, we always assume that $S = S^{-1}$.

Moreover, we will only use multiplicative or additive groups and hence will not indicate the group operation if this is already clear from the set G.

Example 2.26. Let us look at a simple example of a Cayley graph. For this, let $G := (\mathbb{Z}/5\mathbb{Z}, +)$ and $S := \{2, 3\}$. Then, the graph $X = \mathcal{G}(G, S)$ is of the following form (see Figure 2 below).

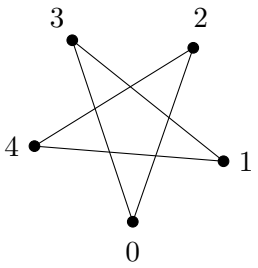

Figure 2: Example of a Cayley graph.

Before looking at some interesting properties of Cayley graphs, we need an additional definition.

Definition 2.27. [3, Section 1.4] We say that a graph X is *vertex-transitive* if the group $\mathrm{Aut}(X)$ of automorphisms of X acts transitively on the vertex set V. In other words, for every pair of vertices x, y there exists some $\alpha \in \mathrm{Aut}(X)$ such that $\alpha(x) = y$.

Proposition 2.28. [3, Proposition 4.1.2] *Let $\mathcal{G}(G, S)$ be a Cayley graph and set $k := |S|$. Then, the following properties hold.*

(i) *$\mathcal{G}(G, S)$ is a simple, k-regular, vertex-transitive graph.*

(ii) *$\mathcal{G}(G, S)$ has no loop if and only if $1 \notin S$.*

(iii) *$\mathcal{G}(G, S)$ is connected if and only if S generates G.*

(iv) *If there exists a homomorphism χ from G to the multiplicative group $\{-1, 1\}$ such that $\chi(S) = \{-1\}$, then $\mathcal{G}(G, S)$ is bipartite. The converse holds if $\mathcal{G}(G, S)$ is connected.*

Part (ii) is trivial and the proofs of the other parts can be found in [3, p. 111].

We would like to point out the well-known connection between the girth of a vertex-transitive Cayley graph $X = \mathcal{G}(G, S)$ and the set S. Namely, the girth of a Cayley graph can be determined only by looking at the elements of S. For simplicity and as we will use it later in this form, we will write the following computations in multiplicative form (i.e. we omit $*$).

Let $v_0 v_1 \dots v_{d-1} v_d$ be a cycle of X of length $d > 2$, i.e. $v_0 = v_d$. As X is vertex-transitive, we can assume that $v_0 = 1$ where 1 denotes the neutral element of G. By Definition 2.24 there exists an element $s_1 \in S$ such that $v_1 = 1 \cdot s_1$. In fact, for every $1 \leq k \leq d$ there exists an $s_k \in S$ with $v_k = v_{k-1} s_k$. Hence, $v_i = s_1 s_2 \dots s_i$ for all $1 \leq i \leq d$ and, in particular, $1 = s_1 s_2 \dots s_d$.
It follows that the girth $g(X)$ is equal to the smallest d such that there exists a subset

$\{s_1, s_2, \ldots, s_d\} \subseteq S$ with $s_i s_{i+1} \neq 1$ for all $1 \leq i < d$ (otherwise there would be backtracking) and $\prod_{i=1}^{d} s_i = 1$.

This implies, in particular, that if there does not exist any non-trivial relation between the elements of S (in this case G has to be infinite), then $\mathcal{G}(G, S)$ is an infinite tree. By contrast, if S is abelian and $|S| \geq 2$, the girth will be at most 4 because for two elements $s_1, s_2 \in S$ it holds that $s_1 s_2 s_1^{-1} s_2^{-1} = 1$. Therefore, in order to obtain a Cayley graph with large girth, S must be nonabelian.

3 Introduction to Coding Theory

The aim of this chapter is to give an overview of the basics of coding theory. We will in particular have a look at a specific type of linear codes, the low-density parity-check codes, and introduce one of their decoding algorithms, the sum-product algorithm.

Throughout this chapter, let $\mathbb{F} = \mathbb{F}_q$ denote a finite field of q elements where $q \in \mathbb{N}$ is a prime power. Moreover, as it is usually done in coding theory, vectors will be written as row vectors.

3.1 Linear Codes and the Hamming Distance

In this section, we define linear codes and introduce two matrices associated with them. Furthermore, we look at a metric used to quantify the distance between codewords. The theory has mainly been taken from [15, Chapters 1, 2 and 4].

Definition 3.1. [15, Section 2.1] An $[n, k]$ *linear (block) code* C over $\mathbb{F}$ is a k-dimensional linear subspace of $\mathbb{F}^n$ over $\mathbb{F}$ and its elements are called *codewords*. The parameter n is called the *code length* and $|C|$ is the *code size*.

Note that if C is an $[n, k]$ linear code, then its size is equal to $|C| = q^k$. Moreover, linear codes have a compact representation in terms of two matrices.

Definition 3.2. [15, Section 2.1] Let C be an $[n, k]$ linear code over $\mathbb{F}$. A $(k \times n)$-matrix G over $\mathbb{F}$ is a *generator matrix* of C if its rows form a basis of the code. Hence, C can be described by

$$C = \{vG \mid v \in \mathbb{F}^k\}.$$

Definition 3.3. [15, Section 2.3] Let C be an $[n, k]$ linear code over $\mathbb{F}$ and $r \geq n - k$. An $(r \times n)$-matrix H over $\mathbb{F}$ is a *parity-check matrix* of C if $c \in \mathbb{F}^n$ is a codeword if and only if $Hc^\top = 0$. Consequently, $C = \ker(H)$.

Remark 3.4. [15, p. 29] The fact that $C = \ker(H)$ implies that $\dim(\ker(H)) = k$ and thus $\mathrm{rk}(H) = n - \dim(\ker(H)) = n - k$. It also follows by Definitions 3.2 and 3.3 that $HG^\top = GH^\top = 0$.

Definition 3.5. [15, p. 29] A generator matrix G of an $[n, k]$ linear code is called *systematic* if it has the following form

$$G = (I_k \mid M)$$

where I_k is the $(k \times k)$ identity matrix and M is a $(k \times (n - k))$-matrix.

Remark 3.6. [15, p. 29] If a generator matrix G is in systematic form, a corresponding parity-check matrix H can quickly be computed as we can just take $H = \left(-M^\top \mid I_{n-k}\right)$.

Parity-check matrices are particularly useful to detect errors which might have occurred during the transmission of a codeword from the transmitter to the receiver. For this, we multiply a

received vector with a parity-check matrix of the respective code. If the result (which is also called *syndrome*) is nonzero, we know (by the definition of a parity-check matrix) that an error must have occurred.
However, it is not only important to detect an error but also to be able to correct it. This requires a measure to quantify how much a received vector differs from suitable codewords. In other words, if a codeword c of an $[n, k]$ linear code C has been sent and a different vector $\tilde{c} = c + e$ is received (where e is an error vector), we need a measure to quantify the distance between c and $\tilde{c}$. For this purpose, we define the Hamming distance which counts the number of coordinates on which two vectors differ.

Definition 3.7. [15, Section 1.3] Let $x, y \in \mathbb{F}^n$, $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. The *Hamming distance* d_H between x and y is defined as

$$d_H(x, y) := |\{i \in \{1, \ldots, n\} \mid x_i \neq y_i\}|.$$

It can be shown that the Hamming distance is a metric.

Definition 3.8. [15, Section 1.3] Let $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$. The *Hamming weight* wt_H of x is

$$\mathrm{wt}_H(x) := |\{i \in \{1, \ldots, n\} \mid x_i \neq 0\}|.$$

Note that $x_i \neq y_i$ if and only if $x_i - y_i \neq 0$ for every $1 \leq i \leq n$. Consequently, we can express the Hamming distance of two vectors $x, y \in \mathbb{F}^n$ as the Hamming weight of their difference, i.e.

$$d_H(x, y) = \mathrm{wt}_H(x - y). \tag{3}$$

Another important parameter of a (linear) code is its minimum distance that determines its error-correction performance, as we will discuss in Proposition 3.11.

Definition 3.9. [15, Section 1.3] The *minimum distance* of an $[n, k]$ linear code C is given by

$$d(C) := \min\{d_H(c_1, c_2) \mid c_1, c_2 \in C,\, c_1 \neq c_2\}.$$

Using the fact that C is linear and by Equation (3), we obtain

$$d(C) = \min\{\mathrm{wt}_H(c) \mid c \in C \setminus \{0\}\}.$$

From now on, if the minimum distance d of an $[n, k]$ linear code C is known, we write that C is an $[n, k, d]$ linear code.

There is another way to compute the minimum distance of a linear code that does not involve computing the weight of every codeword but instead looks at the columns of a corresponding parity-check matrix. This approach is presented in Proposition 3.10 below and a proof can be found in [15, p. 31].

Proposition 3.10. [15, Theorem 2.2] *Let H be a parity-check matrix of a nonzero $[n, k]$ linear code C. Then, $d(C)$ is equal to the largest integer d such that every set of $d-1$ columns of H is linearly independent.*

The minimum distance d determines the error-detection and error-correction capabilities of a code. If the number of errors that occurred during transmission is below a certain threshold (which depends on d), the errors can be detected or even corrected and the original codeword can be retrieved. Proposition 3.11 formalises this property of the minimum distance (a proof can be found in [15, p. 14-15]).

Proposition 3.11. [15, Proposition 1.5] *Let C be an $[n, k, d]$ linear code over $\mathbb{F}$. Then,*

(i) up to $d-1$ errors can be detected,

(ii) up to $\lfloor \frac{d-1}{2} \rfloor$ errors can be corrected.

Example 3.12. Let us look at an example of a linear code C over $\mathbb{F}_2$ with the following generator matrix G and parity-check matrix H (both in systematic form)

$$G := \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}, \quad H := \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We can check that $HG^\top = 0 \pmod 2$. Moreover, we see that G is a (3×7)-matrix with full rank, so C is a $[7, 3]$ linear code with cardinality $|C| = 2^3 = 8$.
By definition, $C = \{vG \mid v \in \mathbb{F}_2^3\}$. Consequently, three examples of possible codewords are

$$c_1 := \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \quad c_2 := \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \quad c_3 := \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

We can also determine the minimum distance of C. For this, we first note that every two columns of H are linearly independent but for instance, the third, fourth and fifth column are linearly dependent. Therefore, it follows, by Proposition 3.10, that $d(C) = 3$. Hence, up to two errors can be detected and one error can be corrected.

3.2 Bounds on Codes

In this section, we will state some bounds on different parameters of linear codes. All these bounds are written in terms of linear codes but they can also be reformulated for nonlinear codes (see for example [15, p. 94 - 97] and [19, Theorem 5.1.7]).

If we want to be able to correct/detect as many errors as possible, Proposition 3.11 implies that the minimum distance of a code should ideally be as large as possible. However, there is a limit to the minimum distance, captured by the Singleton Bound.

Theorem 3.13 (Singleton Bound)**.** [15, Theorem 4.1] *Let C be an $[n, k, d]$ linear code over $\mathbb{F}$. Then $d \leq n - k + 1$.*

Proof. The proof follows [15, p. 94].
Let H be a parity-check matrix of C. We know by Remark 3.4 that $\mathrm{rk}(H) = n - k$ which implies that H contains a set of $n - k + 1$ linearly dependent columns. Hence, it follows by Proposition 3.10 that the minimum distance is at most $n - k + 1$. □

The next bound is called Sphere-Packing Bound and it is a bound on the cardinality of a code. For this, we first need to define the notion of a ball in $\mathbb{F}^n$.

Definition 3.14. [15, Section 4.2] Let $x \in \mathbb{F}^n$ and $r \in \mathbb{R}_{\geq 0}$. The *ball* of radius r around x is

$$B_r(x) := \{y \in \mathbb{F}^n \mid d_H(x, y) \leq r\}.$$

We denote its volume by $V_q(n, r) := |B_r(x)|$.

If c is a codeword of a linear code with minimum distance d, then $B_{\lfloor (d-1)/2 \rfloor}(c)$ contains no other codeword. This leads to the following upper bound for the cardinality of a linear code (a proof can be found in [15, Section 4.2]).

Theorem 3.15 (Sphere-Packing Bound)**.** [15, Theorem 4.3] *For any $[n, k, d]$ linear code C over $\mathbb{F}$ it holds that*

$$|C| = q^k \leq \frac{q^n}{V_q(n, \lfloor (d-1)/2 \rfloor)}.$$

If we have a code C with minimum distance d and maximal code size, the balls of radius $d - 1$ around all the codewords cover the whole space $\mathbb{F}^n$. Based on this observation, a lower bound for the maximal code size can be determined. This bound is called the Gilbert-Varshamov bound.

Theorem 3.16 (Gilbert-Varshamov bound)**.** [19, Theorem 5.1.7] *Let $n, d \in \mathbb{N}$ such that $d \leq n$. Furthermore, let $A_q(n, d)$ denote the maximal number M such that a code $C \subseteq \mathbb{F}^n$ exists with $|C| = M$ and $d(C) = d$. Then,*

$$A_q(n, d) \geq \frac{q^n}{V_q(n, d-1)}.$$

Proof. The proof follows [19, p. 60].
Let $C \subseteq \mathbb{F}^n$ be a code with $d(C) = d$ and $|C| = A_q(n, d)$. Assume by contradiction that there exists a vector $x \in \mathbb{F}^n$ such that $d(x, c) \geq d$ for all $c \in C$. Then, $\tilde{C} := C \cup \{x\} \subseteq \mathbb{F}^n$ is a code with $d(\tilde{C}) = d$ and $|\tilde{C}| = |C| + 1$. This is a contradiction as $|C| = A_q(n, d)$. Hence, for every $x \in \mathbb{F}_q^n$ there exists a codeword $c \in C$ such that $x \in B_{d-1}(c)$ or equivalently, $\bigcup_{c \in C} B_{d-1}(c) = \mathbb{F}^n$. This implies that

$$|C| \cdot V_q(n, d-1) \geq |\mathbb{F}^n|$$

which is equivalent to

$$A_q(n, d) \geq \frac{q^n}{V_q(n, d-1)}.$$

□

Remark 3.17. As pointed out by Margulis [12], the Gilbert-Varshamov bound can be compared with the Erdős-Sachs bound (Proposition 2.22) as the proofs of both of them are nonconstructive. In other words, both proofs do not show how we can explicitly construct an $[n, k]$ code C with $|C| \geq \frac{q^n}{V_q(n,d-1)}$ respectively such a k-regular graph X with girth at least g; they just show that such codes respectively graphs exist.

3.3 Low-Density Parity-Check Codes

In this part, we will introduce a particular class of linear block codes called low-density parity-check codes and their representation by a bipartite graph, mainly following [16].
Low-density parity-check codes (LDPC codes) were introduced by Robert G. Gallager [6]. A *binary LDPC-code* is a linear block code over $\mathbb{F}_2$ whose parity-check matrix H contains only a few ones and the remaining entries are all zero.

Definition 3.18. [16, Section 2.1] Let H be an $(m \times n)$ parity-check matrix of an LDPC code C, where $m \geq n - k$. The code C is called a (λ, ρ)-*regular LDPC code* if every column of H contains exactly λ ones and each row of H contains exactly ρ ones.
If the number of ones in each column or row is not constant, C is called an *irregular LDPC code*.

Note that, by definition of an LDPC code, λ and ρ must fulfil $\lambda << m$ and $\rho << n$. From now on, we will only consider (λ, ρ)-regular LDPC codes C.

Instead of representing C by a parity-check matrix H, we can also consider its corresponding Tanner graph. This is a bipartite graph which was first described by R. Michael Tanner [18].
The two disjoint sets of nodes (see Definition 2.9) in a Tanner graph are called variable nodes (or bit nodes) and check nodes (or function nodes). For simplicity, we will call them v-nodes and c-nodes respectively. There is an edge between c-node f_j and v-node c_i if and only if the entry h_{ji} of H is equal to 1. In other words, the entry h_{ji} corresponds to the j^{th} check node and the i^{th} variable node. Consequently, there are m c-nodes and n v-nodes.

Example 3.19. We consider a $(2, 3)$-regular LDPC code represented by the following (8×12) parity-check matrix

$$H := \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \tag{4}$$

The corresponding Tanner graph (see Figure 3) thus contains 8 c-nodes $f_0, \ldots, f_7$ and 12 v-nodes $c_0, \ldots, c_{11}$. We see that every c-node is connected to exactly $\rho = 3$ v-nodes and every v-node

is connected to exactly $\lambda = 2$ c-nodes. The fact that $Hc^\top = 0$ for any codeword c implies that the values of the variable nodes connected to the same check node f_j must add up to 0. The resulting equation is called check equation f_j.
For example, if $\tilde{c} = (\tilde{c}_0, \tilde{c}_1, \ldots, \tilde{c}_{11})$ is a codeword, it follows, due to the first row of H, that $\tilde{c}_0 + \tilde{c}_3 + \tilde{c}_4 \equiv 0 \pmod 2$. Looking at Figure 3, we see that f_0, which represents this first row, is connected to the bit nodes c_0, c_3 and c_4. The same can be checked for the remaining rows respectively check nodes.

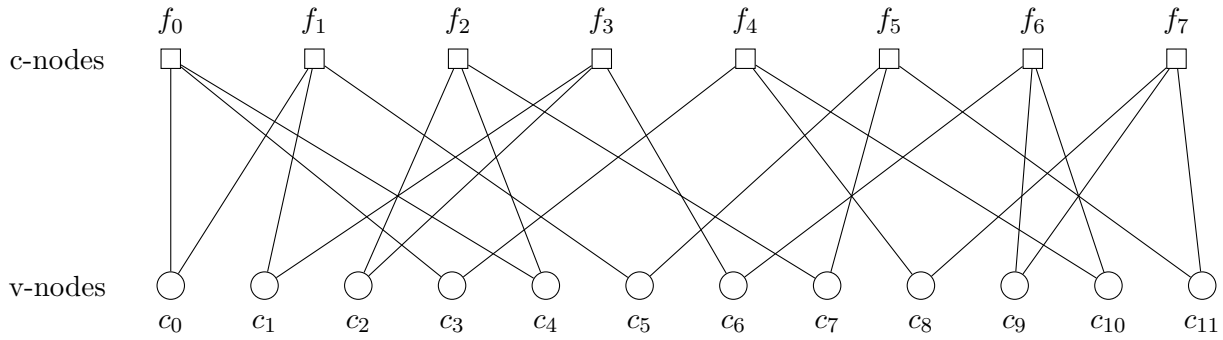

Figure 3: Tanner graph representation of the matrix H defined in (4).

3.4 Sum-Product Algorithm

In his paper [6], Gallager also presented several decoding algorithms for LDPC codes, one of which is called the sum-product algorithm (SPA). There exist other decoding algorithms for LDPC codes (see for example [17, Chapter 5]), but we will only introduce the SPA which is based on the representation of the code as a Tanner graph. The aim of this section is to introduce the idea of the SPA and to describe how it works. This part is mainly based on [6, p. 23-24], [16, Chapter 4] and [17, Chapter 5].

The basic idea of the SPA is to view the v-nodes and c-nodes as processors that can send messages along the edges. Therefore, two processors can communicate with each other if and only if there is an edge between the corresponding nodes. The messages sent from v-nodes to c-nodes contain information about the probability of the respective v-node to be a 0 or a 1. The messages sent from c-nodes to v-nodes contain information about the probability of the respective check equation to be satisfied.

The aim of this algorithm is to compute the a posteriori-probability (APP)

$$\mathbb{P}[c_i = 1 \,|\, y]$$

for every bit c_i where $c = (c_0, \ldots, c_{n-1})$ is the transmitted binary codeword and $y = (y_0, \ldots, y_{n-1})$ is the received word (i.e. $y = c + e$ where e is some error vector). However, we will eventually

compute a slightly different version of the APP, namely the log-likelihood ratio (LLR) (more details can be found in Section 3.4.2)

$$L(c_i) := \ln\left(\frac{\mathbb{P}[c_i = 0 \,|\, y]}{\mathbb{P}[c_i = 1 \,|\, y]}\right).$$

The SPA consists of several iterations where messages are sent from v-nodes to c-nodes and vice versa. During the first half of an iteration, every v-node receives some input messages, processes them and sends the resulting messages to its adjacent c-nodes. We denote the message from v-node c_i to c-node f_j by $m_{\uparrow ij}$ (the arrow shows the direction of the message, according to the illustration of a Tanner graph in Figure 3). This message contains the information available to c_i through the received word and through its adjacent nodes except f_j (see Figure 4 for an illustration). The message sent from c_i to f_j should only contain "new" information, i.e. information that the check node f_j does not already have. So, only *extrinsic information* is passed along.
In simplified terms, $m_{\uparrow ij}$ consists of $\mathbb{P}[c_i = b \,|\, \text{input messages}]$ where $b \in \{0, 1\}$ or of the logarithm of the ratio of these two probabilities.

During the second half of an iteration, messages are sent the other way round. Consequently, we denote the message from c-node f_j to v-node c_i by $m_{\downarrow ji}$. It contains the information $\mathbb{P}[\text{check equation } f_j \text{ is satisfied} \,|\, \text{input messages}]$ respectively the logarithm of the ratio of these probabilities. As before, only extrinsic information is sent.
Figure 4 shows an illustration of how messages are sent and received at a v-node c_i. It sends a message $m_{\uparrow ij_2}$ to c-node f_{j_2} which depends on the received messages from all other adjacent c-nodes and from the channel (y_i).

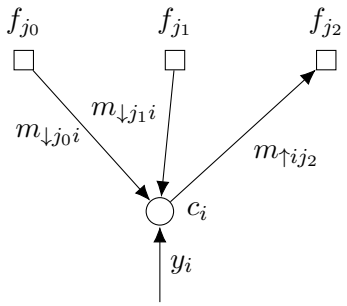

Figure 4: Segment of a Tanner graph which shows how messages are sent between the nodes.

Finally, the APP or LLR is computed to decide for each bit of the codeword whether it is a 0 or a 1. The algorithm terminates when it reaches a certain number of iterations or if $H\tilde{c}^\top = 0$ where $\tilde{c}$ is the decoded codeword obtained in the respective iteration.
Note that even if $H\tilde{c}^\top = 0$, this does not guarantee that the algorithm was successful. It might also have found a different codeword $\tilde{c} \neq c$ (these are called *undetected errors*; see Section 5.2 for more details).

3.4.1 Probability-Domain Sum-Product Algorithm

In a first step, we describe the Probability-Domain SPA Decoder, which computes APPs, before we modify it to obtain LLRs. We use the following notations.

- $\mathbf{V_j} := \{i' \mid h_{ji'} = 1\} = \{\text{indices of v-nodes connected to c-node } f_j\}$
- $\mathbf{C_i} := \{j' \mid h_{j'i} = 1\} = \{\text{indices of c-nodes connected to v-node } c_i\}$
- $\mathbf{P_i} := \mathbb{P}[c_i = 1 \mid y]$
- $\mathbf{S_{ij}}$:= event that the check equations $\{f_{j'} \mid j' \in C_i \setminus \{j\}\}$ are satisfied (i.e. the check equations involving c_i except f_j)
- $\mathbf{q_{ij}(b)} := \mathbb{P}[c_i = b \mid S_{ij}, y]$ where $b \in \{0, 1\}$. For the APP algorithm $m_{\uparrow ij} = q_{ij}(b)$ and for the LLR algorithm $m_{\uparrow ij} = \ln(q_{ij}(0)/q_{ij}(1))$.
- $\mathbf{r_{ji}(b)} := \mathbb{P}[\text{check equation } f_j \text{ is satisfied } \mid c_i = b]$ where $b \in \{0, 1\}$. For the APP algorithm $m_{\downarrow ji} = r_{ji}(b)$ and for the LLR algorithm $m_{\downarrow ji} = \ln(r_{ji}(0)/r_{ji}(1))$.

Remark 3.20. We assume that all the sent messages ($m_{\downarrow ji}$ and $m_{\uparrow ij}$) are independent random variables even though this actually only holds if the Tanner graph is acyclic. Yet, simulations have shown that the SPA still works quite well if the girth is strictly larger than 4 (see [16, Section 4.1]).

Initialisation

The algorithm starts with the v-nodes sending $m_{\uparrow ij} = \mathbb{P}[c_i = b \mid y]$ to their adjacent c-nodes, where i, j are such that $h_{ji} = 1$.

Calculation of $\mathbf{q_{ij}(b)}$ and $\mathbf{r_{ji}(b)}$

For the calculation of $q_{ij}(b)$ and $r_{ji}(b)$ we need the following lemma (a proof can be found in [6, page 24]).

Lemma 3.21. [16, Section 4.2] *Let $\{a_i\}_{i=1}^d$ be a sequence of d independent binary digits with $\mathbb{P}[a_i = 1] = p_i$. The probability that $\{a_i\}_{i=1}^d$ contains an even number of 1's is equal to*

$$\frac{1}{2} + \frac{1}{2}\prod_{i=1}^{d}(1 - 2p_i).$$

Now we note that, given $c_i = 0$, check equation f_j is satisfied if an even number of all other adjacent v-nodes of f_j is equal to 1. Using Lemma 3.21, we obtain

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{i' \in V_j \setminus \{i\}}(1 - 2q_{i'j}(1)) \quad \text{and} \quad r_{ji}(1) = 1 - r_{ji}(0). \tag{5}$$

To find a similar expression for $q_{ij}(b)$, we first need to rearrange $q_{ij}(b)$ using Bayes' theorem[1]

$$q_{ij}(0) = \mathbb{P}[c_i = 0 \mid S_{ij}, y] = \frac{\mathbb{P}[S_{ij} \mid c_i = 0, y] \cdot \mathbb{P}[c_i = 0 \mid y]}{\mathbb{P}[S_{ij} \mid y]} = \frac{\mathbb{P}[S_{ij} \mid c_i = 0, y] \cdot (1 - P_i)}{\mathbb{P}[S_{ij} \mid y]}.$$

We also need the observation that, due to the independence assumption, we can write

$$\mathbb{P}[S_{ij}] \quad = \prod_{j' \in C_i \setminus \{j\}} \mathbb{P}[\text{check equation } f_{j'} \text{ is satisfied}].$$

If we apply this to $\mathbb{P}[S_{ij} \mid c_i = 0, y]$ and recall the definition of $r_{ji}(0)$, we arrive at

$$q_{ij}(0) = \frac{(1 - P_i) \cdot \mathbb{P}[S_{ij} \mid c_i = 0, y]}{\mathbb{P}[S_{ij} \mid y]} = K_{ij}(1 - P_i) \cdot \prod_{j' \in C_i \setminus \{j\}} r_{j'i}(0) \tag{6}$$

where K_{ij} is defined as $K_{ij} := 1/\mathbb{P}[S_{ij} \mid y]$. Similarly,

$$q_{ij}(1) = K_{ij} P_i \cdot \prod_{j' \in C_i \setminus \{j\}} r_{j'i}(1). \tag{7}$$

Now that we know how to calculate all the messages, we can write down the SPA. Its steps are presented in a certain order but other orders are also possible (for example, step 4 could be calculated before step 3).

Overview of the Probability-Domain SPA

1. Set $q_{ij}(0) = 1 - P_i$ and $q_{ij}(1) = P_i$ for all $0 \leq i \leq n-1$ and all j such that $h_{ji} = 1$.
2. Compute $\{r_{ji}(b) | h_{ij} = 1,\, b \in \{0, 1\}\}$ using Equation (5).
3. Compute $\{q_{ij}(b) | h_{ij} = 1,\, b \in \{0, 1\}\}$ using Equations (6) and (7).
4. For all $i \in \{0, \ldots, n-1\}$, compute

   $$Q_i(0) := K_i(1 - P_i) \cdot \prod_{j \in C_i} r_{ji}(0) \;\text{ and }\; Q_i(1) := K_i P_i \cdot \prod_{j \in C_i} r_{ji}(1)$$

   where the K_i's are constants such that $Q_i(0) + Q_i(1) = 1$.
5. For all $i \in \{0, \ldots, n-1\}$ define $\tilde{c}_i := \begin{cases} 1 \text{ if } Q_i(1) > Q_i(0) \\ 0 \text{ else} \end{cases}$.
6. If $H\tilde{c}^\top = 0$ or if the maximal number of iterations is attained, stop the algorithm. Otherwise, continue with step 2.

As we can see, there are many multiplications involved in the Probability-Domain SPA which we try to avoid when implementing the algorithm. That is why we also look at the log-domain version of the SPA.

[1] Let A, B, C be three events such that $\mathbb{P}[B \,|\, C] \neq 0$. Then, $\mathbb{P}[A \,|\, B \cap C] = \frac{\mathbb{P}[B|A \cap C] \cdot \mathbb{P}[A|C]}{\mathbb{P}[B|C]}$.

3.4.2 Log-Domain Sum-Product Algorithm

Let us first briefly describe the channel model which we used for the simulations in Chapter 5. There exist many different channel models, but we will only consider the Binary Additive White-Gaussian-Noise Channel (BI-AWGN channel). For this, the bits of a codeword $c = (c_0, c_1, \dots, c_{n-1})$ are mapped to $x_i := 1 - 2c_i$, i.e. $x_i = 1$ (respectively -1) if $c_i = 0$ (respectively 1). Then, the received word is $y = x + e$ where $e = (e_0, \dots, e_{n-1})$ and the e_i's are independent and identically distributed (i.i.d.) random variables, distributed according to $\mathcal{N}(0, \sigma^2)$ for some $\sigma > 0$. Here, $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with mean 0 and variance σ^2. Consequently, the channel transition probability density function is

$$p(y_i \mid x_i) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(y_i - x_i)^2}{2\sigma^2}\right). \tag{8}$$

As before, let us introduce some notations.

$$\begin{aligned} L(c_i) &:= \ln\left(\frac{\mathbb{P}[c_i = 0 \mid y_i]}{\mathbb{P}[c_i = 1 \mid y_i]}\right), \\ L(r_{ji}) &:= \ln\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right), \\ L(q_{ij}) &:= \ln\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right), \\ L(Q_i) &:= \ln\left(\frac{Q_i(0)}{Q_i(1)}\right). \end{aligned}$$

Initialisation

Similar to the Probability-Domain SPA, we set $m_{\uparrow ij} = L(c_i)$ for any i, j with $h_{ji} = 1$. As already mentioned, we assume that our codeword c is sent through a BI-AWGN channel with variance σ^2. Using Bayes' theorem and then Equation (8), we arrive at

$$\mathbb{P}[x_i = x \mid y_i] = \frac{1}{1 + \exp\left(-\frac{2y_i x}{\sigma^2}\right)}$$

where $x \in \{\pm 1\}$ and we assume that $\mathbb{P}[x_i = 1] = \mathbb{P}[x_i = -1] = \frac{1}{2}$. It follows that

$$L(q_{ij}) = L(c_i) = \frac{2y_i}{\sigma^2}. \tag{9}$$

Calculation of $\mathbf{L(r_{ji})}$

Using Equation (5) and twice the fact that $\tanh\left(\frac{1}{2} \cdot \ln\left(\frac{p_0}{p_1}\right)\right) = 1 - 2p_1$ for $p_0, p_1 > 0$ with $p_0 + p_1 = 1$, we obtain

$$\tanh\left(\frac{1}{2} \cdot L(r_{ji})\right) = \prod_{i' \in V_j \setminus \{i\}} \tanh\left(\frac{1}{2} \cdot L(q_{i'j})\right). \tag{10}$$

It is numerically more convenient to remove the tanh function and to instead have a minimum

of products. For this, we first define

$$\alpha_{ij} := \operatorname{sgn}(L(q_{ij})) \quad \text{and} \quad \beta_{ij} := |L(q_{ij})|$$

such that $L(q_{ij}) = \alpha_{ij} \cdot \beta_{ij}$.
Using that $\tanh(-a) = -\tanh(a)$ for any $a \in \mathbb{R}$, Equation 10 can be rewritten as

$$\tanh\left(\frac{1}{2}L(r_{ji})\right) = \left(\prod_{i' \in V_j \setminus \{i\}} \alpha_{i'j}\right) \cdot \left(\prod_{i' \in V_j \setminus \{i\}} \tanh\left(\frac{1}{2}\beta_{i'j}\right)\right). \tag{11}$$

Now, define the map

$$\phi(x) := -\ln\left(\tanh\left(\frac{x}{2}\right)\right) = \ln\left(\frac{e^x + 1}{e^x - 1}\right)$$

and note that $\phi^{-1}(x) = \phi(x)$ for all $x > 0$. Using this, we can further rewrite Equation 11

$$\begin{aligned}
L(r_{ji}) &= \left(\prod_{i' \in V_j \setminus \{i\}} \alpha_{i'j}\right) \cdot 2\tanh^{-1}\left(\prod_{i' \in V_j \setminus \{i\}} \tanh\left(\frac{1}{2}\beta_{i'j}\right)\right) \\
&= \left(\prod_{i' \in V_j \setminus \{i\}} \alpha_{i'j}\right) \cdot 2\tanh^{-1}\ln^{-1}\ln\left(\prod_{i' \in V_j \setminus \{i\}} \tanh\left(\frac{1}{2}\beta_{i'j}\right)\right) \\
&= \left(\prod_{i' \in V_j \setminus \{i\}} \alpha_{i'j}\right) \cdot 2\tanh^{-1}\ln^{-1}\left(-\sum_{i' \in V_j \setminus \{i\}} -\ln\left(\tanh\left(\frac{1}{2}\beta_{i'j}\right)\right)\right) \\
&= \left(\prod_{i' \in V_j \setminus \{i\}} \alpha_{i'j}\right) \cdot \phi\left(\sum_{i' \in V_j \setminus \{i\}} \phi(\beta_{i'j})\right).
\end{aligned} \tag{12}$$

Calculation of $\mathbf{L(q_{ij})}$ and $\mathbf{L(Q_i)}$

By Equations (6) and (7) and by the definition of $L(q_{ij})$, we obtain

$$L(q_{ij}) = L(c_i) + \sum_{j' \in C_i \setminus \{j\}} L(r_{j'i}). \tag{13}$$

Similarly, we obtain the expression

$$L(Q_i) = L(c_i) + \sum_{j \in C_i} L(r_{ji}). \tag{14}$$

We can now write down the procedure of the Log-Domain SPA. Note that this procedure is basically the same as the one of the probability-domain version, except that different quantities are computed.

Overview of the Log-Domain SPA for a BI-AWGN Channel

1. For $i \in \{0, 1, \ldots, n-1\}$ and for all j with $h_{ji} = 1$, initialise $L(q_{ij})$ using Equation (9).
2. Compute $\{L(r_{ji}) \,|\, h_{ji} = 1\}$ according to Equation (12).
3. Compute $\{L(q_{ij}) \,|\, h_{ji} = 1\}$ according to Equation (13).
4. Compute $\{L(Q_i) \,|\, i \in \{0, 1, \ldots, n-1\}\}$ according to Equation (14).
5. For all $i \in \{0, \ldots, n-1\}$ define $\tilde{c}_i := \begin{cases} 1 \text{ if } L(Q_i) < 0 \\ 0 \text{ else} \end{cases}$.
6. If $H\tilde{c}^\top = 0$ or if the maximal number of iterations is attained, stop the algorithm. Otherwise, continue with step 2.

4 Constructions of Cayley Graphs with Large Girth

In this chapter, we look at two constructions of graphs with large girth (both were already mentioned in Chapter 2.3). We will have a closer look especially at the first one and present a slight improvement of the lower bound for its girth.

4.1 Construction by Margulis

We discuss a construction of a family of 4-regular graphs with large girth introduced by Margulis [11, Section 5]. This subsection is mainly based on [11] and [3, Appendix].

We define the matrices

$$A := \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

Clearly, A and B are elements of $\mathrm{SL}_2(\mathbb{Z})$ (i.e. they are (2×2)-matrices with coefficients in $\mathbb{Z}$ and with determinant one). A *word* w of length $k \in \mathbb{N}$ over $\{A, B\}$ is an expression of the form $w := w_1 \dots w_k$ where, for every $1 \leq i \leq k$, either w_i or w_i^{-1} is an element of $\{A, B\}$. Moreover, w is said to be *reduced* if $w_i \neq w_{i-1}^{-1}$ for all $1 < i \leq k$.

Proposition 4.1. [11, Section 5] *Any two reduced words over $\{A, B\}$ define different elements of $SL_2(\mathbb{Z})$ (i.e. there does not exist any nontrivial multiplicative relation between A and B).*

Proof. We follow the proof of Proposition A.2.2 in [3].
First, note that if there existed two reduced words $w, \tilde{w}$ over $\{A, B\}$ such that $w = \tilde{w}$, then $w\tilde{w}^{-1} = 1$, where 1 denotes the identity element in $\mathrm{SL}_2(\mathbb{Z})$. Hence, instead of showing that any two such words define different elements, we show that $w \neq 1$ for any nonempty reduced word w over $\{A, B\}$ (i.e. we show that there does not exist any nontrivial multiplicative relation between A and B). Let H be the set of all nonempty reduced words over $\{A, B\}$. Then, H can be split into four disjoint subsets

$$\begin{aligned}
H_1 &:= \{A^{k_1} B^{l_1} A^{k_2} B^{l_2} \dots A^{k_n} B^{l_n} A^{k_{n+1}} \mid n \in \mathbb{N},\, k_i, l_j \in \mathbb{Z} \setminus \{0\} \,\forall\, 1 \leq i \leq n+1,\, 1 \leq j \leq n\}, \\
H_2 &:= \{B^{k_1} A^{l_1} B^{k_2} A^{l_2} \dots B^{k_n} A^{l_n} B^{k_{n+1}} \mid n \in \mathbb{N},\, k_i, l_j \in \mathbb{Z} \setminus \{0\} \,\forall\, 1 \leq i \leq n+1,\, 1 \leq j \leq n\}, \\
H_3 &:= \{A^{k_1} B^{l_1} A^{k_2} B^{l_2} \dots A^{k_n} B^{l_n} \mid n \in \mathbb{N},\, k_i, l_j \in \mathbb{Z} \setminus \{0\} \,\forall\, 1 \leq i, j \leq n\}, \\
H_4 &:= \{B^{k_1} A^{l_1} B^{k_2} A^{l_2} \dots B^{k_n} A^{l_n} \mid n \in \mathbb{N},\, k_i, l_j \in \mathbb{Z} \setminus \{0\} \,\forall\, 1 \leq i, j \leq n\}.
\end{aligned}$$

Let $\mathrm{SL}_2(\mathbb{Z})$ act on $\mathbb{R}^2$ by the usual linear action. We then define the following two subsets $E, F \subseteq \mathbb{R}^2$

$$E := \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 \,\middle|\, |y| > |x| \right\} \quad \text{and} \quad F := \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 \,\middle|\, |x| > |y| \right\}.$$

It is easily checked that $A^k(E) \subset F$ for all $k \in \mathbb{Z} \setminus \{0\}$ and $B^l(F) \subset E$ for all $l \in \mathbb{Z} \setminus \{0\}$. Indeed,

if we take $(x,y)^\top \in E$ and $k \in \mathbb{Z} \setminus \{0\}$, then

$$A^k \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 2k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+2ky \\ y \end{pmatrix} \in F \tag{15}$$

because $|x+2ky| > |y|$ as $|2ky| \geq |2y|$ and $|y| > |x|$. The other case is similar.

We now want to prove that any word $w^{(i)} \in H_i$, for $1 \leq i \leq 4$, is a non-identity element of $\mathrm{SL}_2(\mathbb{Z})$ by using a method called the "ping-pong lemma" (this method was first presented by Fricke and Klein [5]). So, let $w^{(1)} \in H_1$, i.e. $w^{(1)} = A^{k_1}B^{l_1}A^{k_2}B^{l_2}\dots A^{k_n}B^{l_n}A^{k_{n+1}}$ for some $n \in \mathbb{N}$. We consider $w^{(1)}(E)$ by noting that

$$\begin{gathered} A^{k_{n+1}}(E) \subset F, \\ B^{l_n}A^{k_{n+1}}(E) \subset E, \\ A^{k_n}B^{l_n}A^{k_{n+1}}(E) \subset F, \\ B^{k_{n-1}}A^{k_n}B^{l_n}A^{k_{n+1}}(E) \subset E, \\ \vdots \\ w^{(1)}(E) = A^{k_1}B^{l_1}\dots B^{l_n}A^{k_{n+1}}(E) \subset F. \end{gathered}$$

It follows that $w^{(1)} \neq 1$ as applying $w^{(1)}$ to an element of E results in an element of F and $E \cap F = \emptyset$. For a word $w^{(2)}$ in H_2 we can do exactly the same by applying it to F and we obtain that $w^{(2)}(F) \subset E$ which implies that $w^{(2)} \neq 1$.
Finally, consider $w^{(3)} = A^{k_1}B^{l_1}\dots A^{k_n}B^{l_n} \in H_3$ for some $n \in \mathbb{N}$. Take $k \in \mathbb{Z} \setminus \{0\}$ such that $k \neq k_1$ and consider $w := A^{-k}w^{(3)}A^k$. We see that $w \in H_1$ and therefore $w \neq 1$ which implies that also $w^{(3)} \neq 1$. Similarly, we obtain that all words in H_4 are non-identity elements. □

Now, let p be any odd prime and denote $\mathbb{Z}_p := \mathbb{Z}/p\mathbb{Z}$. We consider $\varphi_p : \mathrm{SL}_2(\mathbb{Z}) \to \mathrm{SL}_2(\mathbb{Z}_p)$ where $\varphi_p(M)$ is the matrix obtained by reducing every entry of M modulo p. We define

$$A_p := \varphi_p(A),\ B_p := \varphi_p(B) \text{ and } S_p := \{A_p, A_p^{-1}, B_p, B_p^{-1}\}. \tag{16}$$

Note that $|S_p| = 4$ as p is odd. Then,

$$X_p := \mathcal{G}(\mathrm{SL}_2(\mathbb{Z}_p), S_p)$$

is a 4-regular graph (by Proposition 2.28 (i)). Moreover, X_p is connected and has $p(p^2-1)$ vertices (see [3, Lemma A.1.1] for a proof).

4.1.1 Large Girth Property

The aim of this section is to prove Theorem 4.2 about the girth of X_p.

Theorem 4.2. [11, Section 6] *Let $n(X_p)$ denote the number of vertices of X_p. For any odd prime p it holds that*

$$g(X_p) \geq \left(\frac{2}{3} \cdot \frac{1}{\log_3(1+\sqrt{2})} + o(1) \right) \log_3(n(X_p)).$$

Before proving this theorem, we first need some preliminary considerations. Let $\|\cdot\|_2$ denote the Euclidean norm in $\mathbb{R}^2$.

Definition 4.3. [3, Appendix] Let M be a real (2×2)-matrix. The *operator norm* of M is defined as

$$\|M\| := \sup \left\{ \frac{\|Mv\|_2}{\|v\|_2} \,\middle|\, v \in \mathbb{R}^2 \setminus \{(0,0)\} \right\}.$$

By noting that $\|Mv\|_2 \leq \|M\| \, \|v\|_2$ for any real (2×2)-matrix M and any $v \in \mathbb{R}^2$, it is easily shown that

$$\|MM'\| \leq \|M\| \, \|M'\| \quad \text{for any real } (2 \times 2)\text{-matrices } M, M'. \tag{17}$$

We also need the following lemma, which will be stated without proof (a proof can be found in [3, p. 135-136]).

Lemma 4.4. [3, Lemma A.3.3] *Let $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with $a, b, c, d \in \mathbb{R}$. Then, the following properties hold.*

(i) $\|M\| = \|M^\top\|$,

(ii) $\|M\| = \|M^\top M\|^{1/2}$,

(iii) $\|M\| \geq \max\{|a|, |b|, |c|, |d|\}$,

(iv) If M is symmetric and $\lambda_1, \lambda_2 \in \mathbb{R}$ are its eigenvalues, then $\|M\| = \max\{|\lambda_1|, |\lambda_2|\}$.

We can now come to the proof of Theorem 4.2. The proof stated below follows the structure of the proof in [3, p. 136-137] but it is modified a bit (as the original proof shows a slightly weaker bound).

Proof of Theorem 4.2. For simplicity, let $g := g(X_p)$. Let us consider a cycle of length g starting and ending at the same vertex v_0. As the graph X_p is vertex-transitive (by Proposition 2.28 (i)), we can assume without loss of generality that $v_0 = 1$ (where 1 denotes the identity matrix in $\mathrm{SL}_2(\mathbb{Z}_p)$). Thus, we consider a cycle of the form

$$v_0 = 1, v_1, v_2, \ldots, v_{g-1}, v_g = 1.$$

Since X_p is a Cayley graph, there exist matrices $s_1, s_2, \ldots, s_g \in S_p$ such that $v_i = s_1 s_2 \ldots s_{i-1} s_i$ for all $1 \leq i \leq g$. In particular, it holds that $s_1 s_2 \ldots s_{g-1} s_g = 1$. By definition of the set S_p (see (16)), for any $1 \leq i \leq g$ there exists a unique element $\tilde{s}_i \in \{A, A^{-1}, B, B^{-1}\}$ such that

$\varphi_p(\tilde{s}_i) = s_i$. Consequently, $\tilde{s}_1\tilde{s}_2\dots\tilde{s}_{g-1}\tilde{s}_g$ is a reduced word over $\{A, B\}$. It therefore follows, by Proposition 4.1, that $\tilde{s}_1\tilde{s}_2\dots\tilde{s}_{g-1}\tilde{s}_g \neq 1$ or equivalently

$$\tilde{s}_1\dots\tilde{s}_{\lfloor g/2\rfloor} \neq \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2\rfloor+1}^{-1}. \tag{18}$$

On the other hand, $\varphi_p\left(\tilde{s}_1\dots\tilde{s}_{\lfloor g/2\rfloor}\right) = \varphi_p\left(\tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2\rfloor+1}^{-1}\right)$ which implies that all entries of the matrix $\tilde{s}_1\dots\tilde{s}_{\lfloor g/2\rfloor} - \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2\rfloor+1}^{-1}$ are divisible by p.
However, we know by (18) that $\tilde{s}_1\dots\tilde{s}_{\lfloor g/2\rfloor} - \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2\rfloor+1}^{-1} \neq 0$ and therefore, applying Lemma 4.4 (iii) and the fact that all its entries are divisible by p, we obtain

$$\|\tilde{s}_1\dots\tilde{s}_{\lfloor g/2\rfloor} - \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2\rfloor+1}^{-1}\| \geq p.$$

By the triangle inequality, we get that

$$\underbrace{\|\tilde{s}_1\dots\tilde{s}_{\lfloor g/2\rfloor}\|}_{=:c_1} \geq p - \underbrace{\|\tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2\rfloor+1}^{-1}\|}_{=:c_2}.$$

Note that if $c_2 < \frac{p}{2}$, then necessarily $c_1 \geq \frac{p}{2}$. It hence follows that $\frac{p}{2} \leq \max\{c_1, c_2\}$.

Furthermore, we can quickly verify (using Lemma 4.4 (ii) and (iv)) that the operator norm of all matrices in $\{A, A^{-1}, B, B^{-1}\}$ is equal to $1+\sqrt{2}$. Applying the submultiplicativity of the norm (see Equation (17)), we obtain

$$\frac{p}{2} \leq \max\{c_1, c_2\} \leq (1+\sqrt{2})^{\lfloor g/2\rfloor+1} \leq (1+\sqrt{2})^{g/2+1}. \tag{19}$$

Then, first applying $\log_3(\cdot)$ and then using the fact that $n(X_p) = p^3 - p \leq p^3$, we get

$$\log_3\left(\frac{p}{2}\right) \leq \left(\frac{g}{2}+1\right)\log_3(1+\sqrt{2}) \text{ and } \log_3\left(\frac{p}{2}\right) \geq \log_3\left(\frac{\sqrt[3]{n(X_p)}}{2}\right) = \frac{1}{3}\log_3\left(\frac{n(X_p)}{8}\right).$$

We thus arrive at

$$\left(\frac{g}{2}+1\right)\log_3(1+\sqrt{2}) \geq \frac{1}{3}\log_3\left(\frac{n(X_p)}{8}\right)$$

which is equivalent to

$$(g+2)\log_3(1+\sqrt{2}) \geq \frac{2}{3}\log_3\left(\frac{n(X_p)}{8}\right).$$

We can further rearrange this to obtain

$$g \geq \frac{2}{3}\cdot\frac{\log_3(n(X_p))}{\log_3(1+\sqrt{2})} - \frac{2}{3}\cdot\frac{\log_3(8)}{\log_3(1+\sqrt{2})} - 2$$

which is equivalent to

$$g \geq \left(\frac{2}{3} \cdot \frac{1}{\log_3(1+\sqrt{2})} - \underbrace{\frac{2}{3} \cdot \frac{\log_3(8)}{\log_3(1+\sqrt{2}) \cdot \log_3(n(X_p))} - \frac{2}{\log_3(n(X_p))}}_{\to 0 \text{ for } n(X_p) \to \infty} \right) \log_3(n(X_p))$$

and this concludes the proof. □

Remark 4.5. The family of graphs $(X_p)_{p \geq 3}$, p prime, is not only a family with large girth, it is also a family of expander graphs (see for example [2, p. 626]).

4.1.2 Other Matrices

For $a, b \in \mathbb{N}$, let

$$A_a := \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}, \quad B_b := \begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix}.$$

We saw in the beginning of Chapter 4.1 that Margulis used $a = b = 2$ to build his Cayley graphs. The question arises if it is possible to obtain a better bound if we choose different a and b.
In this section, we will discuss this question and look at some additional approaches with different matrices A_a, B_b where for example $a, b \notin \mathbb{N}$. We set $S := \{A_a, A_a^{-1}, B_b, B_b^{-1}\}$.

First of all, we note that the lower bound for the girth in Theorem 4.2 depends on the operator norm of the elements of S, which can be seen in Equation (19). Consequently, if we want to follow this type of proof, we observe that the girth increases if the norm decreases. So, in order to obtain a larger girth, we need to choose a and b in such a way that the norms of the resulting matrices are as small as possible. By computing $A_a^\top A_a$ respectively $B_b^\top B_b$ and their eigenvalues, we get

$$\|A_a\|^2 = \frac{a^2 + a\sqrt{a^2+4} + 2}{2} \quad \text{and} \quad \|B_b\|^2 = \frac{b^2 + b\sqrt{b^2+4} + 2}{2}.$$

Hence, to minimize the norm, a and b must be chosen as small as possible. Consequently, if we were able to show that there does not exist any nontrivial multiplicative relation between A_a and B_b for $a = b = 1$, we would obtain an even better bound than the one in Margulis' construction. However, if we choose $a = b = 1$, then the statement in (15) does not hold anymore (i.e. in general $A_1^k(E) \not\subseteq F$). In fact, this reasoning in (15) only works if $a, b \geq 2$.
Moreover, we can even show directly that there exists a nontrivial multiplicative relation between A_1 and B_1 because

$$A_1 B_1 A_1^{-1} B_1 A_1 B_1^{-1} = \mathrm{Id}_2.$$

Thus, choosing $a = b = 1$ does not lead to an improvement of Margulis' bound. Another option would be to choose $a = 2$ and $b = 1$ or the other way round. However, this does not work either as

$$A_2 B_1 A_2^{-1} B_1 A_2^{-1} B_1 A_2^{-1} B_1 A_2^{-2} = \mathrm{Id}_2$$

and for $a = 1$ and $b = 2$ the same relation holds.

Therefore, if we only allow $a, b \in \mathbb{N}$, the best choice for a and b is indeed $a = b = 2$.

Another attempt is to choose $a = b$ in $\mathbb{Z}[i]$ or in $\mathbb{Z}[\sqrt{d}]$ for some square-free $d \in \mathbb{N}$, i.e. we consider matrices A_a, B_b in $\mathrm{SL}_2(\mathbb{Z}[i])$ respectively in $\mathrm{SL}_2(\mathbb{Z}[\sqrt{d}])$. We still want to keep the norms of A_a and B_b as small as possible, so $|a|$ and $|b|$ must be smaller than 2. Otherwise, the construction thus obtained would not be an improvement of the one with A_2 and B_2. However, there exists a nontrivial multiplicative relation between A_a and B_b in all of the following cases

$$\begin{aligned}
a = b = i: &\quad A_a B_b A_a^2 B_b^{-1} A_a^{-1} B_b^{-2} = \mathrm{Id}_2,\\
a = b = \sqrt{2}: &\quad A_a B_b^{-1} A_a B_b^{-1} A_a^2 B_b^{-1} A_a B_b^{-1} A_a^{-1} = \mathrm{Id}_2,\\
a = b = \sqrt{3}: &\quad A_a B_b^{-1} A_a B_b^{-1} A_a B_b^{-1} = \mathrm{Id}_2.
\end{aligned}$$

There is also the option to choose a and b differently, for example set $a = 1$ and choose b again in $\{i, \sqrt{2}, \sqrt{3}\}$. Nevertheless, there exist nontrivial multiplicative relations between A_a and B_b (of length 24 respective length 18) in all of these cases

$$\begin{aligned}
b = i: &\quad (A_1 B_b A_1^{-1} B_b^{-1} A_1 B_b)^4 = \mathrm{Id}_2\\
b = \sqrt{2}: &\quad A_1 B_b^{-1} A_1 B_b A_1^{-1} B_b^{-1} A_1^{-1} B_b A_1^{-2} B_b A_1 B_b A_1^{-1} B_b A_1^{-1} B_b A_1^{-1} = \mathrm{Id}_2\\
b = \sqrt{3}: &\quad A_1^{-1} B_b A_1^{-1} B_b^{-1} A_1 B_b^{-1} A_1 B_b A_1 B_b^{-1} A_1 B_b A_1 B_b^{-1} A_1 B_b A_1^{-1} B_b A_1^{-1} B_b^{-1} A_1^{-1} B_b A_1^{-1} B_b^{-1} = \mathrm{Id}_2.
\end{aligned}$$

The second and third relations have been found by computing the result of the multiplication of all possible combinations of n matrices in S (where $n = 1, 2, 3, \dots$). Consequently, there does not exist any shorter relation between those A_1 and B_b. The used Python code is similar to the one presented in Section 7.1 in the appendix.

Furthermore, we could consider two matrices A, B which are not necessarily triangular matrices.

Lemma 4.6. [9, Section 2] *Let*

$$A := \begin{pmatrix} -a & b \\ -c & d \end{pmatrix}, \quad B := \begin{pmatrix} -e & -f \\ g & h \end{pmatrix}$$

with $a, \dots, h \in \mathbb{N}_0$, $\det(A) = \det(B) = 1$ *and* $|\mathrm{Tr}(A)|, |\mathrm{Tr}(B)| \geq 2$. *Then,* A *and* B *generate a free group.*

The proof can be found in [9, p. 162].
Again, we want to find such A and B whose norms are as small as possible. Unfortunately, we will see in Corollary 4.7 that the minimum norm is equal to $1 + \sqrt{2}$ which is exactly the norm of A_2 and B_2. Therefore, these matrices do not lead to an improvement of the bound.

Corollary 4.7. *Let* A *and* B *be as in Lemma 4.6. Then,* $\|A\| \geq 1 + \sqrt{2}$ *and* $\|B\| \geq 1 + \sqrt{2}$.

Proof. Without loss of generality, we only consider $\|A\|$ (as $B^\top$ is of the same form as A and $\|B^\top\| = \|B\|$ by Lemma 4.4 (i)). By Lemma 4.4 (ii) and (iv) it holds that $\|A\|^2 = \max\{|\lambda_1|, |\lambda_2|\}$ where λ_1, λ_2 are the eigenvalues of $A^\top A$. Thus, $\|A\|^2$ is equal to

$$\frac{a^2 + b^2 + c^2 + d^2 + \sqrt{a^4 + b^4 + c^4 + d^4 + 2a^2b^2 + 2a^2c^2 - 2a^2d^2 + 2b^2d^2 + 2c^2d^2 - 2b^2c^2 + 8abcd}}{2}. \tag{20}$$

Note that $c \neq 0$ as otherwise $\det(A) = -ad = 1$ which is not possible. The above expression can be estimated by using three relations which we derive now. First, we recall that $|\mathrm{Tr}(A)| = |d - a| \geq 2$. Using this and the fact that a and d are mutually interchangeable in Equation (20), we can assume without loss of generality that

$$d \geq a + 2. \tag{21}$$

Second, we know that $\det(A) = bc - ad = 1$ which becomes (by using that $c \neq 0$)

$$b = \frac{1 + ad}{c} \overset{\text{by (21)}}{\geq} \frac{1 + a(a+2)}{c} = \frac{(a+1)^2}{c}. \tag{22}$$

Third, it additionally follows from $\det(A) = 1$ that $-2 \cdot \det(A)^2 = -2$ which is equivalent to

$$-2c^2b^2 + 4abcd - 2a^2d^2 = -2. \tag{23}$$

These relations in (21), (22) and (23) lead to the following expression, which depends only on a and c,

$$\|A\|^2 \geq \frac{a^2 + \frac{(a+1)^4}{c^2} + c^2 + (a+2)^2 + \sqrt{m}}{2}$$

where

$$\begin{aligned} m := a^4 + \frac{(a+1)^8}{c^4} + 2a^2\frac{(a+1)^4}{c^2} + c^4 + 2a^2c^2 + (a+2)^4 \\ + 2(a+2)^2\frac{(a+1)^4}{c^2} + 2c^2(a+2)^2 + 4a(a+1)^2(a+2)^2 - 2. \end{aligned}$$

We see that it attains the smallest possible value if we set $a = 0$ which results in

$$\|A\|^2 \geq \frac{c^{-2} + c^2 + 4 + \sqrt{c^{-4} + c^4 + 16 + 8c^{-2} + 8c^2 - 2}}{2}.$$

Now, note that if $a = 0$, then $\det(A) = bc = 1$ which implies that $b = c = 1$. So, plugging $c = 1$ in the above inequality gives

$$\|A\|^2 \geq 3 + 2\sqrt{2} = (1 + \sqrt{2})^2.$$

□

4.1.3 Calculation of Exact Girth

The actual girth of X_p is still not known as Theorem 4.2 only gives a lower bound and does not say anything about its accuracy. Consequently, the girth might actually be (much) larger than the bound.

We therefore wrote a programme on Python which calculates the exact girth of X_p for any chosen odd prime p. The whole Python code can be found in the appendix in Section 7.1. We ran this programme for all odd primes up to 463. The obtained results are surprising as the computed girth often differs considerably from the computed lower bound. Figure 5 shows a plot of these results (the exact results can be found in the appendix in Section 7.2). The used formula for the bound appears during the proof of the asymptotic lower bound in [11, p. 74]

$$g(X_p) \geq 2\log_{1+\sqrt{2}}(p/2) - 1. \tag{24}$$

As the girth is naturally an integer, we applied the ceiling function to the computed bounds to obtain useful values; this bound is labeled as Margulis' bound in Figure 5.

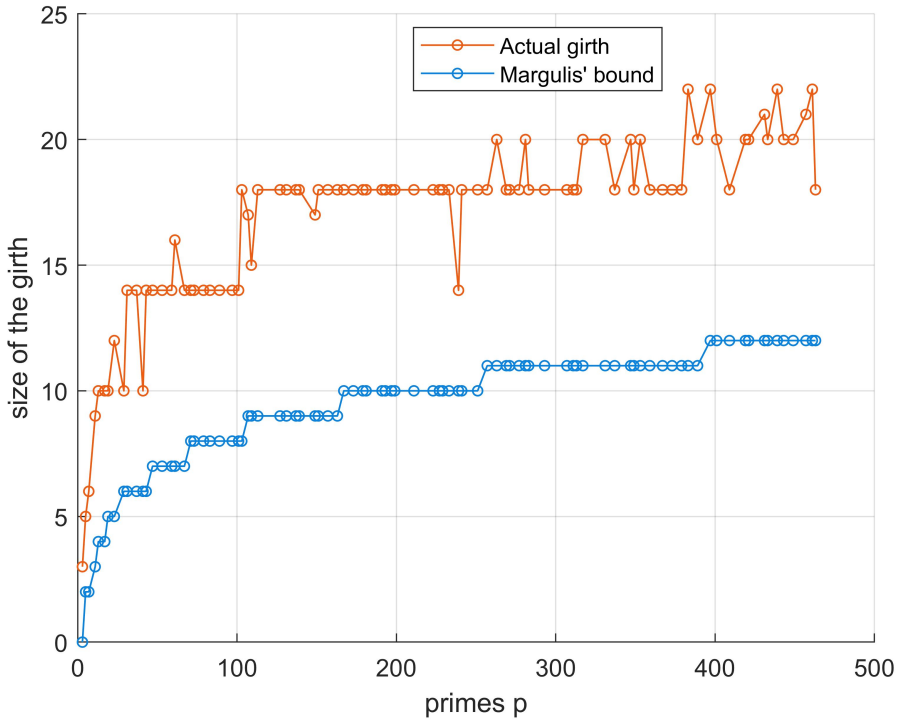

Figure 5: Plot of the obtained girths compared to the respective lower bounds for prime numbers up to 463.

It is recognisable that the difference between the actual girth and Margulis' bound is quite large in all considered cases. For some primes p, the girth is even twice as large as the lower bound (for example for $p = 19$, $p = 47$ or $p = 383$). The relative differences between the girth and the bound vary between 40% and 200% (see Figure 6) and the average relative difference is $86,5\%$.

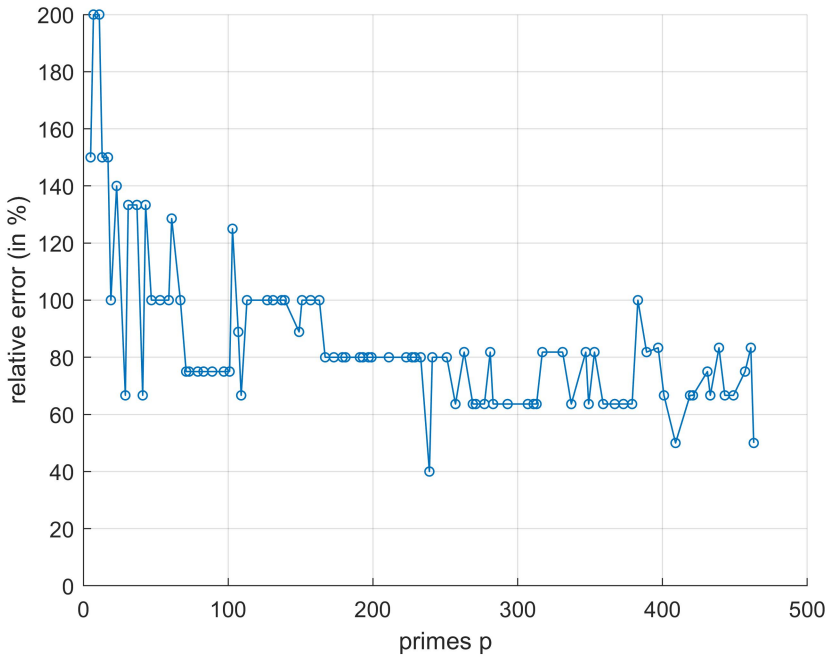

Figure 6: Plot of the relative differences between girth and lower bound for prime numbers up to 463.

As these differences are so large, we wondered if it might be possible to improve Margulis' lower bound. Unfortunately, we could not find an asymptotically better bound but we found a small improvement of the bound in Equation (24).

For this, let A and B be again defined as at the beginning of Section 4.1, i.e.

$$A := \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

Note that for any $k, n \in \mathbb{Z}$ we have

$$A^k B^n = \begin{pmatrix} 1 & 2k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2n & 1 \end{pmatrix} = \begin{pmatrix} 1+4kn & 2k \\ 2n & 1 \end{pmatrix}. \tag{25}$$

Let now $M_t := A^{k_1} B^{n_1} A^{k_2} B^{n_2} \cdot \ldots \cdot A^{k_t} B^{n_t}$ where $t \in \mathbb{N}_{\geq 2}$, $k_i, n_j \in \mathbb{Z} \setminus \{0\}$ for $2 \leq i \leq t$, $1 \leq j \leq t-1$ and $k_1, n_t \in \mathbb{Z}$. For $t = 1$, $M_t := A^{k_1} B^{n_1}$ with $k_1, n_1 \in \mathbb{Z}$ such that $k_1 \neq 0$ or $n_1 \neq 0$. Furthermore, we denote

$$M_t = \begin{pmatrix} m_{11}^{(t)} & m_{12}^{(t)} \\ m_{21}^{(t)} & m_{22}^{(t)} \end{pmatrix}.$$

Lemma 4.8. *The entries $m_{11}^{(t)}, m_{22}^{(t)}$ on the diagonal of M_t are odd and the remaining entries $m_{12}^{(t)}, m_{21}^{(t)}$ are even for all $t \in \mathbb{N}_{\geq 1}$.*

Proof. We prove this by induction on t.

The base case $t = 1$ follows directly from Equation (25) as this matrix is equal to M_1. So, we assume that the claim holds for an arbitrary but fixed $t \in \mathbb{N}_{\geq 1}$. Now, consider M_{t+1} which is equal to

$$\begin{aligned} M_{t+1} = M_t A^{k_{t+1}} B^{n_{t+1}} &= \begin{pmatrix} m_{11}^{(t)} & m_{12}^{(t)} \\ m_{21}^{(t)} & m_{22}^{(t)} \end{pmatrix} \begin{pmatrix} 1 + 4k_{t+1}n_{t+1} & 2k_{t+1} \\ 2n_{t+1} & 1 \end{pmatrix} \\ &= \begin{pmatrix} m_{11}^{(t)}(1 + 4k_{t+1}n_{t+1}) + 2n_{t+1}m_{12}^{(t)} & 2k_{t+1}m_{11}^{(t)} + m_{12}^{(t)} \\ m_{21}^{(t)}(1 + 4k_{t+1}n_{t+1}) + 2n_{t+1}m_{22}^{(t)} & 2k_{t+1}m_{21}^{(t)} + m_{22}^{(t)} \end{pmatrix}. \end{aligned}$$

Using the induction hypothesis, the claim now follows. □

Let p still be any odd prime. Note that the girth $g(X_p)$ can at most be equal to p as

$$A^p \equiv B^p \equiv \mathrm{Id} \pmod{p}.$$

Lemma 4.9. *Let* $V := M_r = A^{k_1}B^{n_1} \cdot \ldots \cdot A^{k_r}B^{n_r}$, $W := M_s = A^{\tilde{k}_1}B^{\tilde{n}_1} \cdot \ldots \cdot A^{\tilde{k}_s}B^{\tilde{n}_s}$ *where* $r, s \in \mathbb{N}_{\geq 1}$ *are such that* $V \equiv W \pmod{p}$ *but* $V \neq W$. *Then, it holds that* $\max\{\|V\|, \|W\|\} \geq p$.

Proof. For simplicity denote

$$V = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}.$$

We know, by Lemma 4.8, that the diagonal entries of V and W are odd and the others are even. Therefore, $v_{ij} - w_{ij}$ is even for all $i, j \in \{1, 2\}$. In addition, as $V - W \equiv 0 \pmod{p}$ but $V - W \neq 0$, there exist $a, b \in \{1, 2\}$ such that $v_{ab} - w_{ab} \neq 0$. As p is odd and divides $v_{ab} - w_{ab}$ (which is even), it follows that $|v_{ab} - w_{ab}| \geq 2p$. This implies that $\max\{|v_{ab}|, |w_{ab}|\} \geq p$. It then follows by Lemma 4.4 (iii) that $\max\{\|V\|, \|W\|\} \geq p$. □

Corollary 4.10. *Let* $\alpha := 1 + \sqrt{2}$. *Then,* $g(X_p) \geq 2\log_\alpha(p) - 1$.

Proof. Recall the proof of Theorem 4.2 and let $\tilde{s}_i$, for $1 \leq i \leq g = g(X_p)$, be as in this proof. For simplicity, let us use the same notation as above, that is

$$V = \tilde{s}_1 \ldots \tilde{s}_{\lfloor g/2 \rfloor} \quad \text{and} \quad W = \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1} \ldots \tilde{s}_{\lfloor g/2 \rfloor + 1}^{-1}.$$

As V and W satisfy all the conditions of Lemma 4.9, it follows that $\max\{\|V\|, \|W\|\} \geq p$. Analogous to the inequality in (19), we have

$$p \leq \max\{\|V\|, \|W\|\} \leq (1 + \sqrt{2})^{\max\{\lfloor g/2 \rfloor, g - \lfloor g/2 \rfloor\}}.$$

We can check that $\max\{\lfloor g/2 \rfloor, g - \lfloor g/2 \rfloor\} = g - \lfloor g/2 \rfloor$. Using this, the above inequality becomes

$$\alpha^{g - \lfloor g/2 \rfloor} \geq p$$

which is equivalent to

$$g - \lfloor g/2 \rfloor \geq \log_\alpha(p). \tag{26}$$

Moreover, a short computation gives that

$$g \geq 2(g - \lfloor g/2 \rfloor) - 1. \tag{27}$$

Putting the two inequalities in (26) and (27) together, we finally obtain the lower bound

$$g \geq 2\log_\alpha(p) - 1.$$

□

Remark 4.11. The new bound differs from Margulis' bound by $2\log_\alpha(2)$. This results in a difference of either 1 or 2 when applying the ceiling function (this can also be seen in Figure 7 below).

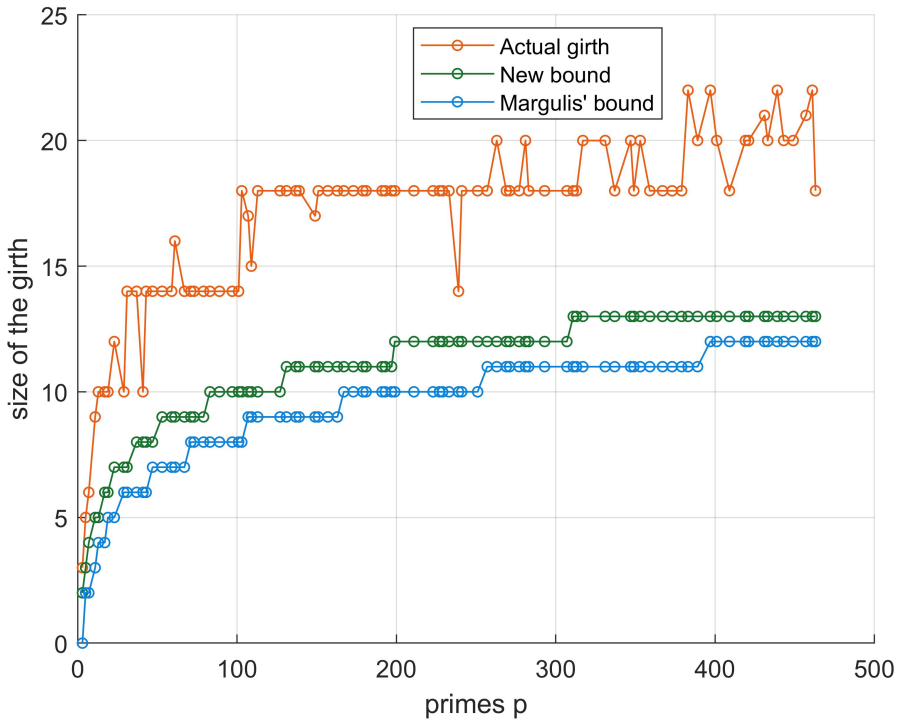

Figure 7: Plot of the new bound compared to Margulis' bound and the actual girth for prime numbers up to 463.

Consequently, the relative differences between the new lower bound and the actual girth also get smaller. Now, these differences vary between $16,66\%$ and 100% and the average relative difference decreases to $56,6\%$. Figure 8 shows a comparison of these relative differences.
However, it can be checked that this new bound leads to the same asymptotic bound as Margulis' bound. Hence, even though the average relative difference decreased, asymptotically it is the same bound as before.

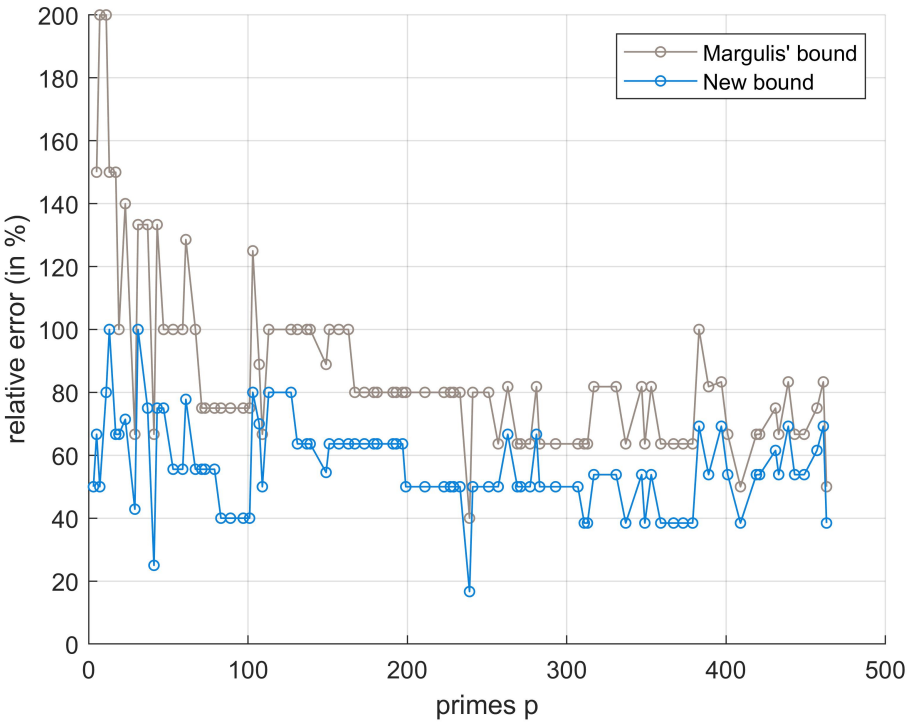

Figure 8: Plot of the relative differences between girth and the two lower bounds for prime numbers up to 463.

Therefore, we wondered if it might be possible to improve this bound further. Looking again at the proof of Theorem 4.2, we see that we could maybe improve the estimates of the operator norm of the products $\tilde{s}_1 \dots \tilde{s}_{\lfloor g/2 \rfloor}$ and $\tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2 \rfloor+1}^{-1}$ respectively the norm of the difference of these two products.

However, we can not improve the lower bound

$$\|\tilde{s}_1 \dots \tilde{s}_{\lfloor g/2 \rfloor} - \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1}\dots\tilde{s}_{\lfloor g/2 \rfloor+1}^{-1}\| \geq 2p$$

further (in fact, this bound is already an improvement; see Lemma 4.9). Indeed, we will see later that there are prime numbers p where this norm is exactly equal to $2p$.
In addition, according to Lemma 4.12, we can not improve this upper bound

$$\|\tilde{s}_1\tilde{s}_2\dots\tilde{s}_{g-1}\tilde{s}_g\| \leq (1+\sqrt{2})^g$$

either.

Lemma 4.12. *Let $M = (AB)^n$ or $M = (AB)^n A$ for some $n \in \mathbb{N}$. Then, $\|M\| = (1+\sqrt{2})^{2n}$ respectively $\|M\| = (1+\sqrt{2})^{2n+1}$.*

Proof. First note that by definition $A = B^\top$ respectively $B = A^\top$. Using this, we have, for $M = (AB)^n$, that

$$M^\top M = ((AB)^n)^\top \cdot M = (B^\top A^\top)^n \cdot M = (B^\top B)^n \cdot (B^\top B)^n = (B^\top B)^{2n}.$$

It follows that the eigenvalues of $M^\top M$ are equal to the eigenvalues of $B^\top B$ to the power of $2n$. By part (iv) of Lemma 4.4, we then have

$$\|M^\top M\| = \|(B^\top B)^{2n}\| = \|B^\top B\|^{2n}$$

which implies (due to part (ii) of Lemma 4.4)

$$\|M\| = \|M^\top M\|^{1/2} = \|B^\top B\|^n = \|B\|^{2n} = (1+\sqrt{2})^{2n}.$$

The second case is similar as

$$M^\top M = A^\top (B^\top A^\top)^n \cdot M = A^\top (AA^\top)^n \cdot (AA^\top)^n A = (A^\top A)^{2n+1}$$

which leads to $\|M\| = (1+\sqrt{2})^{2n+1}$. □

Remark 4.13. The same also holds for $M = (BA)^n$, $M = (BA)^n B$ and for the inverses of these matrix products.

The upper bound could thus only be improved if we could show that the matrix products $s_1 s_2 \dots s_g \equiv \mathrm{Id} \pmod p$ with $s_i \in \{A_p, B_p, A_p^{-1}, B_p^{-1}\}$ are never of the form as M in Lemma 4.12. However, this is not true as for $p = 41$ one of the shortest nontrivial multiplicative relations is

$$(BA)^5 \equiv \mathrm{Id} \pmod{41}.$$

So, the only inequality which might be possible to improve is

$$\max\{\|\tilde{s}_1 \dots \tilde{s}_{\lfloor g/2 \rfloor}\|, \|\tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1} \dots \tilde{s}_{\lfloor g/2 \rfloor+1}^{-1}\|\} \geq p. \tag{28}$$

We adapted the previously mentioned Python code such that it does not only compute the girth but also returns all possible combinations of $\tilde{s}_1, \dots, \tilde{s}_g$ such that $\tilde{s}_1 \cdot \ldots \cdot \tilde{s}_g \equiv 1 \pmod p$. By writing another short Matlab code, we then computed $V := \tilde{s}_1 \dots \tilde{s}_{\lfloor g/2 \rfloor}$ and $W := \tilde{s}_g^{-1}\tilde{s}_{g-1}^{-1} \dots \tilde{s}_{\lfloor g/2 \rfloor+1}^{-1}$ for all of these combinations and returned the matrices V and W for which $\max\{\|V\|, \|W\|\}$ is minimal. We did this for all primes between $p = 7$ and $p = 199$ ($p = 3$ and $p = 5$ are special cases as the girths are $g = 3$ respectively $g = 5$). The results suggest that it might indeed be possible to improve the lower bound in (28). More precisely, the results suggest that $\max\{\|V\|, \|W\|\} \geq 2p$.

Observation 4.14. Let p be a prime such that $7 \leq p \leq 199$ and let the girth g be even. Then, there exist $\tilde{s}_1, \dots, \tilde{s}_g \in \{A^{\pm 1}, B^{\pm 1}\}$ such that $\tilde{s}_1 \dots \tilde{s}_g \equiv 1 \pmod p$ (but $\tilde{s}_1 \dots \tilde{s}_g \neq 1$) and V, W are such that $\|V\| = \|W\| \geq 2|\lambda|p$ for some $\lambda \in \mathbb{Z} \setminus \{0\}$. Moreover, $\max\{\|V\|, \|W\|\}$ is minimal among all other V and W with the above properties and one of the following two options is fulfilled.

1) If $V = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z})$, then $W = \begin{pmatrix} a & -c \\ -b & d \end{pmatrix}$ and $V - W = \begin{pmatrix} 0 & 2\lambda p \\ 2\lambda p & 0 \end{pmatrix}$.

2) $V = W^\top$ and $V - W = \begin{pmatrix} 0 & 2\lambda p \\ -2\lambda p & 0 \end{pmatrix}$.

Let us look at two examples, one for each of the two cases.

Example 4.15. For $p = 83$, we have that $g = 14$ and $\|V\| = \|W\| = 166,006$ where

$$V = \begin{pmatrix} -71 & 126 \\ 40 & -71 \end{pmatrix}, W = \begin{pmatrix} -71 & -40 \\ -126 & -71 \end{pmatrix} \text{ and hence } V - W = \begin{pmatrix} 0 & 166 \\ 166 & 0 \end{pmatrix}.$$

For $p = 29$, we have that $g = 10$ and $\|V\| = \|W\| = 58,017$ where

$$V = \begin{pmatrix} 21 & 50 \\ -8 & -19 \end{pmatrix}, W = \begin{pmatrix} 21 & -8 \\ 50 & -19 \end{pmatrix} \text{ and hence } V - W = \begin{pmatrix} 0 & 58 \\ -58 & 0 \end{pmatrix}.$$

If we wanted to show this for all primes $p \geq 7$ with even girth, we would need to prove the following three conjectures.

1. For every $p \geq 7$ with even girth $g = g(X_p)$ there exist V and W as in 1) or 2).
2. For these V and W $\max\{\|V\|, \|W\|\}$ is minimal,
3. and $\max\{\|V\|, \|W\|\} \geq 2p$.

Assuming the first conjecture, we can show the third one as follows.

Lemma 4.16. *Assume that for every $p \geq 7$ with even girth $g = g(X_p)$ there exist V and W as in 1) or 2). Then, max$\{\|V\|, \|W\|\} = \|V\| = \|W\| \geq 2p$.*

Proof. Let $a, b, c, d \in \mathbb{Z}$ such that b, c are even and a, d are odd (see Lemma 4.8) and write

$$V = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

We will look at cases 1) and 2) separately. Let us start with case 1), so we assume that

$$W = \begin{pmatrix} a & -c \\ -b & d \end{pmatrix} \text{ and consequently } V - W = \begin{pmatrix} 0 & b+c \\ b+c & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2\lambda p \\ 2\lambda p & 0 \end{pmatrix}$$

for some $\lambda \in \mathbb{Z} \setminus \{0\}$. We remark that then $\|V\| = \|W\|$ by Lemma 4.4, part (ii) and (iv), and by the fact that $\det(V) = \det(W)$. Hence, from now on we only consider $\|V\|$. Furthermore, note that if $\operatorname{sgn}(b) \neq \operatorname{sgn}(c)$, then $\max\{|b|, |c|\} \geq 2p$ (as $b + c = 2\lambda p$). It then follows by part (iii) of Lemma 4.4 that $\|V\| \geq 2p$. Hence, we can assume that $\operatorname{sgn}(b) = \operatorname{sgn}(c)$, i.e. $bc \geq 0$ and therefore also $ad \geq 0$ (as $\det(V) = ad - bc = 1$).
Since $b + c = 2\lambda p$, we can write $c = 2\lambda p - b$. We further denote

$$s := \left\lceil \sqrt{bc + 1} \right\rceil = \left\lceil \sqrt{b(2\lambda p - b) + 1} \right\rceil.$$

To illustrate the method of this proof, let us assume for the moment that $|s-a| = |s-d|$. In other words, there exists some $k \in \mathbb{Z}$ such that $a = s+k$, $d = s-k$ and $ad - bc = 1$. Without loss of generality, we can assume that a and d are both positive as we will only consider their product ad and additionally, we know that $ad = bc + 1$ and that the product bc is non-negative. Then, the matrix V is now defined as

$$V = \begin{pmatrix} s+k & b \\ 2\lambda p - b & s-k \end{pmatrix}.$$

Its determinant is equal to 1 which leads to

$$(s+k)(s-k) - b(2\lambda p - b) = 1$$

and this is equivalent to

$$s^2 - 2b\lambda p + b^2 = k^2 + 1. \tag{29}$$

The norm of V can be calculated by computing the square root of the maximal eigenvalue of $V^\top V$ (see Lemma 4.4). It can be checked that this eigenvalue (and thus also $\|V^\top V\|$) is equal to

$$s^2 - 2b\lambda p + b^2 + k^2 + 2\lambda^2 p^2 + 2 \cdot \sqrt{b^2k^2 + \lambda^4 p^4 - 2b\lambda^3 p^3 + b^2\lambda^2 p^2 + k^2\lambda^2 p^2 - 2bk^2\lambda p + k^2 s^2 + \lambda^2 p^2 s^2}.$$

We can replace the first three summands using (29) and then further simplify it by applying (29) again as

$$\begin{aligned}
&k^2 + 1 + k^2 + 2\lambda^2 p^2 + 2 \cdot (k^2 (\underbrace{b^2 - 2b\lambda p + s^2}_{=k^2+1}) + \lambda^4 p^4 - 2b\lambda^3 p^3 + b^2\lambda^2 p^2 + k^2\lambda^2 p^2 + \lambda^2 p^2 s^2)^{(1/2)} \\
&= 2k^2 + 1 + 2\lambda^2 p^2 + 2 \cdot (k^2(k^2+1) + \lambda^4 p^4 - 2b\lambda^3 p^3 + b^2\lambda^2 p^2 + k^2\lambda^2 p^2 + \lambda^2 p^2 s^2)^{(1/2)} \\
&= 2k^2 + 1 + 2\lambda^2 p^2 + 2 \cdot (k^4 + k^2 + \lambda^4 p^4 + \lambda^2 p^2 (\underbrace{-2b\lambda p + b^2 + s^2}_{=k^2+1}) + k^2\lambda^2 p^2)^{(1/2)} \\
&= 2k^2 + 1 + 2\lambda^2 p^2 + 2 \cdot (k^4 + k^2 + \lambda^4 p^4 + \lambda^2 p^2 (k^2+1) + k^2\lambda^2 p^2)^{(1/2)} \\
&= \underbrace{2k^2 + 1}_{\geq 0} + 2\lambda^2 p^2 + 2 \cdot (\underbrace{k^4 + k^2 + \lambda^2 p^2 (2k^2+1)}_{\geq 0} + \lambda^4 p^4)^{(1/2)} \\
&\geq 2\lambda^2 p^2 + 2 \cdot \sqrt{\lambda^4 p^4} = 4\lambda^2 p^2.
\end{aligned}$$

It hence follows by part (ii) of Lemma 4.4 that $\|V\| \geq \sqrt{4\lambda^2 p^2} = 2|\lambda| p \geq 2p$.
In the general case, we write $a = s + k_1$ and $d = s - k_2$ where $k_1, k_2 \in \mathbb{Z}$ are such that $ad - bc = 1$ (we still assume without loss of generality that a and d are positive). The proof of the general case follows the exact same strategy as before but requires more and especially longer calculations. Therefore, we only present a sketch of this part of the proof, starting with a claim.

Claim 4.17. If $k_1 k_2 \neq 0$, then $\operatorname{sgn}(k_1) = \operatorname{sgn}(k_2)$.

Proof. We assume that $k_1 k_2 \neq 0$, hence k_1 and k_2 are nonzero integers. Let us assume by contradiction that $\operatorname{sgn}(k_1) \neq \operatorname{sgn}(k_2)$. We first consider the case where $k_1 > 0$ and $k_2 < 0$. Then, $a = s + k_1 > s$ and $d = s - k_2 > s$ which implies that $ad > s^2 \geq bc + 1$. This is a contradiction as $ad = bc + 1$. Hence, we assume that $k_1 < 0$ and $k_2 > 0$. Similarly, we obtain that $ad \leq (s-1)^2 < bc + 1$ which is again a contradiction. Thus, we conclude that $\operatorname{sgn}(k_1) = \operatorname{sgn}(k_2)$. □

Then, we again compute the maximal eigenvalue of $V^\top V$ and use Claim 4.17 as well as

$$s^2 + k_1 s - k_2 s - 2b\lambda p + b^2 = k_1 k_2 + 1,$$

which follows from the fact that $ad - bc = 1$, to obtain the expression

$$\frac{1}{2} \cdot \Big(\underbrace{2k_1k_2 + 2 + k_1^2 + k_2^2}_{\geq 0} + 4\lambda^2 p^2 + \sqrt{\underbrace{2k_1k_2(2k_1^2 + 3k_1k_2 + 4) + 4k_1^2 + k_1^4 + k_2^4 + 4k_2^2}_{\geq 0} + 16\lambda^4 p^4 + \underbrace{8\lambda^2 p^2 (2k_1k_2 + 2 + k_1^2 + k_2^2)}_{\geq 0}} \Big).$$

Hence, it follows that

$$\|V^\top V\| \geq \frac{1}{2} \cdot \left(4\lambda^2 p^2 + \sqrt{16\lambda^4 p^4} \right) = 4\lambda^2 p^2$$

which implies as above that $\|V\| \geq 2p$.

For case 2), we assume that

$$W = V^\top = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \text{ and consequently } V - W = \begin{pmatrix} 0 & b-c \\ c-b & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2\lambda p \\ -2\lambda p & 0 \end{pmatrix}$$

for some $\lambda \in \mathbb{Z} \setminus \{0\}$. We again only consider $\|V\|$ as $\|V\| = \|W\|$ (by part (i) of Lemma 4.4). Without loss of generality, we can assume that $\operatorname{sgn}(b) \neq \operatorname{sgn}(c)$ and that $bc \neq 0$ (otherwise $\max\{|b|, |c|\} \geq 2p$). Consequently, this means that $bc < 0$. Moreover, as $b - c = 2\lambda p$, we can write $c = b - 2\lambda p$ and further define

$$s := \left\lceil \sqrt{|bc+1|} \right\rceil = \left\lceil \sqrt{|b(b-2\lambda p)+1|} \right\rceil = \left\lceil \sqrt{-b(b-2\lambda p)-1} \right\rceil = \left\lceil \sqrt{2b\lambda p - b^2 - 1} \right\rceil.$$

As in case 1), we then write $a = s + k_1$ and $d = k_2 - s$ where $k_1, k_2 \in \mathbb{Z}$ are such that $ad - bc = 1$. Recall (by Lemma 4.8) that a and d are odd, in particular they are nonzero, and that $ad = bc + 1$. As in addition $bc < 0$, we can assume without loss of generality that a is positive and d is negative (since we will only consider the product ad). Before computing the eigenvalues of $V^\top V$, we first need two additional claims.

Claim 4.18. It holds that $2b\lambda p - b^2 - 1$ is not a square number, i.e. $a \neq d$.

Proof. Note first that, as $|ad| = |bc+1| = 2b\lambda p - b^2 - 1$, $a = d$ is only possible if $2b\lambda p - b^2 - 1$ is a square number. Recall that b is even and let us assume by contradiction that there exists an integer $b \in 2\mathbb{Z}$ such that $2b\lambda p - b^2 - 1$ is a square. Therefore, there exists another integer $x \in \mathbb{Z}$ such that $x^2 = 2b\lambda p - b^2 - 1$. As b is even, we can further write $b = 2\tilde{b}$ for some $\tilde{b} \in \mathbb{Z}$, which leads to

$$x^2 = 4\tilde{b}\lambda p - 4\tilde{b}^2 - 1. \tag{30}$$

We can check that square numbers are always equal to 0 or 1 modulo 4. Therefore, reducing Equation (30) modulo 4 gives

$$x^2 \equiv 3 \pmod 4$$

which is a contradiction. Thus, we conclude that $2b\lambda p - b^2 - 1$ is not a square number respectively that $a \neq d$. □

Now, using $a = s + k_1$ and $b = k_2 - s$ in $ad - bc = 1$ yields

$$s^2 + k_1 s - k_2 s - 2b\lambda p + b^2 = k_1 k_2 - 1. \tag{31}$$

Using this equation, we can prove the following claim.

Claim 4.19. It holds that $k_1 k_2 \neq 0$, i.e. k_1 and k_2 are both nonzero integers. Moreover, $\operatorname{sgn}(k_1) = \operatorname{sgn}(k_2)$.

Proof. Let us first show that $k_1 k_2 \neq 0$. We know by Claim 4.18 that $2b\lambda p - b^2 - 1$ is not a square number, so at least one of the two integers k_1 and k_2 is different from zero. We assume by contradiction that $k_1 k_2 = 0$ and first consider the case where $k_1 = 0$ and $k_2 \neq 0$. As $a = s + k_1$ is odd, this is only possible if s is odd. As $d = k_2 - s$ is odd too, it follows that $|k_2| \geq 2$. Let $x := \sqrt{|b^2 - 2b\lambda p + 1|} = \sqrt{2b\lambda p - b^2 - 1}$, then $s = \lceil x \rceil$. Plugging $k_1 = 0$ and $x^2 = 2b\lambda p - b^2 - 1$ into Equation (31) gives $s^2 - x^2 = k_2 s$ or equivalently

$$\underbrace{(s - x)}_{<1}\underbrace{(s + x)}_{<2s} = k_2 s.$$

This implies that $k_2 < 2$. Moreover, it holds that $k_2 > 0$ as $s^2 - x^2 > 0$. Thus, we arrive at a contradiction because we said above that $|k_2| \geq 2$. Consequently, $k_1 \neq 0$. The second case where $k_2 = 0$ can be done similarly, so we conclude that also $k_2 \neq 0$.
For the second part of the claim, let us assume by contradiction that $\operatorname{sgn}(k_1) \neq \operatorname{sgn}(k_2)$. Hence, let first $k_1 > 0$ and $k_2 < 0$. Then, $a = s + k_1 > s$ and $d = k_2 - s < -s$ respectively $|d| > s$. Consequently,

$$|ad| = |a||d| > s^2 > x^2 = |bc + 1|$$

which is a contradiction as $ad = bc + 1$. Assuming that $k_1 < 0$ and $k_2 > 0$ leads to $|ad| < x^2 = |bc + 1|$ which is also a contradiction. We hence conclude that $\operatorname{sgn}(k_1) = \operatorname{sgn}(k_2)$. □

Now, we again compute the maximal eigenvalue of $V^\top V$ (which is the same as in case 1)) and use (31) to arrive at

$$\frac{1}{2}\Big(\underbrace{2k_1k_2+k_1^2+k_2^2-2}_{>0}+4\lambda^2p^2+\sqrt{\underbrace{k_1^4+k_2^4+2k_1k_2(2k_1^2+6k_1k_2-8)}_{>0}+16\lambda^4p^4+16\lambda^2p^2\underbrace{(k_1k_2-1)}_{\geq 0}+\underbrace{(4k_1^2+4k_2^2)}_{>0}(2\lambda^2p^2-1)}\Big).$$

Using Claim 4.19, we obtain again that $\|V^\top V\| \geq 4\lambda^2p^2$ which implies that $\|V\| \geq 2p$. Therefore, we can conclude that $\max\{\|V\|, \|W\|\} \geq 2p$. □

Remark 4.20. Even though Observation 4.14 only holds if the girth is even, the results suggest that the bound could also be improved if the girth is odd. In this case (still for $7 \leq p \leq 199$), it also holds that $\max\{\|V\|, \|W\|\} \geq 2p$.

Remark 4.21. We know that $\max\{\|V\|, \|W\|\} \geq 2p$ for all $7 \leq p \leq 199$. Similar to Corollary 4.10, this implies for these primes p that we can further improve the lower bound to obtain $g(X_p) \geq 2\log_{1+\sqrt{2}}(2p) - 1$ (as we get an improvement for the bound in (28), respectively we get a higher lower bound for the first inequality in (19)). Consequently, if we could prove Observation 4.14 for all prime numbers, we would obtain a further improvement of Margulis' lower bound.

4.2 Construction by Lubotzky, Phillips and Sarnak

The aim of this subsection is to describe the construction of families of Ramanujan graphs with large girth as presented by Lubotzky, Phillips and Sarnak [8]. The material has mainly been taken from [8] and [3, Chapters 2 and 4].

4.2.1 Quaternions and Sums of Four Squares

The construction of these Ramanujan graphs is based on quaternions which will be introduced in this section.

Definition 4.22. [3, Definition 2.5.1] Let R be any commutative ring with 1. The *Hamilton quaternion algebra* over R, denoted by $\mathbb{H}(R)$, is the associative algebra with unit such that

(i) $\mathbb{H}(R)$ is the free R-module over the symbols $1, i, j, k$, i.e.

$$\mathbb{H}(R) := \{a_0 + a_1i + a_2j + a_3k \mid a_0, a_1, a_2, a_3 \in R\},$$

(ii) 1 is the multiplicative unit,

(iii) $i^2 = j^2 = k^2 = -1$,

(iv) $ij = -ji = k$, $jk = -kj = i$, $ki = -ik = j$.

Definition 4.23. [3, p. 57] Let $\alpha := a_0 + a_1i + a_2j + a_3k$ be a quaternion over R. Then, its *conjugate* quaternion is $\bar{\alpha} := a_0 - a_1i - a_2j - a_3k$ and the *norm* of α is $N(\alpha) := \alpha\bar{\alpha} = \bar{\alpha}\alpha = a_0^2 + a_1^2 + a_2^2 + a_3^2$.

From now on, we will only consider the ring $R = \mathbb{Z}$ and the elements of $\mathbb{H}(\mathbb{Z})$ will be called *integer quaternions*. Note that a natural number is the sum of four squares if and only if it is the norm of some quaternion in $\mathbb{H}(\mathbb{Z})$. Moreover, we can check that $N(\alpha_1)N(\alpha_2) = N(\alpha_1\alpha_2)$ for all $\alpha_1, \alpha_2 \in \mathbb{H}(\mathbb{Z})$.

Remark 4.24. [3, p. 59] Let $\alpha \in \mathbb{H}(\mathbb{Z})$ be an invertible element. Then, $1 = N(1) = N(\alpha\alpha^{-1}) = N(\alpha)N(\alpha^{-1})$ which implies that $N(\alpha) = 1$. Therefore, we see that $\pm 1, \pm i, \pm j, \pm k$ are the only invertible elements in $\mathbb{H}(\mathbb{Z})$.

Now, we consider all integer quaternions α with $N(\alpha) = p^l$ for some odd prime p and some $l \in \mathbb{N}$. The number of such quaternions is given by the following theorem which will be stated without proof (a proof can be found in [3, Section 2.4]).

Theorem 4.25. [3, Theorem 2.4.1] *Let $n \in \mathbb{N}$ be odd. Then, the number of $(x_0, x_1, x_2, x_3) \in \mathbb{Z}^4$ which fulfil the equation $x_0^2 + x_1^2 + x_2^2 + x_3^2 = n$ is equal to $8\sum_{d|n} d$.*

Let $p > 2$ be a prime. The number of integer quaternions $\alpha = a_0 + a_1 i + a_2 j + a_3 k$ with $N(\alpha) = p$ is equal to the number of solutions of

$$a_0^2 + a_1^2 + a_2^2 + a_3^2 = p. \tag{32}$$

By Theorem 4.25 this equation has $8(1 + p)$ solutions.
Recall that a square x^2 is either equal to 0 (mod 4) (if x is even) or equal to 1 (mod 4) (if x is odd). Consequently, if $p \equiv 1 \pmod 4$, exactly one a_i is odd and if $p \equiv 3 \pmod 4$, exactly one a_i is even. In either case, the parity of exactly one term a_i^* is different from the others.

Moreover, note that if $u \in \{\pm 1, \pm i, \pm j, \pm k\}$ is a unit and α is such that $N(\alpha) = p$, then $N(u\alpha) = p$ too. There are two cases.

a) If $a_i^* \neq 0$, then exactly one of the eight solutions $u\alpha$ has $|a_i^*|$ as its first component (follows from Definition 4.22 (iii) and (iv)).

b) If $a_i^* = 0$, then for the same reason there exists a unit u such that the first component of both $u\alpha$ and $-u\alpha$ is equal to 0. In this case, we can define one of them to be distinguishable.

In this way, we obtain $p+1$ different solutions of (32) where the corresponding integer quaternion α fulfils either $\alpha \equiv 1 \pmod 2$ or $\alpha \equiv i + j + k \pmod 2$. In addition, we remark that in case a) α and $\bar{\alpha}$ are defined as two different solutions whereas in case b) only one of them is considered. Finally, the set of solutions to (32) is given by

$$S_p := \{\alpha_1, \bar{\alpha_1}, \ldots, \alpha_s, \bar{\alpha_s}, \beta_1, \ldots, \beta_t\} \tag{33}$$

where the α's belong to case a) and the β's to case b).

Before we can come over to the construction of the graphs, we need one more property of quaternions.

Proposition 4.26. [3, Proposition 2.5.2] *Let K be a field with characteristic different from 2. Suppose that there exist $x, y \in K$ with $x^2 + y^2 + 1 = 0$. Then, $\mathbb{H}(K)$ is isomorphic to $M_2(K)$ (the algebra of 2-by-2 matrices over K).*

Proof. The proof follows [3, p. 58].
We define the map $\psi : \mathbb{H}(K) \to M_2(K)$ by

$$\psi(a_0 + a_1 i + a_2 j + a_3 k) := \begin{pmatrix} a_0 + a_1 x + a_3 y & -a_1 y + a_2 + a_3 x \\ -a_1 y - a_2 + a_3 x & a_0 - a_1 x - a_3 y \end{pmatrix}. \tag{34}$$

Note that $\mathbb{H}(K)$ and $\mathrm{M}_2(K)$ are both 4-dimensional vector spaces over K. Thus, to show that ψ is a bijection, it is enough to prove that it is injective. This is shown by noting that the homogeneous linear system corresponding to the equation $\psi(a_0 + a_1 i + a_2 j + a_3 k) = 0$ has determinant $-4(x^2 + y^2) = 4 \neq 0$ (as K is not of characteristic 2). Furthermore, ψ is K-linear and $\psi(\alpha\beta) = \psi(\alpha)\psi(\beta)$ for any $\alpha, \beta \in \mathbb{H}(K)$. □

Remark 4.27. [3, Exercise 2, p. 59]

a) Let $\alpha = a_0 + a_1 i + a_2 j + a_3 k \in \mathbb{H}(K)$ and $x, y \in K$ as in Proposition 4.26. A short computation shows that $\det(\psi(\alpha)) = a_0^2 - a_1^2(x^2+y^2) + a_2^2 - a_3^2(x^2+y^2) = a_0^2 + a_1^2 + a_2^2 + a_3^2 = N(\alpha)$.

b) Let $\alpha \in \mathbb{H}(K)$ be real, i.e. $a_1 = a_2 = a_3 = 0$, then it follows, by definition of ψ, that $\psi(\alpha) = a_0 \mathrm{Id}$. In other words, if α is real, its image under ψ is a scalar matrix.

We will need this isomorphism for $K = \mathbb{F}_q$ where q is any odd prime. Consequently, we need to check if all conditions of Proposition 4.26 are fulfilled.

Proposition 4.28. [3, Proposition 2.5.3] *Let q be any odd prime. Then, there exist $x, y \in \mathbb{F}_q$ with $x^2 + y^2 + 1 = 0$.*

A nonconstructive as well as a constructive proof can be found in [3, p. 59].
Recall that for any odd prime q, -1 is a square modulo q if and only if $q \equiv 1 \pmod 4$. In this case, we can set $y = 0$ and choose x such that $x^2 \equiv -1 \pmod q$. Then, $x^2 + y^2 + 1 \equiv 0$ and

$$\psi(a_0 + a_1 i + a_2 j + a_3 k) = \begin{pmatrix} a_0 + a_1 x & a_2 + a_3 x \\ -a_2 + a_3 x & a_0 - a_1 x \end{pmatrix}. \tag{35}$$

4.2.2 Construction of the Ramanujan Graphs $X_{p,q}$

Let us first give a short introduction to the groups $\mathrm{PGL}_2(q)$ and $\mathrm{PSL}_2(q)$ as they are the vertex sets of the Cayley graphs $X_{p,q}$.

Recall that $\mathrm{GL}_2(K)$ denotes the group of invertible 2-by-2 matrices over a field K and $\mathrm{SL}_2(K)$ its subgroup consisting of matrices having determinant 1. Then, $\mathrm{PGL}_2(K)$ and $\mathrm{PSL}_2(K)$ are the respective quotient groups defined by

$$\mathrm{PGL}_2(K) := \mathrm{GL}_2(K) \Big/ \left\{ \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \,\middle|\, \lambda \in K^\times \right\},$$
$$\mathrm{PSL}_2(K) := \mathrm{SL}_2(K) \Big/ \left\{ \begin{pmatrix} \mu & 0 \\ 0 & \mu \end{pmatrix} \,\middle|\, \mu \in \{\pm 1\} \right\}.$$

Later, we also need another description of these groups. For this, let $P^1(K) = K \cup \{\infty\}$ be the projective line over K. To a matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{GL}_2(K)$ we relate the fractional linear transformation

$$\varphi_A : P^1(K) \to P^1(K),$$
$$z \mapsto \frac{az+b}{cz+d}$$

where

$$\varphi_A(\infty) = \begin{cases} \frac{a}{c} & \text{if } c \neq 0 \\ \infty & \text{if } c = 0 \end{cases} \quad \text{and} \quad \varphi_A\left(\frac{-d}{c}\right) = \infty.$$

We now get a group homomorphism $\varphi : \mathrm{GL}_2(K) \to \mathrm{Sym}(P^1(K))$, $A \mapsto \varphi_A$ where $\mathrm{Sym}(P^1(K))$ denotes the group of permutations of $P^1(K)$. We see that

$$\ker(\varphi) = \left\{ \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \,\middle|\, \lambda \in K^\times \right\}$$

and

$$\ker\left(\varphi|_{\mathrm{SL}_2(K)}\right) = \left\{ \begin{pmatrix} \mu & 0 \\ 0 & \mu \end{pmatrix} \,\middle|\, \mu \in \{\pm 1\} \right\}.$$

Therefore, it follows, by definition of $\mathrm{PGL}_2(K)$ respectively of $\mathrm{PSL}_2(K)$ and by the first isomorphism theorem, that

$$\varphi(\mathrm{GL}_2(K)) \cong \mathrm{PGL}_2(K) \quad \text{respectively} \quad \varphi(\mathrm{SL}_2(K)) \cong \mathrm{PSL}_2(K).$$

For the construction we only need $K = \mathbb{F}_q$ for some prime q and we write $\mathrm{GL}_2(q)$ to indicate that we consider matrices over $\mathbb{F}_q$ (the other groups are denoted accordingly).

Now, we can start with the actual construction. Let $p \neq q$ be two odd primes. In the original construction by Lubotzky et al. [8] the primes p and q needed to be congruent to 1 (mod 4). However, we will here state the construction presented by Davidoff et al. [3, Section 4.2] which also works if one or both primes are equal to 3 (mod 4).

Let $\tau_q : \mathbb{H}(\mathbb{Z}) \to \mathbb{H}(\mathbb{F}_q)$, $\alpha \mapsto \alpha \pmod q$. Let $x, y \in \mathbb{Z}$ be such that $x^2 + y^2 + 1 = 0$ (cf. Proposition 4.28) and consider the corresponding isomorphism ψ_q (as defined in (34)). Note that in the case where $q \equiv 1 \pmod 4$ (and as described in [8]), ψ_q has the simpler form as in (35).
Recall the set of $p+1$ quaternions with norm p defined in (33) and take $\alpha \in S_p$. We note that $\det(\psi_q(\tau_q(\alpha))) = N(\alpha) = p \neq q$ (by Remark 4.27 a)) which implies that $\psi_q(\tau_q(\alpha)) \in \mathrm{GL}_2(q)$. This allows us to further apply the map φ and we define

$$S_{p,q} := (\varphi \circ \psi_q \circ \tau_q)(S_p)$$

which we regard as a subset of $\mathrm{PGL}_2(q)$. It holds, in addition, that $\psi_q(\tau_q(\alpha\bar{\alpha})) = \psi_q(\tau_q(\bar{\alpha}\alpha))$ is a scalar matrix in $\mathrm{GL}_2(q)$ (by Remark 4.27 b)). As $\ker(\varphi)$ consists exactly of the group of scalar matrices, it follows that $\varphi(\psi_q(\tau_q(\alpha)))^{-1} = \varphi(\psi_q(\tau_q(\bar{\alpha})))$. If $\beta \in S_p$ is such that $\bar{\beta} \notin S_p$, then β^2 is real which implies as before that $\tau_q(\psi_q(\beta))^{-1} = \tau_q(\psi_q(\beta))$. These two results show that $S_{p,q}$ is symmetric, i.e. $S_{p,q}^{-1} = S_{p,q}$.

Lemma 4.29. [3, Lemma 4.2.1] *Let $p \neq q$ be odd primes such that $q > 2\sqrt{p}$. Then, $|S_{p,q}| = p+1$.*

The proof can be found in [3, p. 113].
The group which we will take as vertex set for the graphs $X_{p,q}$ differs depending on whether p is a square modulo q or not. So, let us first introduce a symbol which describes this property.

Definition 4.30. [3, p. 48] Let q be an odd prime and $a \in \mathbb{Z}$. The *Legendre symbol* is defined as

$$\left(\frac{a}{q}\right) := \begin{cases} 0 & \text{if } q \text{ divides } a \\ 1 & \text{if } q \text{ does not divide } a \text{ and } a \text{ is a square modulo } q \\ -1 & \text{if } q \text{ does not divide } a \text{ and } a \text{ is not a square modulo } q. \end{cases}$$

Definition 4.31. [3, p. 113] Let $p \neq q$ be two odd primes. We define

$$X_{p,q} := \begin{cases} \mathcal{G}(\mathrm{PSL}_2(q), S_{p,q}) & \text{if } \left(\frac{p}{q}\right) = 1 \\ \mathcal{G}(\mathrm{PGL}_2(q), S_{p,q}) & \text{if } \left(\frac{p}{q}\right) = -1. \end{cases}$$

Eventually, we come to the main theorem of this section which describes lower bounds for the girth of $X_{p,q}$.

Theorem 4.32. [3, Theorem 4.2.2] *Let $p \neq q$ be odd primes with $p \geq 5$ and $q > p^8$. Then, $X_{p,q}$ are connected, $(p+1)$-regular graphs which are also Ramanujan. In addition,*

- *If $\left(\frac{p}{q}\right) = 1$, then $X_{p,q}$ is a nonbipartite graph with $\frac{q(q^2-1)}{2}$ vertices and $g(X_{p,q}) \geq 2\log_p(q)$.*
- *If $\left(\frac{p}{q}\right) = -1$, then $X_{p,q}$ is a bipartite graph with $q(q^2-1)$ vertices and the girth satisfies $g(X_{p,q}) \geq 4\log_p(q) - \log_p(4)$.*

It follows by Lemma 4.29 that $X_{p,q}$ are $(p+1)$-regular graphs. The number of vertices can be verified by computing the cardinality of $\mathrm{PSL}_2(q)$ respectively of $\mathrm{PGL}_2(q)$. Moreover, the fact that $X_{p,q}$ is bipartite (if p is not a square modulo q) follows by Proposition 2.28 (iv) where we use the homomorphism

$$\begin{aligned} \chi : \mathrm{PGL}_2(q) &\to \{\pm 1\}, \\ A &\mapsto \left(\frac{\det(A)}{q}\right). \end{aligned}$$

Indeed, we can show that $\chi(S_{p,q}) = \{-1\}$ (see Exercise 3 in [3, p. 73]).

The proof of the other statements of the theorem is fairly long and includes the construction of a different family of $(p+1)$-regular graphs $Y_{p,q}$. For that reason, we will only give a short outline of this construction (the details can be found in [3, Section 4.3]).
First, we define the set

$$\Lambda' := \{\alpha \in \mathbb{H}(\mathbb{Z}) \mid \alpha \equiv 1 \,(\mathrm{mod}\, 2) \text{ or } \alpha \equiv i+j+k \,(\mathrm{mod}\, 2),\ N(\alpha) = p^k \text{ for } k \in \mathbb{N}\}$$

which contains S_p. Then, we define the following equivalence relation on it.
Two elements $\alpha, \beta \in \Lambda'$ are equivalent, and we write $\alpha \sim \beta$, if there are two positive integers $n, m \in \mathbb{N}$ such that $p^m \alpha = \pm p^n \beta$. Denote $\Lambda := \Lambda'/\sim$ and let $Q : \Lambda' \to \Lambda$ be the map which maps an element $\alpha \in \Lambda'$ to its corresponding equivalence class $[\alpha] \in \Lambda$. It can then be shown that the Cayley graph $\mathcal{G}(\Lambda, Q(S_p))$ is the $(p+1)$-regular tree (see [3, Proposition 4.3.1]). In particular, it does not contain any cycles.

To define the graphs $Y_{p,q}$, we need a further quotient group of Λ (this will be the vertex set of $Y_{p,q}$) and the image of $Q(S_p)$ under a particular group homomorphism which we will not define explicitly here. The resulting Cayley graphs are isomorphic to the graphs $X_{p,q}$. This allows us to show the remaining properties of Theorem 4.32 by proving them for $Y_{p,q}$. However, we will not show these proofs here but they can be found in [3, Section 4.3] and the proof for the Ramanujan property in [8, Section 4].

Let us briefly recall the graph $\mathcal{G}(\mathrm{SL}_2(\mathbb{Z}), \{A, A^{-1}, B, B^{-1}\})$ from Chapter 4.1. We saw that this graph does not contain any cycles which means that it is the 4-regular tree. Hence, both, Margulis' construction and the graphs $Y_{p,q}$, are based on regular trees. Consequently, proving the lower bounds on the girth of $X_{p,q}$ respectively $Y_{p,q}$ can be done in a similar way as in Margulis' proof. Namely, by considering a cycle in the graph and transferring it back to the respective regular tree. However, even though these graphs are kind of similar, the bound on the girth of $X_{p,q}$ is much better than the one by Margulis.

Corollary 4.33. [3, Corollary 4.3.6] *Let $q > p^8$. Then, the graphs $X_{p,q}$ are $(p+1)$-regular connected graphs. Moreover,*

(i) If $\left(\frac{p}{q}\right) = 1$, then $X_{p,q}$ is nonbipartite and $g(X_{p,q}) \geq \frac{2}{3}\log_p(|X_{p,q}|)$.

(ii) If $\left(\frac{p}{q}\right) = -1$, then $X_{p,q}$ is bipartite and $g(X_{p,q}) \geq \frac{4}{3}\log_p(|X_{p,q}|) - \log_p(4)$.

Proof. The girth estimates follow directly from Theorem 4.32. Indeed, note that $|X_{p,q}|$ is equal to $\frac{q(q^2-1)}{2}$ respectively to $q(q^2-1)$ which in both cases implies that $\sqrt[3]{|X_{p,q}|} \leq q$. Plugging this into the lower bounds stated in Theorem 4.32 gives the desired result. The remaining properties are already stated in Theorem 4.32. □

Remark 4.34. As already mentioned in Chapter 2.3, Margulis also constructed a family of graphs with the same asymptotic lower bound. His construction is based on quaternions too but in addition also uses p-adic numbers. We will not explain this construction here; it can be found in [12].

5 Construction and Simulation of Margulis' LDPC Codes

5.1 Margulis' Construction of LDPC Codes

Recall the construction of graphs with large girth by Margulis, described in Section 4.1. In his paper [11], Margulis did not just describe these graphs, he also showed how a regular LDPC code can be constructed from such a graph. For this, let $G := \mathrm{SL}_2(\mathbb{F}_p)$ be the special linear group for some odd prime p and note that G consists of $p^3 - p$ elements. Furthermore, define the matrices $A, B \in G$ as in Section 4.1 by

$$A := \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

Recall that there does not exist any nontrivial multiplicative relation between A and B. This implies the following lemma.

Lemma 5.1. [11, Section 7] *There does not exist any nontrivial multiplicative relation between A^2, ABA^{-1} and B.*

Proof. We assume by contradiction that there exists a nontrivial multiplicative relation between A^2, ABA^{-1} and B. Therefore, there exists an integer $m \in \mathbb{N}$ and there exist matrices

$$M_1, \ldots, M_m \in \{A^2, A^{-2}, ABA^{-1}, AB^{-1}A^{-1}, B, B^{-1}\}$$

such that $M_1 \cdot \ldots \cdot M_m = \mathrm{Id}_2$. Let $n \in \mathbb{N}_{\geq m}$ and $N_1, \ldots, N_n \in \{A^{\pm 1}, B^{\pm 1}\}$ be such that M_1 is equal to N_1 respectively to N_1N_2 respectively to $N_1N_2N_3$ (depending on the definition of M_1) and so on. Hence, also $N_1N_2 \cdot \ldots \cdot N_n = \mathrm{Id}_2$. As there does not exist any nontrivial multiplicative relation between A and B, this can only be the case if n is even and if $N_i = N_{n-i}^{-1}$ for all $1 \leq i \leq n/2$. We consider the following two cases.

(i) It holds that $M_j^{\pm 1} \neq ABA^{-1}$ for all $1 \leq j \leq m$.
Consequently, $M_1, \ldots, M_m \in \{A^{\pm 2}, B^{\pm 1}\}$ which clearly implies that $M_1 \cdot \ldots \cdot M_m \neq \mathrm{Id}_2$ (as there does not exist any nontrivial multiplicative relation between A and B). This is a contradiction.

(ii) There exists an integer $j \in \{1, 2, \ldots, m\}$ such that $M_j^{\pm 1} = ABA^{-1}$.
Then, there exists another integer $i \in \{1, 2, \ldots, n\}$ such that $N_iN_{i+1}N_{i+2} = M_j$. Let i be such that $N_kN_{k+1}N_{k+2} \neq (ABA^{-1})^{\pm 1}$ for all $k \in \{i+1, i+2, \ldots, n/2\}$. We then have that $N_{i+3}, \ldots, N_{n-i-3} \in \{A^{\pm 2}, B^{\pm 1}\}$ and $N_{i+3}N_{i+4} \cdot \ldots \cdot N_{n-i-4}N_{n-i-3} = \mathrm{Id}_2$. We already know by case (i) that this is a contradiction.

Thus, we conclude that there does not exist any nontrivial multiplicative relation between A^2, ABA^{-1} and B. □

To define Margulis' LDPC code, we first need to construct a bipartite graph Y_p. The left vertices of this graph Y_p are the elements of G and the right vertices are the elements of two copies of G

which we call G_1 and G_2. Any right vertex $g_1 \in G_1$ is then connected to the left vertices

$$g_1A^2,\ g_1ABA^{-1},\ g_1B$$

and any right vertex $g_2 \in G_2$ is connected to the left vertices

$$g_2A^{-2},\ g_2AB^{-1}A^{-1},\ g_2B^{-1}.$$

Then, a parity-check matrix of an LDPC code will be the adjacency matrix of Y_p where the rows correspond to the elements of G and the columns to the elements of G_1 and G_2 (see Figure 9 for an illustration and Figure 10 for an example). Consequently, the resulting code is a $(3, 6)$-regular LDPC code of length $2 \cdot |G| = 2 \cdot (p^3 - p)$.

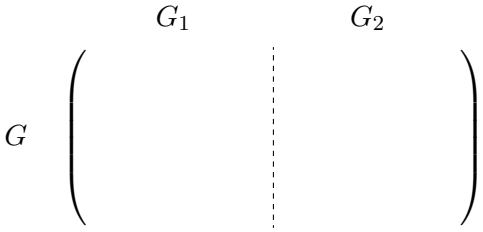

Figure 9: Sketch of the structure of the adjacency matrix of Y_p.

Figure 10 below shows an example of a parity-check matrix for the LDPC code for $p = 7$. The yellow dots represent ones and the blue dots represent zeros.

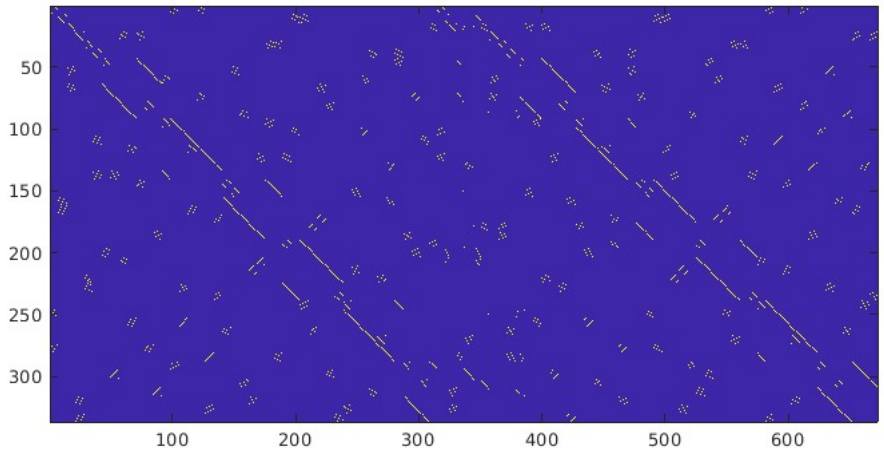

Figure 10: A parity-check matrix for the LDPC code for $p = 7$.

Let now $w = w_1w_2 \dots w_t$ for some $t \in \mathbb{N}_{>0}$ be a reduced word over $\{A^2, ABA^{-1}, B\}$. We can also see w as a reduced word over $\{A, B\}$; let us write $\tilde{w} = \tilde{w}_1 \dots \tilde{w}_t$ in this case. If $w_i^{\pm 1} \neq ABA^{-1}$ for all $1 \leq i \leq t$, $\tilde{w}$ is a word of length at most $2t$. If there exists $i \in \{1, \dots, t\}$ such that $w_i^{\pm 1} = ABA^{-1}$, then it can be verified that $\tilde{w}_i\tilde{w}_{i+1}$ (or $\tilde{w}_{i-1}\tilde{w}_i$ if $i = t$) is a word of length at

most 5. Continuing this argumentation, we can check that $\tilde{w}$ is a word of length at most $2t+1$. This leads to the following lemma.

Lemma 5.2. [14, p. 251] *We consider A and B as elements of $G = SL_2(\mathbb{F}_p)$. If the smallest nontrivial multiplicative relation between A and B has length t, then the smallest nontrivial multiplicative relation between A^2, ABA^{-1}, B has length at least $\frac{t}{2} - 1$. This implies that*

$$g(Y_p) \geq \log_{1+\sqrt{2}}(p/2) - 1$$

for all odd primes p.

We can improve this bound for the girth by using the improved bound from Corollary 4.10.

Corollary 5.3. *Let p be any odd prime. Then,*

$$g(Y_p) \geq \log_{1+\sqrt{2}}(p) - 1.$$

Proof. This follows directly from Lemma 5.2 and from Corollary 4.10. □

5.2 Simulation

Rosenthal and Vontobel already simulated the performance of the codes introduced in Section 5.1 for $p = 7$ and $p = 13$ (see [14, Chapter 2]). Later, MacKay and Postol simulated the performance for $p = 11$ (see [10, Section 2.1]). We wondered if there are (significant) differences between the performance of such codes for different prime numbers (and as a consequence thereof for different girths).

We decided to simulate the performance of five different codes, i.e. for $p \in \{7, 11, 17, 19, 23\}$. Let us call these codes C_p. First, we constructed the parity-check matrices of these codes, computed the corresponding generator matrices in systematic form and then simulated the transmission of random codewords for different signal-to-noise ratios (SNRs) through a BI-AWGN channel, using the SPA to decode the received messages. The decoded words were then compared with the original codewords and the number of wrong bits were counted. By dividing this number by the total number of sent bits, we obtain the bit error rate (BER). The so obtained plot of BERs for each SNR can be seen in Figure 13. A more detailed explanation of the simulation as well as parts of the used Sage respectively Matlab programme can be found in the appendix in Section 7.3.

Before discussing the results of the simulation, let us first consider the minimum distances of these codes. Analogous to [10], we found for all these codes the row of the corresponding generator matrices with the lowest Hamming weight to obtain an upper bound for the minimum distances. These results can be found in Table 1 below.

p	length of C_p	upper bound on $d(C_p)$
7	672	132
11	2640	220
17	9792	14
19	13680	1140
23	24288	5830

Table 1: Upper bounds on the minimum distances of C_p for $p \in \{7, 11, 17, 19, 23\}$.

Note that the upper bound for $d(C_{17})$ is notably smaller than the bounds for the other codes C_p and therefore gives a relatively good estimate for the actual minimum distance of C_{17}.
This difference between the upper bound for $d(C_{17})$ and the upper bounds for the minimum distances of the other codes C_p is also visible by looking at their generator matrices. Figure 11 below shows as an example a generator matrix of C_7. Generator matrices (in systematic form) for C_p with $p \in \{11, 19, 23\}$ all have a similar structure as this one, in the sense that the density of ones in the left half of every row of the matrix is high.

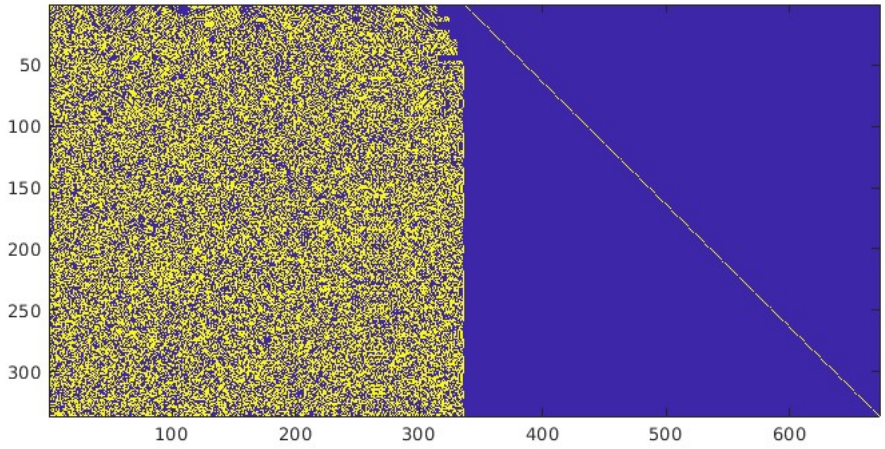

Figure 11: A generator matrix for the LDPC code for $p = 7$.

Figure 12 shows a generator matrix for C_{17}. It is also in systematic form even though the identity matrix on the right-hand side is not visible here (due to the large scale). We can see that the density of ones in the first few 1000 rows is much smaller than that of the generator matrix in Figure 11.

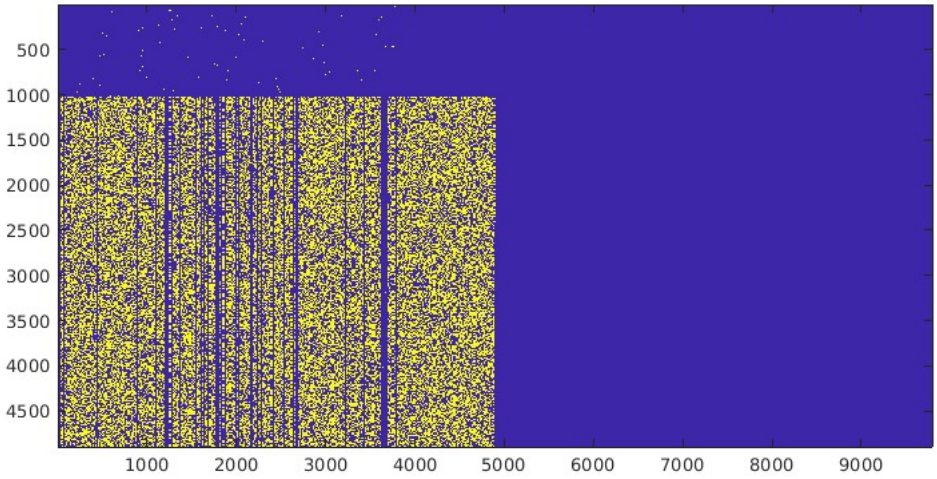

Figure 12: A generator matrix for the LDPC code for $p = 17$.

More precisely, the code C_{17} has at least 646 different codewords of Hamming weight 14 (as its generator matrix has that many rows of Hamming weight 14). As a consequence, C_{17} also has many near-codewords.

Definition 5.4. [10, Section 2.1] Let $C \subseteq \mathbb{F}^n$, where $\mathbb{F}$ is a finite field, be a code with parity-check matrix H. A (w, v) *near-codeword* is a vector $x \in \mathbb{F}^n$ of Hamming weight w and whose syndrome $Hx^\top$ has Hamming weight v.

We found several $(15, 3)$, $(13, 3)$ and $(16, 6)$ near-codewords of C_{17} by adding errors with small Hamming weight to the codewords with Hamming weight 14. MacKay and Postol [10] already found various $(12, 4)$ and $(14, 4)$ near-codewords for the code C_{11} which caused an error floor in the corresponding performance curve. An *error floor* describes a region in the curve where the slope of the curve becomes considerably smaller than before. When plotting the BER for each predefined SNR-value, an error floor corresponds to a decrease in the error-correcting performance.
Consequently, the found near-codewords as well as the large number of codewords of Hamming weight 14 of C_{17} indicate that the error-correcting performance of C_{17} might be worse than the performance of the other codes. This is indeed the case as the plot in Figure 13 shows.

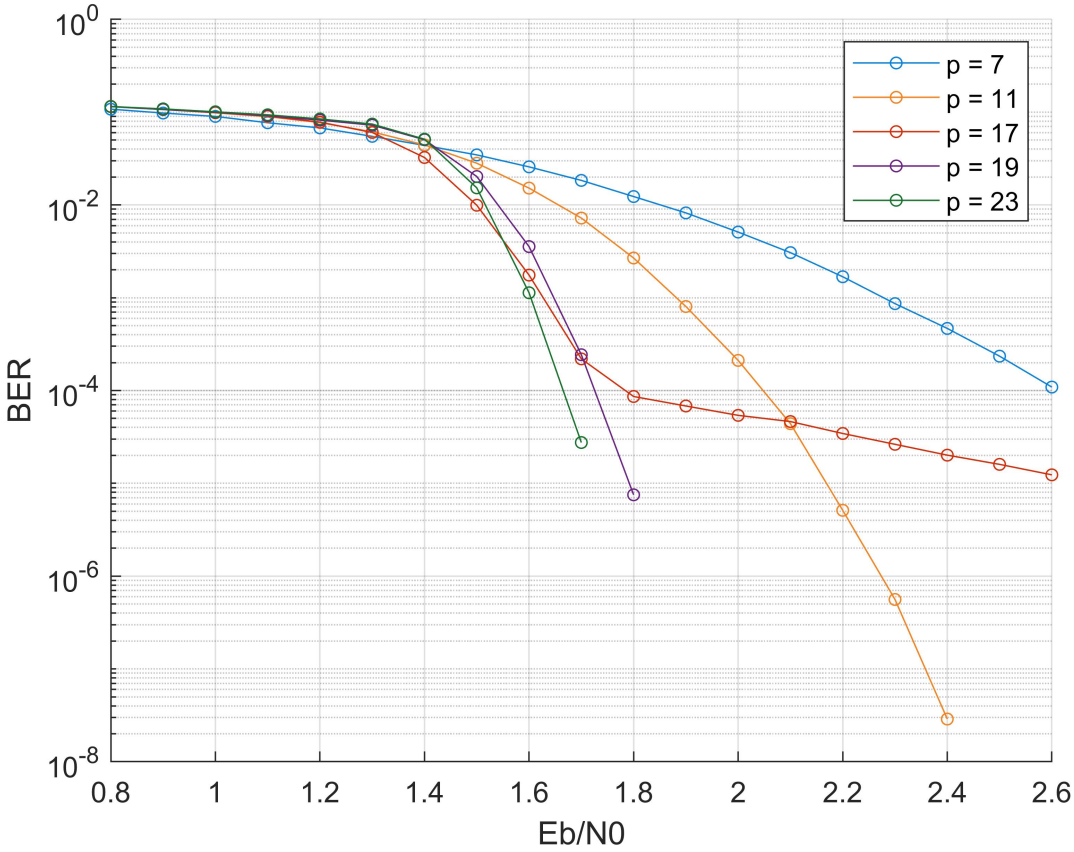

Figure 13: Error-correction performance of C_p, for $p \in \{7, 11, 17, 19, 23\}$, on the BI-AWGN channel. The SPA stopped after maximal 64 iterations and if for some SNR, respectively Eb/N0, no error had been found, then this result has not been plotted.

We see that the curve of C_{17} flattens quickly whereas all other curves still have a high slope. In other words, there is indeed an error floor in the performance curve of C_{17}. This is particularly remarkable because the corresponding graph X_{17} (as described in Section 4.1) has the same girth as X_{19} (see Section 7.2 in the appendix) and both codes are also of similar length. Hence, the fact that X_{17} has a larger girth than X_{11} or X_7 does not result in a better error-correction performance. This indeed reflects what we already predicted on the page before.
One possible explanation for this behaviour of C_{17} is that, due to the small minimum distance, the SPA Decoder makes many undetected errors. An *undetected error* is an error which is not detected by the decoder. More precisely, the decoder finds a valid codeword and returns it but this codeword is different from the originally sent one. Therefore, by modifying the used Matlab programme, we determined the number of undetected errors for C_7, C_{11} and C_{17}. As expected, the number of undetected errors for C_{17} is very large; namely 7255 undetected errors were found. Contrary to this, only 13, respectively 0, undetected errors were found for C_7, respectively C_{11}.

6 Conclusion

In this Master's thesis, we discussed Margulis' lower bound on the girth of a family of Cayley graphs $(X_p)_p$ and presented the results of simulations of the error-correction performance of some particular LDPC codes. First, the fundamental definitions of graph theory were introduced and we explained some important types of graphs, as for example graphs with large girths or expander graphs. Moreover, we gave a short overview of coding theory and in particular introduced LDPC codes and the SPA.

Afterwards, we looked at two constructions of Cayley graphs with large girth where we focused on the construction by Margulis. This construction is based on two matrices A and B and we saw that any two reduced words over $\{A, B\}$ define different elements of $\mathrm{SL}_2(\mathbb{Z})$. This fact combined with the operator norms of $A^{\pm 1}$ and $B^{\pm 1}$ led to the lower bound on the girth. We discussed some alternative choices for A and B and concluded that none of these lead to a better bound. By computing the exact girth of several graphs X_p, we detected that Margulis' lower bound is quite far away from the actual girth values (more precisely, the average difference is about 87%). We showed an improvement to the bound and presented further results which indicate that there might exist an even better bound. This is still an open problem as we only proved the part that if we assume that the matrices V and W meet certain conditions, then their norms are at least $2p$.

Then, we introduced a construction of LDPC codes, again by Margulis. We took a closer look at these LDPC codes for $p \in \{7, 11, 17, 19, 23\}$ by first computing upper bounds for the minimum distances of these codes. The results were surprising as the upper bound for $d(C_{17})$ was only equal to 14. Following this, we also found several $(15, 3), (13, 3)$ and $(16, 6)$ near-codewords of C_{17} and we observed that the SPA made several thousand undetected errors whereas it made almost no such errors for $p = 7$ and $p = 11$. Our simulations of the error-correction performance of these codes over a BI-AWGN channel confirmed that the performance of C_{17} is indeed much worse than the one of the other codes. It would certainly be interesting to simulate the error-correction performance also for higher primes to see if there are other codes whose performances are similar to the one of C_{17}.

7 Appendix

7.1 The Python Code for Calculating the Exact Girth

Let the matrices A and B be defined as in Chapter 4.1. Recall that we want to find the smallest integer $g \in \mathbb{N}_{>0}$ such that $M := M_1 M_2 \cdot \ldots \cdot M_g \equiv \mathrm{Id}_2 \pmod p$ but $M \neq \mathrm{Id}_2$ where $M_i \in \{A^{\pm 1}, B^{\pm 1}\}$. Without loss of generality, we assume that $M_1 = A^{\pm 1}$ (if not, we can consider M^{-1} or $M^\top$).
Then, for n greater or equal than Margulis' lower bound, all possible combinations of matrices in $\{A^{\pm 1}, B^{\pm 1}\}$ of length n are computed and it is checked if the resulting matrices are equivalent to the identity matrix (mod p). This is done by identifying these matrix products with lists. The general idea is the following.
We consider lists $L = [k_1, k_2, \ldots, k_n] \in \mathbb{Z}^n$ where k_1 is the exponent of A, k_2 the exponent of B, k_3 the exponent of A and so on. In other words, the list $L = [k_1, k_2, \ldots, k_n]$ corresponds to the matrix $A^{k_1} B^{k_2} A^{k_3} \cdot \ldots \cdot M_n^{k_n}$ where $M_n = A$ (respectively B) if n is odd (respectively even). We call a list L *valid* if the following two properties hold.

a) The sum over the absolute values of all its entries is equal to n, i.e. $\sum_{i=1}^{n} |k_i| = n$.

b) If there exists an $i \in \{1, \ldots, n\}$ with $k_i = 0$, then $k_j = 0$ for all $i \leq j \leq n$.

The function "matrices" checks if the matrix corresponding to the current list is equivalent to the identity and if not, it calls the function "next_list" which computes the next list.

Its basic principle is the following. If the current list is $L = [k_1, k_2, \ldots, k_n]$ and k_1 is different from n, then the next list is obtained by increasing k_1 by 1. If $k_1 = n$, then k_1 is set to $-n$ and 1 is added to k_2 instead. If also $k_2 = n$, then k_2 is set to $-n$ too and 1 is added to k_3 and so on. This is done as long as the list is not valid. Once we arrive at a valid list, this list is returned. As this would take too much time, many special cases are considered to improve the efficiency. Let us look as an example at the improvement in lines 53-58 of the Python code to understand the underlying principle.
This case is considered if we have a list of the form $L = [1, 1, \ldots, 1, a, b, \ldots]$ with $a > 0$, $b < 0$ and property a) is fulfilled (this is checked in line 46 of the Python code). Without applying any shortcut, the next list would be $L = [2, 1, \ldots, 1, a, b, \ldots]$. However, now property a) is not fulfilled any more, so this list is not valid. As all entries up to a are positive, we will not arrive at a valid list until b is changed. Indeed, all previous changes will either lead to a 0 in a wrong place (i.e. property b) does not hold) or they will increase sum([abs(x) for x in L]) (i.e. property a) does not hold). The first list where b is changed is $L = [-n, -n, \ldots, -n, -n, b+1, \ldots]$, which still does not fulfil property a). Recalling the list we started with, we might obtain a valid list for the first time if the entry before $b+1$ is equal to $-a-1$ and all the preceding entries are equal to -1, i.e. if $L = [-1, -1, \ldots, -1, -a-1, b+1, \ldots]$. Hence, all these steps can be skipped and we can directly go to this list. Note that this list might still be invalid because if $b = -1$, then we now have a 0 in our list (meaning property b) might not hold any more).

Remark 7.1. We always start with the list $L = [-1, -1, \ldots, -1]$ (see line 134 of the Python code) and end with the list end_list $= [-n, 0, 0, \ldots, 0]$ (see line 43 of the Python code). In this way, we find all valid lists by considering for each valid list L also the list $-L$.

The whole code can be found below.

```
1  import numpy as np
2  from numpy.linalg import matrix_power
3  import math
4  from datetime import datetime
5
6  # Chosen prime number (all calculations will be made (mod p))
7  p = 439
8  bound = 2*math.log(p/2,1+math.sqrt(2))-1
9  bound = math.ceil(bound)
10
11 # Defining the matrices
12 A = np.array([[1,2],[0,1]])
13 B = np.array([[1,0],[2,1]])
14 Ai = np.array([[1,-2],[0,1]])
15 Bi = np.array([[1,0],[-2,1]])
16 Id = np.identity(2)
17
18 # Open a file to store the results
19 f = open("results.txt",'a')
20 now = datetime.now()
21 dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
22 print("These calculations were made on:", dt_string, "and we use p =", p
       ,".")
23 f.write("These calculations were made on: ")
24 f.write(dt_string)
25 f.write(" and we use p = ")
26 f.write(str(p))
27 f.write(".\n")
28
29
30 def next_list(expo,n):
31     """Returns the next list (where expo is the current list) of length
          n which fulfils the following properties:
32        - The sum of the absolute values of its entries is equal to n
33        - If an entry is equal to 0, then all the following entries must
             also be equal to 0."""
34     c = 1
35     end_list = [0]*n
36     end_list[0] = -n
37
```

```
38      #checks if the required properties are fulfiled.
39      while sum([abs(x) for x in expo]) != n or c == 1 or allowed_zeroes(
            expo,n) == False:
40          c = 0
41
42          #In this case, there is no further list that could be returned.
43          if expo == end_list:
44              return False
45
46          if sum([abs(x) for x in expo]) == n:
47              L_1 = [x for x in expo if x > 0]
48              L_2 = [x for x in expo if x <= 0]
49              s = expo.index(L_2[0])
50
51              M = [x for x in expo if x != 1 and x > 0]
52
53              #If the list is of the form [1,1,...,1,a,b,...], with a > 0
                    and b<0, we can change it to [-1,-1,...,-1,-a-1,b+1,...]
54              if L_2[0] != 0 and L_1 != [] and expo[0:(s-1)] == [1]*(s-1):
55                  expo[0:(s-1)] = [-1] * (s-1)
56                  expo[s] += 1
57                  expo[s-1] = -expo[s-1]-1
58                  continue
59
60              #If the list is of the form [a_1,...,a_{k-1},0,a_{k+1},...,
                    a_n] with a_i != 0 for i<k and a_j != 0 for some k+1<j<n,
                    then we can change it to [-1,-1,...,â_{k-1},1,a_{k+1},
                    ...,a_n] where â_{k-1}<=0 is s.t. sum(abs([...])) = n.
61              if allowed_zeroes(expo,n) == False and expo.index(0) > 0:
62                  index_0 = expo.index(0)
63                  expo[0:(index_0-1)] = [-1]*(index_0-1)
64                  expo[index_0] += 1
65                  expo_abs = [abs(x) for x in expo]
66                  value = sum(expo_abs) - expo_abs[index_0-1]
67                  expo[index_0-1] = value - n
68                  continue
69
70              #If the list is of the form [1,1,...,1,a,b,...], with a > 1,
                    we can change it to [-1,-1,...,-1,â,b+1,...] where â is
                    s.t. sum(abs([...])) = n.
71              if M != []:
72                  ind = expo.index(M[0])
73                  if ind > 0 and expo[0:ind] == [1]*ind:
74                      expo[0:ind] = [-1] * ind
75                      expo[ind+1] += 1
```

```
76                      expo_abs = [abs(x) for x in expo]
77                      value = sum(expo_abs) - expo_abs[ind]
78                      expo[ind] = value - n
79                      continue
80
81          for j in range(n):
82              #if the j-th entry is n, we set it to -n and continue with
                    the next entry
83              if expo[j] == n:
84                  expo[j] = -n
85                  continue
86
87              if allowed_zeroes == True:
88                  zero_ind = n
89                  #if "allowed" zeroes are in expo, we want to find the
                        index of the first one, i.e. the number of non-zero
                        entries
90                  if 0 in expo:
91                      zero_ind = expo.index(0)
92
93                  #The highest possible value an entry of the list can
                        attain is n minus the number of non-zero entries
                        except expo[j] (call it val).
94                  #If expo[j] is lower than -val, we can directly set it
                        to -val
95                  if expo[j] < -(n-zero_ind)-1:
96                      expo[j] = -(n-zero_ind)-1
97                      break
98
99                  #If expo[j] is greater or equal than val, we can
                        directly set it to -val and continue with the next
                        entry
100                 elif expo[j] >= n-zero_ind+1 and j != n-1 and expo[j+1]
                        != -1:
101                     expo[j] = -(n-zero_ind)-1
102                     continue
103
104                 #If no of the above cases holds, we just add 1 to it.
105                 else:
106                     expo[j] += 1
107                     break
108
109             #If allowed_zeroes == False and expo[j] != n, we just add 1
                    to the value.
110             else:
```

```
111                 expo[j] += 1
112                 break
113     return expo
114 
115 def allowed_zeroes(expo,n):
116     """Checks if a given list of length n is allowed, i.e. if there is
        no 0 in a "wrong" place"""
117     if 0 not in expo:
118         return True
119     else:
120         j = expo.index(0)
121 
122     for k in range(j+1,n):
123         if expo[k] != 0:
124             return False
125     if expo == [0]*n:
126         return False
127     else:
128         return True
129 
130 
131 def matrices(a,b,ai,bi,n):
132     """Checks if there is a nontrivial relation between the matrices a
        and b"""
133     #a,b: matrices (and ai,bi their inverses resp.) and n is the sum of
        the absolute values of the exponents
134 
135     #We always start with this list.
136     L = [-1]*n
137 
138     #The counter is equal to 0 iff -L has not been considered yet.
139     counter = 0
140     k = 0
141 
142     #L = False iff we arrived at the last possible list.
143     while L != False:
144         k += 1
145         C = np.identity(2)
146 
147         #Every entry of the list L is seen as exponent of a resp. b and
            the result is multiplied by C and reduced mod p.
148         for index in range(0,n,2):
149             if L[index] > 0:
150                 C = C.dot(matrix_power(a,L[index])) %p
151             else:
```

```
152                 C = C.dot(matrix_power(ai,abs(L[index]))) %p
153
154             if index+1 <= n-1 and L[index+1] > 0:
155                 C = C.dot(matrix_power(b,L[index+1])) %p
156             elif index+1 <= n-1 and L[index+1] <= 0:
157                 C = C.dot(matrix_power(bi,abs(L[index+1]))) %p
158
159         #If the obtained matrix C is equal to the identity, we've found
               a nontrivial relation and the list of exponents L is returned
160         if np.array_equal(C,Id):
161             print("The girth is equal to", n,"and the nontrivial
                   relation is based on L =",L,".")
162             return L
163         elif counter == 0: #we do the same calculations again for -L
164             L = [-x for x in L]
165             counter = 1
166         else: #we come here if C != Id and we've computed C for L and -L
167             L = [-x for x in L] #-L is transferred back to L
168             counter = 0
169             L = next_list(L,n) #we go over to the next list
170
171     print("No nontrivial relation of length n =",n,"has been found where
           k =",k,"possibilities were checked.")
172     return None
173
174 # We start by a list of length n >= max(bound,2) and stop as soon as a
      relation has been found
175 L = None
176 n = max(bound,2)
177 while L == None:
178     L = matrices(A,B,Ai,Bi,n)
179     n += 1
180
181 # We store the results in the file.
182 f.write("The girth is equal to ")
183 f.write(str(n-1))
184 f.write(" and the nontrivial relation is based on L = ")
185 f.write(str(L))
186 f.write(".\n")
187 f.write("The difference between the lower bound ")
188 f.write(str(bound))
189 f.write(" and the girth thus is ")
190 f.write(str(n-1-bound))
191 f.write(".\n")
192 f.write("\n")
```

```
193 f.close()
194
195 print("The difference between the lower bound",bound,"and the girth",n
    -1,"is:",n-1-bound,".")
```

7.2 Results Girth Calculation

We provide here the results obtained by the Python code presented in Section 7.1.

p	Lower bound	Girth
3	0	3
5	2	5
7	2	6
11	3	9
13	4	10
17	4	10
19	5	10
23	5	12
29	6	10
31	6	14
37	6	14
41	6	10
43	6	14
47	7	14
53	7	14
59	7	14
61	7	16
67	7	14
71	8	14
73	8	14
79	8	14
83	8	14
89	8	14
97	8	14
101	8	14
103	8	18
107	9	17
109	9	15
113	9	18
127	9	18

p	Lower bound	Girth
131	9	18
137	9	18
139	9	18
149	9	17
151	9	18
157	9	18
163	9	18
167	10	18
173	10	18
179	10	18
181	10	18
191	10	18
193	10	18
197	10	18
199	10	18
211	10	18
223	10	18
227	10	18
229	10	18
233	10	18
239	10	14
241	10	18
251	10	18
257	11	18
263	11	20
269	11	18
271	11	18
277	11	18
281	11	20
283	11	18

p	Lower bound	Girth
293	11	18
307	11	18
311	11	18
313	11	18
317	11	20
331	11	20
337	11	18
347	11	20
349	11	18
353	11	20
359	11	18
367	11	18
373	11	18
379	11	18
383	11	22
389	11	20
397	12	22
401	12	20
409	12	18
419	12	20
421	12	20
431	12	21
433	12	20
439	12	22
443	12	20
449	12	20
457	12	21
461	12	22
463	12	18

7.3 Matlab and Sage Codes used for the Simulation of LDPC Codes

7.3.1 Computing a Parity-Check Matrix

The following Sage code computes a parity-check matrix H for the LDPC code as described in Chapter 5 for any chosen prime p.

```
# Choose a prime number and define IF_p
p = 23
G = SL(2,GF(p))

A = matrix(GF(p),2,[1,2, 0,1])
B = matrix(GF(p),2,[1,0, 2,1])

# Create the Cayley graph CG_1 = G(G,S_1) and compute its adjacency
   matrix H_1
S_1 = [A^2, A*B*A.inverse(),B]
CG_1 = G.cayley_graph(generators = S_1)
H_1 = CG_1.adjacency_matrix()

# As above but with a different set (S_2)
S_2 = [(A^2).inverse(), A*B.inverse()*A.inverse(),B.inverse()]
CG_2 = G.cayley_graph(generators = S_2)
H_2 = CG_2.adjacency_matrix()

# Define the parity-check matrix and save it
H = block_matrix([[H_1,H_2]])
import csv
with open('H_matrix_19.csv','w') as f:
    c = csv.writer(f)
    c.writerows(H)
```

7.3.2 Simulation of LDPC Codes

A large part of the code for this simulation was taken from [1]. However, this code is intended for a different type of LDPC codes (i.e. for Quasi Cyclic LDPC codes) so we adapted it. Thus, we will provide below only this adapted respectively new part of the code. The whole code consists of four Matlab files which contain the following steps.

1. Load the parity-check matrix H, which was computed by the Sage code presented in Section 7.3.1, and compute a corresponding generator matrix G in systematic form.

2. For each pre-defined SNR value a random message is created, encoded and Gaussian noise is added.

3. For each pre-defined SNR value, every received codeword is decoded using SPA and the respective bit error rates are computed.

4. The loops stop if a predefined number of errors or iterations is reached.

For the first step, H is first brought into row-reduced echelon form. Such a function already exists in Matlab (it is called "rref"), but not over $\mathbb{F}_2$. We therefore wrote a new one (called "rref_bin") by basically just changing all operations of the function rref into operations in $\mathbb{F}_2$. To compute a generator matrix G, we wrote a new function "Gen_matrix" which is based on the function "coding_matrix_systematic" (taken from [7]). The function "Gen_matrix" can be found below.

```
function [H_new,G] = Gen_matrix(H)
% Computes the generator matrix G of a code given by its parity-check
% matrix H.
%   H: Parity-check matrix
%   H_new: Parity-check matrix with switched columns (s.t. G is in
%   systematic form)
%   G: Generator matrix in systematic form

[~,n] = size(H);
P = eye(n);

% Computes the binary row-reduced echelon form and the rank (k) of H
[H_RR,k] = rref_bin(H);

% Then bring it into systematic form [Id, A]
while(1)
    % Find the index of the first column i which is not equal to e_i
    col = find(diag(H_RR(1:k,1:k))~= 1, 1);

    % If no such column exists, break the loop
    if isempty(col)
        break
    end

    % Find the next column which is equal to e_i
    col_corr = find(H_RR(col,:),1);

    % swap the two columns and "save" the changes for later in P
    H_RR(:,[col col_corr]) = H_RR(:,[col_corr col]);
    P(:,[col col_corr]) = P(:,[col_corr col]);
end

% Compute G and adapt H such that G*H^T = 0
G = gen2par(H_RR);
H_new = mod(H*P,2);
end
```

For the other steps, we used the files "LDPC_Main_Program.m" and "SumProduct.m" which both can be found in [1]. We only made some small adaptions to the first programme (especially calling the function "Gen_matrix" and deleting all parts that were about another decoding algorithm, the Min Sum Algorithm).

References

[1] Ganeshaanand (Rishi) Balasubramanian. Low Density Parity Check (LDPC) Codes MATLAB-Simulation. URL: https://github.com/rbga/Low_Density_Parity_Check_LDPC_Codes_-_MATLAB_Simulation/tree/main.

[2] J. Bourgain and A. Gamburd. Uniform expansion bounds for Cayley graphs of SL2(double struck F p). *Annals of Mathematics*, 167(2), 2008. doi:10.4007/annals.2008.167.625.

[3] G. Davidoff, P. Sarnak, and A. Valette. *Elementary Number Theory, Group Theory, and Ramanujan Graphs*. Cambridge University Press, Cambridge, 2003.

[4] P. Erdős and H. Sachs. Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 12:251–257, 1963.

[5] R. Fricke and F. Klein. *Vorlesungen über die Theorie der elliptischen Modulfunctionen*. Teubner, Leipzig, 1890.

[6] R. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, 8(1):21–28, 1 1962. doi:10.1109/TIT.1962.1057683.

[7] Hicham Janati. Simulation of LDPC Codes & Applications. URL: https://github.com/hichamjanati/pyldpc/tree/master.

[8] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. doi:10.1007/BF02126799.

[9] R. C. Lyndon and J. L. Ullman. Pairs of real 2-by-2 matrices that generate free products. *Michigan Mathematical Journal*, 15(2):161–166, 6 1968. doi:10.1307/mmj/1028999969.

[10] D. J.C. MacKay and M. S. Postol. Weaknesses of Margulis and Ramanujan-Margulis Low-Density Parity-Check Codes. *Electronic Notes in Theoretical Computer Science*, 74:97–104, 10 2003. doi:10.1016/S1571-0661(04)80768-0.

[11] G. A. Margulis. Explicit constructions of graphs without short cycles and low density codes. *Combinatorica*, 2(1):71–78, 1982. doi:10.1007/BF02579283.

[12] G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of information transmission*, 24(1):39–46, 1988.

[13] R. J. Wilson. *Introduction to Graph Theory*. Addison Wesley Longman Limited, 4 edition, 1996.

[14] J. Rosenthal and P. O. Vontobel. Constructions of LDPC Codes using Ramanujan Graphs and Ideas from Margulis. *Proceedings of 38th Allerton Conference on Communication, Control and Computing*, pages 248–257, 10 2000.

[15] R. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006. doi:10.1017/CBO9780511808968.

[16] W. Ryan. An Introduction to LDPC Codes. In *Coding and Signal Processing for Magnetic Recording Systems*, chapter 36. CRC Press, 11 2004. doi:10.1201/9780203490310.ch36.

[17] W. E. Ryan and S. Lin. *Channel codes: Classical and modern*. Cambridge University Press, 2009. doi:10.1017/CBO9780511803253.

[18] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 9 1981. doi:10.1109/TIT.1981.1056404.

[19] J. H. van Lint. *Introduction to Coding Theory*, volume 86. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. doi:10.1007/978-3-642-58575-3.

[20] P. O. Vontobel. *Algebraic Coding for Iterative Decoding*. PhD thesis, Swiss Federal Institute of Technology Zurich, Zurich, 2003.