This project was, given a prime $p$ and exponent $n$, to find all $d$-generator $p$-groups of order $p^n$ whose derived subgroup has size $p$ for $p > 2$ based on Blackburn’s classification [1].

PgrpdGendp is $d$-generator $p$-groups with derived subgroup of order $p$. This new object was coded in the Pgrp2Gen package of GAP originally written by Kelly Brunemann and Dr. Robert Morse in 2011. Additionally, the package was cleaned up, refactored, and updated.
ACKNOWLEDGMENTS

This project was supported by Dr. Morse for the technical contents and by Dr. Hwang for the early phase of the project. I would also like to show my gratitude to the University of Evansville and the Computer Science program for providing me enough knowledge to pursue the project. Without any of these, this project would not be as successful as it is.

FIGURES

Figure 1: Structure of GAP
Figure 2: Functional Decomposition of Groups
Figure 3: Hierarchy of Types
INTRODUCTION and BACKGROUND

1. Introduction

Groups are applied to everyday life from basic algebra and solutions for Rubik’s Cube to quantum computing and error correcting codes. In order to stimulate further research in the field, this project especially focuses on extending a package called Pgrp2Gen for the GAP system written by Kelly Brunemann and Dr. Robert F. Morse in 2011 to include a new classification of $p$-groups by Blackburn [1]. The original package covers the $p$-groups classified by Ahmad, Magidin, and Morse [2]. This project extends the original package to include $d$-generator $p$-groups with derived subgroup of order $p$ classified by Blackburn [1]. Also, the package is cleaned up, refactored, and updated by transforming documentation formats.

GAP – Groups, Algorithms, and Programming – is an open system for computational discrete algebra, with a particular emphasis on Computational Group Theory. The first version was developed in 1986 at Lehrstuhl D für Mathematik, Germany. It is used for research as well as teaching the theory of groups including representations of groups [3]. The source code of the system is free to everyone, and there exists hundreds of developers who build packages on the system based on their own research interests. Packages are used to extend GAP. Many of the packages are available in the website. Currently, thousands of users take advantage of this system. The users vary from mathematicians to chemists and physicists. GAP can execute many different calculations. However, the system is especially helpful to users to find counterexamples and to support conjectures about groups that have not been proven yet. The current version of GAP provides all groups up to size two thousand in a library that can be searched, and groups of size or order $p^6$, $p^7$, and $3^8$. There exists groups that are not covered in GAP and our package adds more $p$-groups.
Dr. Robert Morse at the University of Evansville focuses on classifying groups of prime power order or $p$-groups. This is very crucial because $p$-groups are the bases of building other groups, just like any number can be represented by product of prime numbers. If you would like to understand numbers, it is essential to understand prime numbers. In order to include a new classification of $p$-groups in GAP, Dr. Morse desires to extend the system. This project is preceded by a project by Kelly Brunemann, who graduated from the University of Evansville in 2011, collaborating with Dr. Morse to include 2-generator $p$-groups of class 2 in GAP by making a new package called Pgrp2Gen. Now, the current project is to extend the package developed by Brunemann to enumerate a larger class of $p$-groups based on Blackburn’s classification.

The secondary requirements are to document previous version of package in GAPDoc, do general cleanup, and refactor. Therefore, it needed to be documented in the modern format called GAPDoc. Details are given in the next section.

2. Mathematical Definitions

A group is a set equipped with an associative binary operator in which there is an identity element and an inverse. In symbols,

$$\cdot : G \times G \rightarrow G$$

such that

$$\forall a, b, c \in G, (a \cdot b) \cdot c = a \cdot (b \cdot c),$$

$$\exists e \in G, \forall a \in G, e \cdot a = a \cdot e = a,$$

$$\forall a \in G, \exists b \in G, a \cdot b = b \cdot a = e$$

For example, integers with the binary operator addition compose a group because they satisfy all of a group’s properties. There are associative because addition of any order of integers provides same result. These have an identity element of 0. The negative number of an integer is its additive inverse.
A subgroup is a subset of a group that is itself a group with the same group operation.

The derived subgroup is defined as

$$ G' = < a^{-1}b^{-1}ab | a, b \in G > $$

The derived subgroup measures how commutative a group is. If $a$ and $b$ commute then

$$ a^{-1}b^{-1}ab = a^{-1}b^{-1}ba = a^{-1}a = e $$

So for commutative groups $G'$ is trivial (contains only the identity).

Let $p$ be a prime. A finite $p$-group is a group whose size or order is some prime power.

An example of a $p$-group is the group of integers mod $p$ where the operation is addition modulo $p$, consisting of the numbers from 0 to $p-1$.

This project entails classifying $p$-groups. By classification we mean identifying $p$-groups up to isomorphism. An isomorphism $i$ is a bijective homomorphism of groups. In notation,

$$ i : G_1 \rightarrow G_2 $$

such that

$$ i(ab) = i(a) i(b) \text{ (operation preserving)} $$

An $i$ is a bijection (onto and 1-1).

For example in geometry, similar triangles have same measure in angles but different measure in side lengths. So the trigonometric functions are the same for similar triangles. Hence, we can think of similar triangles being the same in some sense. Similarly, for every $n$ there are fixed finite number of non-isomorphic groups of order $p^n$.

**PROBLEM STATEMENT**

Given a prime $p$ and exponent $n$, the problem is to find all $d$-generator $p$-groups of order $p^n$ whose derived subgroup has size $p$ for $p > 2$ in GAP. The extended package will help Dr. Morse as well as other users who are interested in $p$-groups to identify more properties of $p$-
groups, not limited to the smaller collections of groups. Also, this project will help further research in the area.

**REQUIREMENTS and SPECIFICATIONS**

In order to solve the problem mentioned in the earlier section, it was required to classify up to isomorphism all $d$-generator $p$-groups whose derived subgroup has size $p$ for $p > 2$ based on the Blackburn’s classification [1]. Following the above, adding PgrpdGendp to the Pgrp2Gen package, cleaning up and refactoring code, and updating the entire package was specified to be completed in this project. More specifically, closely related to Pgrp2Gen, this project included constructors for PgrpdGendp and its recognition function, cleaned up and refactored to organize the entire package with regard to source code, and updated the documentation format from Tex to GAPDoc.

**DESIGN APPROACH**

![Figure 1: Structure of GAP](image)
1. System Overview

The main system used in the project is called GAP – Groups, Algorithms, and Programming. The GAP system is composed of the following structural hierarchy shown in Figure 1. GAP consists of a kernel, a documentation for the GAP system itself, packages, a library, and data files. The kernel written in C is where interpreters are at for GAP language. It has about 200,000 source lines of code. The library is used for all functionalities for GAP. It has 1.6 million source lines of code. The GAP system can be extended with packages by any programmers. These packages are all specialties for the system. Some GAP packages are automatically loaded when you start running GAP. Some other packages may be loaded by the “LoadPackage” command. Some packages may require a specific operating system depending on the external programs. Each GAP package contains GAP code and documentation and may contain data files and external programs so that GAP code can provide an interface [4]. The GAP code consists of GAP declaration and implementation files.

2. Third-Party Subsystem

This project requires to use an open-source system called GAP. The GAP system is open source because there will be no charge to use and the user is free to pass it on within certain limits. The further copyright and license can be found at [5]. This system is extensible, which means that the user can write the user’s own package in GAP language to extend the form of the system as known as the “library” [4]. The basic tutorial is available on the GAP system website at [6]. The GAP system can be executed in GAP language, which is a script language interpreted by the kernel. The GAP library itself is also written in GAP language. Therefore, it is easier for users to extend the system if the users wish to do so. In order to download the GAP system, the user can go to their website at [3]. The user can find the latest version of the system under “Downloads” and download one based on the user’s operating system. In this project, the GAP
version 4.8.8 was used. The system name is gap4r8. The Pgrp2Gen package used in this project was provided by Dr. Morse at the University of Evansville. To use the package, the user needs to put it under pkg folder in gap4r8. To run the Pgrp2Gen package, the user can execute the

\[
\text{LoadPackage(“pgrp2gen”)};
\]

command.

---

**Figure 2: Functional Decomposition of Groups**

3. **Functional Decomposition**

The functional decomposition of the project is shown in Figure 2. There were two steps involved to create an object whose type is Pgrp2Gen or PgrpdGendp. First, two major types of representations of groups were involved. One is called the finite presentation and the other is called the pc-group representation. All finite \( p \)-groups are solvable and have pc-group representation. The pc-group representation allows for very fast computing and PgrpdGendp can be represented this way. Therefore, it was necessary to transform the finite presentation with parameterizations to the pc-group representation as the first step to achieve a fast computation for large numbers. This transformation was done by Nilpotent Quotient Algorithm. The second step was to create a new PgrpdGendp type by using the pc-group representation via inheritance from PcGroup. The Pgrp2Gen type was similarly created from PcGroup. This creation was done
by setting compute attributes. This data hierarchy allows the pc-group type to execute functions that are relative to the PgrpdGendp and the Pgrp2Gen types. All mappings and attributes of PgrpdGendp are kept being relative to the mathematical object.

![Diagram of the hierarchy of types](image)

**Figure 3: Hierarchy of Types**

4. **Implementation**

The hierarchy of the types involved in this project is shown in Figure 3. PgrpdGendp inherits the pc-group type. PgrpdGendp will have an invariant set or signature as its attributes parameterized by \((\rho, e, A)\) as seen in Blackburn’s classification. Generators and relations change based on parameterization of \((\rho, e, A)\). Pgrp2Gen from the original version of the Pgrp2Gen package has two constructors: AllPgroups2gen \((n, p)\) and Pgrp2gen \((\alpha, \beta, \gamma, \rho, \sigma, p)\). In the new version of the package, the new type called PgrpdGendp also has two constructors:

- AllPgrpsdgendp \((n, p)\), and Pgrpdgendp \(((\rho, e, A), p)\) where \(p\) is a prime number and \(n\) is a power of prime.
- AllPgrpsdgendp \((n, p)\) generates all PgrpdGendp’s for a given \(p^n\).
- Pgrpdgendp \(((\rho, e, A), p)\) creates an object of PgrpdGendp. An additional constructor called AllSignature \((n)\) is also added to the package. AllSignature \((n)\) creates all possible combinations of signatures in the pc-group representation of a given order for all \(p\).

The function IsPgrp2gen \((G)\) takes a pc-group and returns a signature of Pgrp2Gen if the group is Pgrp2Gen. The other function IsPgrpdgendp \((G)\) also takes a pc-group and returns a signature of PgrpdGendp if the group is PgrpdGendp. While implementing these, extra code was removed. Also, the code is entirely formatted and comments
were added. Documentations in the Pgrp2Gen package was written from TeX to GAPDoc. GAPDoc was written in GAP language. Besides the contents that were already in the package, new contents regarding PgrpdGendp were added as well. This GAPDoc allows to produce multiple formats when it is compiled. PgrpdGendp was tested by comparing an expected value and the return value from the new constructors and methods.

5. Design Evaluation

Pc-group representation allows for the fastest computation for $p$-groups. Therefore, the program can easily work with very large finite $p$-groups. As a result, $5^{109}$ is considered a small group. However, the transformation from finite generated to pc-group was difficult because of powers in relations. Alternately, all groups could be left as a finite presentation whose methods rely on Knuth-Bendix rewriting, which is exceedingly slow.

RESULTS

This project completed many different aspects. After classifying up to isomorphism all $d$-generator $p$-groups whose derived subgroup has size $p$ for $p > 2$ based on Blackburn’s theory [1], PgrpdGendp was added to the Pgrp2Gen package. Its constructors and recognition methods follow Pgrp2Gen constructors and methods and was tested as well to make sure the object’s validity. The package was also cleaned up and refactored for publication by removing extra code, formatting, and adding comments for the entire code to improve readability as a whole. Additionally, the package was successfully updated by writing documents from TeX to GAPDoc, which allows to generate multiple formats such as PDF and HTML. Finally, the group constructors and methods are tested.
CONCLUSION

1. Significance

This project met the requirements and specifications by generating PgrpdGendp of order $p^n$ of such group, improving readability of both packages, and modernizing documentations for the packages. Hence, the project solved the problem defined. This project will allow users including Dr. Morse to pursue his research on the topic by providing groups of such a type. Pc-group representation allow for the fastest computation, which helps especially when dealing with very large size of groups.

2. Future Work

The future work will be to classify the Miech groups 2 generator $p$-group with cyclic derived subgroup [7]. This type of groups is closely related to PgrpdGendp. The derived subgroup of order $p$ implies the subgroup is cyclic.

3. Lessons Learned

There are three lessons learned in the project. First, regarding technical aspect, building a new code on an existing system was not as simple as expected because it required to understand the overall structure within the system such as folders, files, and coding custom. Especially when using a new programming language, a program tended to get bugs easily. Therefore, building code little by little was somewhat effective to avoid unknown errors. Limited resources made the project harder because understanding many from only a couple references to apply these to the project. Updating a system was not as easy as expected either because it was sometimes troublesome to find a corresponding command to a previous version.

The second lesson is rescheduling regularly. Having a fixed schedule before actually starting a project is very helpful. However, following the timeline is not always easy especially when working with other people because some might not be able to accomplish their tasks by the
time specified. This brings a delay to the project. Hence, rescheduling regularly allows engineers to understand and follow the timeline well.

The third and last lesson is about work ethics. When working as a group, showing a respect to other members is important to smoothly proceed a project. Everyone has different life and schedule, and sometimes some people cannot make it on time. Unless it is problematic, that is totally fine. In this case, usually these people find their time to catch up. However, trying to finish an assigned task earlier than expected is always favorable to easily adjust to the progress.

REFERENCES
BIOGRAPHY

Yuka Murata is from Tsukuba, Ibaraki, Japan. She likes to know things from many different topics. She is going to attend a master program in Computer Science from the coming fall semester. She has not decided which university to go by this time. However, it will be fixed shortly. She will mainly study integrated media design, UI, and UX in order to create new technology that is easy for anyone to use and is good-looking.