On the Application of a Generalization of Artin's Primitive Root Conjecture in the Theory of Monoid Rings

Spencer Chapman

May 22, 2022

Abstract

The following paper covers both the theoretical and programming aspects of my Junior Tech Workshop, aimed to aid Dr. Daileda in his number theory research. The paper will briefly mention the content of my preliminary presentation, and focus more on the programming aspect. In addition, some general extensions of the work done will be mentioned at the end. The maple file created during research is attached in the zipped folder accompanying this paper, as well as a txt version.

1 Motivation

My original project I had planned was far from the one I provided; the opportunity to work on this sprouted from a brief conversation with Dr. Daileda after class, as we had worked together previously on my writing workshop project last semester. My previous experience with him certainly awakened a deep interest in number theory that I had forgotton from highschool maths. As such, there is no better motivation I had for this project other than my personal interest, as I believe to be the typical attractiveness for pure math. However, I argue that my intellectual satisfaction is not the only fruit of labor, as polynomials are quite useful in the world of cybersecurity. The knowledge of factorization within polynomial rings is of course guaranteed, but the actual computation of such elements is far from trivial. I targeted my presentation towards this fact, by demonstrating how easily a simple factorization we've used can be lost with some of the loosest restrictions on exponents. In addition, factorization in general is a backbone to cybersecurity, with modern encryption schemes relying on the fact that factorization in the integers is just objectively difficult. However, factorization of polynomials is provably more traceable than the integers, namely to the abundance of finite fields that may be obtained by quotients of the integers. The message to be portrayed here is that the results of this project are not just interesting but useful in their own rite. Without delving too far off topic, I will now open the technology aspect of this project.

2 Technology Aspect

2.1 Setting up the problem

During my preliminary presentation, I focused on introducing the mathematics required to tackle the problem at hand. The work covered some brief theory on monoid rings, such as definitions. Kulosman et. al. focused on how factorization of the binomial $X^\pi - 1$ in these monoid rings changes when we apply restrictions to the exponent monoid; specifically, for some submonoid M of Q_0^+, and for any field F of characteristic zero, we have that the binomial $X^\pi - 1$ is irreducible over $F[X;M]$, provided that $\pi \in M$ is an indivisible element[1]. A natural generalization is then to modify not just the monoid, but the coefficient field, considering a finite field of now prime characteristic. However, as shown in the presentation, factorizations such as $X^7 - 1$ in $F_p[X;M]$ immediately fail. However, it is unclear whether this indicates that factorization of such a binomial in a finite field takes a different form, or that such a binomial is irreducible over this new monoid ring. If the latter is the case, to what extent

is factorization of elements lost with this new restrictions, and how can we quantify such cases?

In the preliminary presentation, I introduced Dr. Daileda's work on the exceptional primes. We construct a set $E(p)$, the set of primes q such that $X^q - 1$ factors over $F_p[X; M]$. This is by definition, the irreducible powers that are the exceptions to Kulosman's theorem after a change of coefficient field. With some clever number-theoretic arguments, which are honestly quite above my head, Dr. Daileda showed that each $E(p)$ is an infinite set, and additionally a way to completely describe the elements of $E(2)$ and $E(3)$[2]. The problem is that his results do not work for $E(5)$ and beyond, and the goal of this project was to create an algorithm to generate the elements of $E(p)$ for any given p prime.

In my final presentation, I recalled the factorization over $Q[X]$,

$$X^q - 1 = (X - 1)(X^{q-1} + X^{q-2} + \cdots + X + 1). \tag{1}$$

Naturally, (1) no longer holds for $M = N \setminus \{1\}$, but regardless, this factorization over the unrestricted monoid gives us plenty of information for $F_p[X; M]$. Indeed, we denote the right hand term as the qth cyclotomic polynomial, $\Phi_q(X)$. This polynomial is irreducible $Q[X]$ as well as $Z[X]$, but not in $F_p[X]$. Specifically, the terms that appear in the factorization of $\Phi_q(X)$ can tell us whether $X^q - 1$ factors in $F_p[X; M]$. As such, we can greatly streamline our code for a more efficient approach, which will now be discussed below.

2.2 The Code

The first step of our algorithm is to first find the factors of $\Phi_q(X)$. We chose to use Maple, as the access to efficient factorization of polynomials, especially those in finite fields, saved us quite a bit of work. We define a list F, to contain the factors of $\Phi_q(x)$ as follows:

$$F := (\text{Factors(NumberTheory:-Phi}(q, X)) \bmod p)[2];$$

The "Factors" function returns an array of two elements; the first element being the sign of the polynomial being factored, and the second element an array of ordered pairs, the first term corresponding to the factor, and the second its exponent. The exponent of the ordered pair is not necessary, so we remove these by $F[i] : F[i][1]$.

The next step is to iterate through the possible divisors of $\Phi_q(X)$ generated by the terms in F. As mentioned in the final slides, not only do we know the total number of factors ahead of time, but each term in the factorization of $\Phi_q(X)$ has multiplicity of 1. As such, the generation of these divisors correspond to iterating through binary sequences from 0 to $2^r - 1$, where r is the size of the array F. To do this, we first have an index j, which we convert to a binary sequence of r digits by the following:

$$b := \text{map2(nprintf,cat("\%0",}r\text{,"d"), convert(j,binary));}$$

With this binary number, we then create an array with each digit an entry of this array, as below:

$$E := \text{map(parse,StringTools:-Explode(convert(}b\text{,string)));}$$

This array E is by definition the powers that the factors of $\Phi_q(X)$ will be raised to. We then compute such a term T as

$$\text{for } k \text{ from 1 to } s \text{ do } T := T * F[k]^{E[k]};$$

Lastly, we simply check if this polynomial T has linear terms; if it does not, we return 1, otherwise we return 0;

$$T := \text{expand}(T); \text{ if evalb(coeff}(T, X, 1) = 0 \bmod \text{p) then return 1 fi;}$$

Putting these functions into a for-loop, we evaluate every factor combination for $\Phi_q(X)$, and define such a procedure to determine if a prime q is exceptional for p. Thus, we have the following algorithm:

```
> isExceptional := proc (q,p)
      local F := (Factors(NumberTheory:-Phi(q,X)) mod p)[2];
      local r := nops(F);
      local i := 1;
      for i from 1 to r do
          F[i] := F[i][1];
      end do;
      local j := 1;
      for j from 1 to 2^r - 1 do
          local b := map2(nprintf,cat("%0",r,"d"),convert(j,binary));
          local E := map(parse,StringTools:-Explode(convert(b,string)));
          local s := nops(E);
          local k := 1;
          local T:= 1;
          for k from 1 to s do
              T := T * (F[k]^E[k]);
          end do;
          T := expand(T);
          if evalb(coeff(T,X,1) = 0 mod p) then return 1 fi;
      end do;
      return 0;
  end proc;
```

Figure 1: isExceptional Function written in Maple

Given a field F of characteristic p, isExceptional(q,p) from Fig. 1 will return true if $X^q - 1$ if factors in $F_p[X; M]$, and false otherwise. Using another for-loop, we iterate over a sequence of prime numbers, and define a set of the primes satisfying Fig. 1, as shown below:

```
> Exceptionals := proc(p,n)
      local P:={};
      for i from 2 to n do
          if isprime(i) then
              if evalb(isExceptional(i,p) = 1) then
                  P := P union {i};
              end if;
          end if;
      end do;
  return P;
  end proc;
```

Figure 2: Exceptionals Function written in Maple

For example, we compute $E(2)$ and $E(3)$ below, and see that the results indeed match the exceptionals found in Dr. Daileda's preprint[2].

```
> E_2 := Exceptionals(2, 100)
```

$$E\_2 := \{7, 17, 23, 31, 41, 43, 47, 71, 73, 79, 89, 97\}$$

```
> E_3 := Exceptionals(3, 100)
```

$$E\_3 := \{11, 13, 23, 37, 41, 47, 59, 61, 71, 73, 83, 97\}$$

Figure 3: the sets $E(2)$ and $E(3)$ computed in Maple.

2.3 Challenges

While not entirely unforseen, there were several issues that I had to overcome. The biggest one being that I have never used a computer algebra system before. While I consider myself to be an avid programmer, it wouldn't be an underestimate to say more than half of my time on this project was spent learning the syntax for maple. Since Maple doubles as a programming language and a functional calculation languages, functions such as "evalb" are used to distinguish calculations from logical comparisons, as they use the same syntax. Another issue that occured from this was the computation of the binary digits b. For example, if I needed the binary sequence "0011011", Maple would store this as an integer with the value $b = 11011$, the absence of the leading zeroes throwing off some calculations, initially leading me to skip certain primes in $E(p)$. However, my knowledge of low-level languages such as C proved to be useful, as I used formatting codes to force the b to be stored as an r-bit integer, doing so by map2(nprintf,cat("%0",r,"d"),convert(j,binary)). Other than these few hiccups, the programming was rather smooth-sailing. I plan to continue work on this project, by calculating distributions of these primes, which I will most likely work on for the remainder of the semester and Summer.

3 Closing Thoughts

A personal thought that I had from this project, extending beyond just my own but towards pure math presentations and projects in general, is that I think a lack of the usual "tangibility" that comes with mathematics not applied to some science or physical field, can impair the the sense of applicability that's portrayed to the other students. I built this opinion from two things: my two previous presentations, and my personal experience of the opposite. Note that I do not believe this is a bad aspect of math; pure math is by nature investigated without the incentive or goal of application, and typically happens as a byproduct. Instead, I believe that when researching pure math, and especially when presenting such results to peers, researchers need to take extra care to consider such applications. This is but a shallow opinion I've formed from my few attempts at math research, both being number theoretic projects, and perhaps this is a different case in the professional mathematics circles, but at the university level and below I feel a sort of incentive to apply such results to my peers. I'm sure there are students in the opposite situation as me, who thrive in these applied projects and do not receive as much satisfaction from the abstract. Nevertheless, this has lead to some mixed results on the content of my presentation, which of course is up to interpretation, as I've had my own opinions as well. All and all, this research project was extremely valuable to myself and the way I think about math, and I definitely think this has changed my perspective on mathematics. Like Matt's chess presentation, the lessons one learns from these projects are just as important as the research itself, and this experience has made me mature as self-proclaimed mathematician.

References

[1] K. Christensen H. Kulosman, R. Gipson. Irreducibility of certain binomials in semigroup rings for nonnegative rational monoids. *Int. Electron. J. Algebra*, 24:50–61, 2018.

[2] R. C. Daileda. On the irreducibility of $x^q - 1$ in monoid rings with positive characteristic. *preprint, https://arvix.org/abs/2112.09080.*