JTAMV10N2-9

by Juli Lusi

Submission date: 15-Sep-2019 02:11PM (UTC+0530)
Submission ID: 1172810310
File name: 9._JTAM-V3N2-Butar_132-142.pdf (1.02M)
Word count: 4404
Character count: 27205

Jurnal Teori dan Aplikasi Matematika (JTAM)
http://journal.ummat.ac.id/index.php/jtam
p-ISSN 2597-7512 | e-ISSN 2614-1175
Vol. 3, No. 2, Oktober 2019, Hal. 132-142

Faktorisasi Polinomial *Square-Free* dan Bukan *Square-Free* atas Lapangan Hingga $\mathbb{Z}_p$

Juli Loisiana Butar-butar[1], Ferdinand Sinuhaji[2]
[1,2]Matematika, Universitas Quality Berastagi, Indonesia
julois.butrz@gmail.com[1], sinuhajiferdinand@gmail.com[2]

INFO ARTIKEL

Riwayat Artikel:

Diterima: 29-08-2019
Disetujui: 01-10-2019

Kata Kunci:

Faktorisasi Polinomial;
Lapangan Hingga;
Algoritma Berlekamp;
Polinomial *Square-free*

Keywords:
Polynomial Factorization;
Finite Field,
Berlekamp Algorithm;
Square-free Polynomial.

ABSTRAK

Abstrak: Faktorisasi polinomial atas lapangan hingga $\mathbb{Z}_p$ memerlukan suatu metode yang tepat yakni algoritma untuk memproses faktorisasi polinomial. Algoritma Faktorisasi Berlekamp merupakan salah satu metode terbaik dalam memfaktorisasi polinomial atas lapangan hingga $\mathbb{Z}_p$. Polinomial atas lapangan terbagi dua kategori berdasarkan faktorisasinya, yaitu polinomial *square-free* dan polinomial bukan *square-free*. Polinomial *square-free* adalah polinomial yang tidak mempunyai faktor berulang. Sedangkan polinomial bukan *square-free* adalah sebaliknya. Penelitian ini bertujuan untuk membuat suatu algoritma untuk menfaktorkan polinomial *square-free* dan bukan *square-free* atas lapangan hingga. Adapun yang menjadi referensi utama dalam penelitian ini adalah berdasarkan (Divasòn, Joosten, Thiemann, & Yamada, 2017). Namun, dibatasi hanya untuk polinomial *square-free* saja. Metode yang digunakan adalah menggabungkan algoritma faktorisasi *square-free* dengan algoritma Berlekamp sehingga diperoleh algoritma baru. Pada bagian akhir penelitian adalah mengimplementasikan algoritma baru yang telah disusun.

Abstract: *Polynomial factorization over finite field $\mathbb{Z}_p$ needs a right method that is an algorithm to process polynomial factorization. Algorithm Factorization Berlekamp is one of the best methods in factoring polynomials over a finite field $\mathbb{Z}_p$. Polynomials over field are divided into two category based on its factorization, namely square-free polynomials and not square-free polynomials. Square-free polynomials are polynomials polynomials that do not have repeated factors. When non square-free polynomials are the opposite. This research aims to set an algorithm for factoring square-free polynomials and non-square-free polynomials over a finite field $\mathbb{Z}_p$. The main reference in this research is based on* (Divasòn, Joosten, Thiemann, & Yamada, 2017). *However, it is restricted only to square-free polynomials. The method used is to combine the square-free factorization algorithm with the Berlekamp algorithm to obtain a new algorithm. At the end of the research is to implement the new algorithm that has been compiled.*

DOI: https://doi.org/10.31764/jtam.v3i2.1079

A. LATAR BELAKANG

Faktorisasi polinomial merupakan salah satu penunjang dasar dalam sistem komputer aljabar. Proses ini dibutuhkan dalam aljabar komputasi, lapangan hingga, serta aljabar linier. Salah satu metode yang digunakan untuk memfaktorkan polinomial tersebut adalah dengan metode Euclidean dan faktor persekutuan terbesar. Namun, pada penelitian ini akan dibahas mengenai faktorisasi polinomial pada lapangan hingga $\mathbb{Z}p$.

Salah satu konsep yang sangat penting dalam pembahasan polinomial adalah akar-akar dari polinomial tersebut. Berdasarkan faktorisasi tak tereduksi, polinomial dibagi dalam dua jenis, yaitu polinomial yang faktorisasi tak tereduksinya tanpa perulangan atau disebut juga polinomial *square-free,* dan polinomial yang faktorisasi tak tereduksinya terdapat perulangan atau disebut juga polinomial bukan *square-free.* Konsep polinomial *square-free* ini sama dengan polinomial *separable* (Saropah, 2012).

Diberikan $f(x) = a_n x^d + a_{n-1}x^{d-1} + \cdots + a_1 x + a_0 \in \mathbb{Z}_p[x]$ merupakan suatu polinomial

atas $\mathbb{Z}_p$ berderajat $\deg(f) = d > 1$ dengan $a_d, a_{d-1}, \cdots, a_1, a_0 \in \mathbb{Z}_p$. Faktorisasi polinomial pada lapangan hingga sangat berguna dalam teori pengkodean, dan bahkan merupakan salah satu tahapan pendukung dalam menentukan grup Galois dari suatu polinomial (Butar-butar, 2018).

Berbeda dengan faktorisasi polinomial atas ring bilangan bulat, faktorisasi polinomial atas $\mathbb{Z}_p$ dapat difaktorisasi dengan cara mencoba-coba membagikan $f(x)$ dengan polinomial tak tereduksi lain $\mathbb{Z}_p[x]$ dengan derajat lebih kecil dari d . Namun, faktorisasi dengan cara ini tidak efisien. Oleh karena itu, diperlukan suatu metode khusus berupa algoritma untuk mencari faktor-faktor tak tereduksi dari polinomial tersebut. Salah satu algoritma yang sangat tepat dan efisien dalam memfaktorkan polinomial $f(x)$ adalah algoritma Berlekamp.

Kebanyakan algoritma faktorisasi membatasi input hanya untuk polinomial *square-free.* Demikian juga halnya dengan Algoritma faktorisasi Berlekamp yang dibatasi untuk polinomial *square-free* (Divas´on, 2017). Selain (Divasón, Joosten, Thiemann, & Yamada, 2017), pada (Hanif, 2011) juga membahas algoritma Berlekamp dan beberapa contoh faktorisasi polinomial atas lapangan hingga serta implementasi algoritma tersebut. Pembatasan hanya untuk polinomial *square-free* inilah yang melatar-belakangi peneliti untuk membuat suatu algoritma faktorisasi yang tidak hanya untuk polinomial *square-free* tetapi juga untuk polinomial bukan *square-free.* Aplikasi dari algoritma ini sangat diperlukan dalam kontruksi kode siklik berulang (Cao & Gao, 2014), (Batoul, Guenda, & Gulliver, 2015).

Di lain sisi, (Lee, 2013) membahas tentang algoritma faktorisasi *square-free* yaitu suatu algoritma untuk mencari faktor *square-free* polinomial dan banyak dari perulangan faktor tersebut. Namun faktor-faktor tersebut belum tentu tak tereduksi. Dengan menyusun langkah-langkah berdasarkan algoritma faktorisasi *square-free* dan algoritma Berlekamp diperoleh algoritma baru.

Dengan demikian yang menjadi tujuan penelitian ini adalah menyusun algoritma faktorisasi polinomial untuk polinomial *square-free* dan polinomial bukan *square-free* dan kemudian mengimplementasikannya.

B. TINJAUAN PUSTAKA

1. Algoritma Faktorisasi Berlekamp

Dalam (Childs, 2009) dijelaskan hal yang melatarbelakangi dari Algoritma Berlekamp untuk memfaktorkan polinomial monik $f(x)$ di $\mathbb{Z}_p[x]$ berderajat d didasarkan pada penentuan polinomial non-konstan $h(x)$ berderajat lebih kecil atau sama dengan d sehingga $f(x)$ membagi habis $h(x)^p - h(x)$ sehingga kemudian diperoleh faktorisasi $f(x)$. Polinomial monik adalah polinomial yang pangkat tertinggi sama dengan satu (Mursyidah, 2017).

Teorema 1. Diberikan polinomial monik *square-free* $f(x)$ di $\mathbb{Z}_p[x]$ berderajat $d > 1$. Jika terdapat $h(x)$ di $\mathbb{Z}_p[x]$ polinomial berderajat berderajat e dengan $1 \le e < d$ sedemikian hingga $f(x)$ membagi habis $h(x)^p - h(x)$, maka

$$f(x) = \gcd\big(f(x), h(x)\big) \cdot \gcd(f(x), h(x) - 1) \cdots \gcd\big(f(x), h(x) - (p-1)\big) \quad (1)$$

adalah faktorisasi nontrivial dari $f(x)$ di $\mathbb{Z}_p[x]$.

Bukti. Berdasarkan yang diketahui $f(x)$ membagi habis $h(x)^p - h(x)$ diperoleh dua fakta.

Fakta pertama, berdasarkan Teorema Fermat, $u^p - u$ mempunyai akar sebanyak p di $\mathbb{Z}_p$, dengan $u = 0,1,2,\cdots, p-1$. Kemudian berdasarkan Teorema Akar, $u^p - u$ terfaktorisasi di $\mathbb{Z}_p$ menjadi

$$u^p - u = u(u-1)(u-2)\cdots\big(u-(p-1)\big).$$

Dibentuk $u = h(x)$ sehingga menghasilkan

$$h(x)^p - h(x) = h(x)(h(x)-1)(h(x)-2)\cdots \big(h(x)-(p-1)\big).$$

Sehingga $h(x)^p - h(x)$ merupakan perkalian dari polinomial-polinomial yang saling prima relatif di $\mathbb{Z}_p[x]$.

Fakta kedua, jika a dan b adalah polinomial saling prima relatif di $F[x]$, dengan F lapangan, maka setiap polinomial g di $F[x]$,

$$\gcd(g, ab) = \gcd(g, a) \cdot \gcd(g, b).$$

Berdasarkan induksi, fakta ini digeneralisasi untuk kasus dari faktor persekutuan terbesar dari g dan yang lebih dari dua pasangan faktor saling prima relatif. Karena $f(x)$ membagi habis $h(x)^p - h(x)$, diperoleh

$$f(x) = \gcd\big(f(x), h(x)^p - h(x)\big).$$

Karena $h(x)-r$ dan $h(x)-s$ saling prima relatif untuk $r,s \in \{0,1,2,\cdots,p-1\}$ dan $r \neq s$, diperoleh

$$\begin{aligned} f(x) &= \gcd\big(f(x), h(x)^p - h(x)\big) \\ &= \gcd(f(x), (h(x)(h(x)-1)\cdots \\ &\quad \big(h(x)-(p-1)\big)\big) \\ &= \gcd\big(f(x), h(x)\big) \cdot \gcd(f(x), h(x)-1)\cdots \\ &\quad \gcd\big(f(x), h(x)-(p-1)\big). \end{aligned}$$ ■

Algoritma faktor persekutuan terbesar sangat menentukan dalam faktorisasi polinomial (Temnhurne & Sathe, 2016). Pada faktorisasi dari Teorema 1, faktor persekutuan terbesar dapat ditentukan secara efisien dengan Algoritma Euclid (Iliev & Kyurkchiev, 2018) di $\mathbb{Z}_p[x]$.

Untuk memfaktorkan $f(x)$ di $\mathbb{Z}_p[x]$ dengan menggunakan cara pada Teorema 1., maka dicari polinomial $h(x)$ berderajat e dengan $1 \leq e < d$, sedemikan hingga $f(x)$ membagi habis $h(x)^p - h(x)$. Ini dilakukan dengan membentuk dan menyelesaikan suatu sistem persamaan linear untuk memperoleh koefisien dari $h(x)$. Hal ini dijamin dalam teorema berikut ini.

Teorema 2. Diberikan polinomial monik *square-free* $f(x)$ di $\mathbb{Z}_p[x]$ berderajat $d > 1$. Diberikan $h(x)$ di $\mathbb{Z}_p[x]$ yang memenuhi Teorema 1 dan Q matriks berukuran $d \times d$ dengan baris ke- i merupakan vektor dari koefisien dari polinomial sisa

$$r_i(x) = x^{ip} \bmod f(x)$$

untuk $i = 0,1,\cdots,d-1$, maka koefisien-koefisien polinomial $h(x)$ merupakan solusi persamaan linear homogen $\boldsymbol{b}(Q-I) = \mathbf{0}$ dengan I adalah matriks identitas berukuran $d \times d$.

Bukti. Dimisalkan

$$h(x) = b_0 + b_1 x + \cdots + b_{d-1}x^{d-1}, \qquad (2)$$

dimana $b_0, b_1, \cdots, b_{d-1}$ di $\mathbb{Z}_p$ merupakan koefisien-koefisien yang ditentukan. Berdasarkan Proposisi 12 ((Childs, 2009), hal 204), diperoleh

$$h(x)^p = b_0^p + b_1^p x^p + b_2^p x^{2p} + \cdots + b_{d-1}^p x^{(d-1)p}.$$

Berdasarkan Teorema Fermat, $b^p = b$ untuk semua b di $\mathbb{Z}_p$ sehingga

$$\begin{aligned} h(x)^p &= b_0 + b_1 x^p + \cdots + b_{d-1}x^{(d-1)p} \\ &= k(x^p). \end{aligned} \qquad (3)$$

Untuk menentukan sisa dari $h(x)^p$ dibagi $f(x)$, cari $x^{ip} \bmod f(x)$ untuk $i = 0,1,\cdots,d-1$

$$x^{ip} = f(x)q_i(x) + r_i(x), \qquad (4)$$

dengan $\deg r_i(x) < d = \deg f(x)$. Oleh karena itu,

$$x^{ip} = r_i(x)\big(\bmod f(x)\big), \qquad (5)$$

sehingga persamaan (3) menjadi

$$\begin{aligned} h(x)^p = (b_0 r_0(x) + b_1 r_1(x) + \cdots \\ + b_{d-1} r_{d-1}(x))\big(\bmod f(x)\big) \end{aligned} \qquad (6)$$

Akibatnya, $f(x)$ membagi habis $h(x)^p - h(x)$ jika dan jika $f(x)$ membagi habis polinomial

$$\begin{aligned} b_0 r_0(x) + b_1 r_1(x) + \cdots + b_{d-1} r_{d-1}(x) \\ -[b_0 + b_1 x + \cdots + b_{d-1}x^{d-1}] \end{aligned} \qquad (7)$$

Tetapi polinomial (7) berderajat $\leq d-1$, sehingga habis dibagi oleh $f(x)$ jika dan hanya jika polinomial (7) merupakan merupakan poliomial nol di $\mathbb{Z}_p[x]$. Kondisi ini akan digunakan untuk menentukan koefisien-koefiesien $b_0, b_1, \cdots, b_{d-1}$ dari $h(x)$. Dengan kata lain, $b_0, b_1, \cdots, b_{d-1}$ harus memenuhi

$$\begin{aligned} b_0 r_0(x) + b_1 r_1(x) + \cdots + b_{d-1} r_{d-1}(x) \\ -[b_0 + b_1 x + \cdots + b_{d-1}x^{d-1}] = 0 \end{aligned} \qquad (8)$$

Jika koefisien-koefisien dari $1, x, x^2, \cdots, x^{d-1}$ pada persamaan (8), diperoleh sebanyak d persamaan-persamaan linier simultan di d tidak diketahui $b_0, b_1, \cdots, b_{d-1}$ dimana koefisien-koefisien di $\mathbb{Z}_p$ dari persamaan b_j pada persamaan merupakan koefisien dari polinomial sisa $r_j(x)$.

Solusi sistem persamaan ini menyajikan elemen-elemen $b_0, b_1, \cdots, b_{d-1}$ yang merupakan koefisien-koefisien dari polinomial $h(x)$ sedemikian hingga $f(x)$ membagi habis $h(x)^p - h(x)$.

Tranformasi bentuk persamaan (8) menjadi bentuk matriks. Dimisalkan I matriks identitas $d \times d$, dimisalkan

$$r_i(x) = r_{i,0} + r_{i,1}x + r_{i,2}x^2 + \cdots + r_{i,d-1}x^{d-1}$$

untuk setiap i sehingga diperoleh matriks yang tak lain adalah matriks Q. Dengan demikian

$$\mathbf{Q} = \begin{bmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,d-1} \\ r_{1,0} & r_{1,1} & \cdots & r_{1,d-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{d-1,0} & r_{d-1,1} & \cdots & r_{d-1,d-1} \end{bmatrix} \qquad (9)$$

merupakan matriks dengan setiap baris merupakan koefisien-koefisien dari polinomial sisa $r_0(x), r_1(x), \cdots, r_{d-1}(x)$. Akibatnya, dengan memeriksa komponen dari vektor $\mathbf{b} = (b_0, b_1, \cdots, b_{d-1})$ diperoleh bahwa koefisien-koefisien $h(x)$ merupakan solusi persamaan homogen $\boldsymbol{b}(\mathbf{Q}-\mathbf{I}) = \mathbf{0}$.

■

Dengan menggunakan Teorema 1 dan Teorema 2 diperoleh teorema berikut.

Teorema 3. Algoritma Faktorisasi Berlekamp
Diberikan $f(x)$ polinomial monik *square-free* di $Z_p[x]$ berderajat d. Diberikan Q matriks berukuran $d \times d$ dengan baris ke-i merupakan vektor dari koefisien dari polinomial sisa $r_i(x) = x^{ip} mod\ f(x)$ untuk $i = 0,1,\cdots,d-1$.
Diberikan $\boldsymbol{b} = (b_0, b_1, \cdots, b_{d-1})$ adalah solusi dari

$$\boldsymbol{b}(Q - I) = \mathbf{0},$$

dan $h_1(x) = b_0 + b_1x + \cdots + b_{d-1}x^{d-1}$. Jika $h(x)$ berderajat ≥ 1, maka terdapat s di $\mathbb{Z}_p$ sehingga $h_1(x) - s$ dan $f(x)$ mempunyai suatu faktor persekutuan berderajat lebih besar atau sama dengan 1.

Bukti. Akan dibuktikan dengan menggunakan aturan kontradiksi. Asumsikan untuk semua s di $\mathbb{Z}_p$ $h_1(x) - s$ dan $f(x)$ tidak mempunyai faktor persekutuan berderajat lebih kecil dari 1. Ini berarti, $\gcd(f(x), h_1(x) - s) = 1$ dengan $s = 1,2,\cdots,p-1$.
Akibatnya,

$$\gcd\big(f(x), h_1(x)\big) \cdot \gcd(f(x), h_1(x) - 1) \cdots \gcd\big(f(x), h_1(x) - (p-1)\big) = 1.$$

Karena $f(x)$ polinomial berderajat yang tidak sama dengan nol, maka

$$\gcd\big(f(x), h_1(x)\big) \cdot \gcd(f(x), h_1(x) - 1) \cdots \gcd\big(f(x), h_1(x) - (p-1)\big) \neq f(x).$$

Akibatnya hal ini kontradiksi Teorema 1 Sehingga berdasarkan Teorema 1 diperoleh untuk semua $h(x)$ di $\mathbb{Z}_p[x]$ merupakan polinomial berderajat ≥ 1 dan $\leq d$ sedemikian hingga $f(x)$ tidak membagi habis $h(x)^p - h(x)$. Karena $h_1(x)$ di $\mathbb{Z}_p[x]$, maka $f(x)$ tidak membagi habis $h_1(x)^p - h_1(x)$. Ini berarti, $\gcd(f(x), h_1(x)^p - h_1(x)) = 1$. Ini berarti, $f(x)$ tidak membagi habis $h_1(x)^p - h_1(x)$. Akibatnya, $f(x)$ tidak membagi habis polinomial

$$b_0r_0(x) + b_1r_1(x) + \cdots + b_{d-1}r_{d-1}(x) - [b_0 + b_1x + \cdots + b_{d-1}x^{d-1}] \quad (10)$$

Kemudian berdasarkan Teorema Akar, diperoleh

$$h_1(x)^p - h_1(x) = h_1(x)(h_1(x) - 1)(h_1(x) - 2) \cdots \big(h_1(x) - (p-1)\big).$$

Karena $h_1(x) - r$ dan $h_1(x) - s$ saling prima untuk $r, s \in \mathbb{Z}_p$ dan $r \neq s$, maka diperoleh

$$\begin{aligned}&\gcd\big(f(x), h_1(x)^p - h_1(x)\big)\\ &= \gcd(f(x), h_1(x)(h_1(x) - 1)(h_1(x) - 2) \cdots h_1(x) - (p-1))\\ &= \gcd\big(f(x), h_1(x)\big) \cdot \gcd(f(x), h_1(x) - 1) \cdots \gcd\big(f(x), h_1(x) - (p-1)\big)\\ &\neq f(x)\end{aligned}$$

Akibatnya, $f(x)$ membagi habis $h(x)^p - h(x)$ jika dan jika $f(x)$ membagi habis polinomial (10).
Karena polinomial (10) berderajat $\leq d-1$ dan tidak habis dibagi oleh $f(x)$, maka polinomial (10) merupakan bukan merupakan poliomial nol di $\mathbb{Z}_p[x]$. Hal ini kontradiksi dengan diketahui $\boldsymbol{b}(Q - I) = \mathbf{0}$. Dengan demikian, asumsi salah. Ini berarti, terdapat s di $\mathbb{Z}_p$ sehingga $h_1(x) - s$ dan $f(x)$ mempunyai suatu faktor persekutuan berderajat lebih besar atau sama dengan 1.
■

Dari proses pembuktian Teorema 3 dapat disimpulkan bahwa bagian terpenting adalah menyelesaikan sistem persamaan linear homogen $\boldsymbol{b}(Q - I) = \mathbf{0}$ dengan $\boldsymbol{b}$ adalah vektor dengan panjang d, $Q - I$ adalah matriks berukuran $d \times d$, dan $\mathbf{0}$ adalah vektor nol dengan panjang d. Sistem persamaan linear homogen tersebut dapat juga diselesaikan dengan cara menyelesaikan bentuk sistem persamaan linear homogen

$$\begin{aligned}\boldsymbol{b}(Q - I) &= \mathbf{0}\\ \big(\boldsymbol{b}(Q - I)\big)^T &= \mathbf{0}^T\\ (Q - I)^T\boldsymbol{b}^T &= \mathbf{0}^T \quad (11)\end{aligned}$$

Oleh karenanya, sistem persamaan linear homogen pada persamaan (11) berhubungan dengan mencari ruang null dari matriks $(Q - I)^T$. Karena matriks $(Q - I)^T$ atas lapangan hingga, maka peneliti akan menggunakan hasil pada (Abdel-Ghaffar, 2012) sebagai referensi untuk menentukan solusi sistem persamaan linear homogen. Referensi ini dipakai dalam proses komputasi.

Suatu polinomial tak tereduksi atas $\mathbb{Z}_p$ juga dapat dideteksi dengan algoritma Berlekamp. Ini dapat menjadi suatu bagian untuk mengkontruksi suatu polinomia tak tereduksi atas lapangan hingga (Couveignes & Lercier, 2013). Namun, algoritma Berlekamp tidak dapat dipakai untuk lapangan perluasan $\mathbb{Z}_{p^k}$ untuk $k \in \mathbb{Z}$ sehingga diperlukan algoritma lain yang lebih tepat (Divasón, Joosten, Thiemann, & Yamada, 2019).

C. METODE PENELITIAN

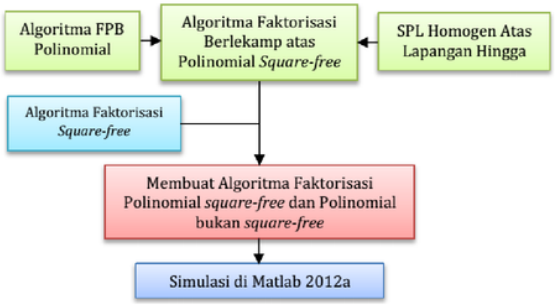
Bagian utama yang menjadi bagian dasar penelitian ini adalah Algoritma faktorisai Berlekamp yang berlaku untuk polinomial *square-free* atas $\mathbb{Z}_p$. Algoritma Pembagian pada polinomial, Metode *Euclidean* dan Faktor Persekutuan Terbesar untuk memfaktorisasi polinomial (Oktafia, Gemawati , &

Endang, 2014) adalah metode yang dipakai untuk menyelesaikan salah satu langkah dari Algoritma faktorisai Berlekamp.

Metode Operasi baris elementer yang telah diformalisasikan dalam (Aransay & Divasón, 2016) berlaku untuk lapangan hingga. Solusi Sistem Persamaan Linear Homogen atas lapangan hingga dengan menggunakan Metode Operasi baris elementer atas lapangan hingga (Abdel-Ghaffar, 2012) sebagai sistem persamaan linear homogen. Dengan metode operasi baris elementer dapat juga ditemukan banyak rank dari suatu matriks atas lapangan hingga (Grout, 2010) yang akibatnya banyak ruang null pun dapat diperoleh.

Pembatasan Algoritma Berlekamp hanya untuk polinomial *square-free* mengharuskan penelitian ini menggunakan metode tambahan lain. Metode ini adalah algoritma untuk menentukan polinomial faktorisasi *square-free* dan banyak polinomial tersebut dari suatu polinomial yang disebut dengan Algoritma *square-free factorization* (Lee, 2013). Selanjutnya, peneliti membuat suatu algoritma baru berdasarkan Algoritma *square-free factorization* dan Algoritma Berlekamp.

Kemudian sebagai bagian akhir penelitian, peneliti telah mengimplementasi algoritma tersebut dalam program Matlab 2012a dimana setiap langkah yang memerlukan penyelesaian secara khusus dapat dijalankan dengan fungsi yang sudah ada di Matlab 2012a atau peneliti membuat fungsi khusus untuk langkah itu. Salah satu fungsi Matlab yang harus diimplementasikan adalah operasi baris elementer atas lapangan $\mathbb{Z}_p$. Dalam operasi baris elementer merupakan metode eliminasi Gauss-Jordan merupakan salah satu langkah dan proses ini telah diimplementasikan dalam Matlab (Irwan, 2017). Lebih jelas mengenai metode penelitian, perhatikan Gambar 1 berikut.



Gambar 1. Diagram Alir Penelitian.

D. HASIL DAN PEMBAHASAN

1. Algoritma Faktorisasi *Square-Free*

Kebanyakan algoritma faktorisasi polinomial atas lapangan hingga mengharuskan polinomial input tidak memiliki faktor berulang atau disebut juga polinomial *square-free* (Lee, 2013). Salah satunya adalah Algoritma Berlekamp yang dibatasi hanya untuk polinomial *square-free* saja.

Berikut ini merupakan algoritma Berlekamp yang terdapat di (Divasón, Joosten, Thiemann, & Yamada, 2017).

Algoritma 1. Faktorisasi Berlekamp

Input: f polinomial *square-free* atas lapangan $\mathbb{Z}_p$ berderajat d.

Output: Konstanta c dan himpunan polinomial monik tak tereduksi $F = \{f_1, f_2, \cdots, f_n\}$ sehingga $f(x) = c \cdot f_1(x) \cdot f_2(x) \cdots f_n(x)$.

1. Bentuk $F = \{\}$
2. c adalah koefisien pangkat tertinggi dari $f(x)$. Bentuk $f(x) \coloneqq \frac{f(x)}{c}$.
3. Hitung matriks Berlekamp $Q \in \mathbb{Z}_p^{d\times d}$ untuk $f(x)$, dengan baris ke-i adalah vektor dari koefisien polinomial $x^{p\cdot i} \bmod f$ untuk $i = 0,1,2, \dots, p-1$.
4. Hitung dimensi r dan basis $b_1, b_2, \cdots, b_r$ dari ruang null $(Q-I)^T$, dimana I adalah matriks identitas berukuran $d \times d$.
5. Untuk setiap vektor $b_i = (b_{i1}, b_{i2}, \cdots, b_{id})$ bentuk menjadi polinomial $h_i(x) = b_{i1}x^0 + b_{i2}x^1 + \cdots + b_{id}x^{d-1}$.
6. Bentuk $G \coloneqq \{f\}, H = \{h_1, \cdots, h_r\} \setminus \{1\}, G_I \coloneqq \emptyset$.
7. Jika $|G| = r \vee H = \emptyset$, kembali c dan $G \cup G_I$.
8. Pilih $h \in H$ dan perbarui $H = H \setminus \{h\}$. Perbarui
$$G \coloneqq \{f_i \in G, 0 \le j < p, \; g_{i,j} = \gcd(f_i, h_i - j), g_{i,j} \neq 1\}$$
9. Perbarui $F = F \cup G$ dan $F_I = G_I$
10. Jika ditemukan polinomial tak tereduksi sebanyak k di F, maka pindahkan ke F_I dan perbarui $r \coloneqq r - k$. Kembali ke langkah 7. Jika tidak ke langkah 10
11. Hasil adalah konstanta c dan himpunan F.
12. Selesai.

Namun, ada suatu algoritma untuk menentukan faktorisasi *square-free* dari polinomial input dan banyak pergandaan dari faktorisasi tersebut. . Algoritma ini disebut dengan Algoritma *square-free factorization*. Berikut ini adalah Algoritma *square-free factorization* dengan mengubah bentuk algoritma dari yang dibahas (Lee, 2013).

Algoritma 2. Algoritma *square-free factorization*
Input: f polinomial monik atas lapangan $\mathbb{Z}_p$ berderajat d.
Output: $FS = \{(g_i, e_i) | g_i$ faktorisasi $square-free$ dari $f, e_i \in \mathbb{N} \ni f(x) = \prod g_i^{e_i}(x)\}$.

1. Bentuk $FS = \{\}$.
2. Tentukan $g = f'$ atas $\mathbb{Z}_p$
3. Jika $g = 0$,maka polinomial menjadi
   $fp(x) \coloneqq f\left(x^{\frac{1}{p}}\right)$.
   Kemudian $FS = FS \cup \{(fp, \deg(f) / \deg(fp)\}$.
   Lanjut ke langkah 23.
   Jika tidak, lanjut ke langkah 4.
4. Hitung $k = \gcd(f, g)$ atas $\mathbb{Z}_p$.
5. Tentukan polinomial $h = f/k$ atas $\mathbb{Z}_p$ dan $dh = \deg(h)$.
6. Jika $d \pmod{dh} \coloneqq 0$, maka
   $ph(x) = \left(h(x)\right)^{\frac{d}{dh}}$ atas $\mathbb{Z}_p$.
   Lanjut ke langkah 7.
   Jika tidak, lanjut ke langkah 8.
7. Jika $ph \coloneqq f$, maka $FS = FS \cup \left\{\left(h, \left(\frac{d}{dh}\right)\right)\right\}$.
   Lanjut ke langkah 22.
   Jika tidak lanjut ke langkah 8.
8. Tentukan $gh = \gcd(h, th)$ atas $\mathbb{Z}_p$.
9. $i = 1$.
10. Hitung $y = \gcd(k, h)$ atas $\mathbb{Z}_p$.
11. Tentukan $fac = h/y$ atas $\mathbb{Z}_p$.
12. Jika $fac = 1$, maka $fac \coloneqq \{\}$. $FS \coloneqq FS \cup fac$.
    Lanjut ke langkah 14.
    Jika tidak lanjut ke langkah 13.
13. Bentuk $FS = FS \cup \{(fac, i)\}$.
14. Ubah $h \coloneqq y$.
15. Ubah $k \coloneqq k/y$ atas $\mathbb{Z}_p$.
16. Jika $k = 1$, bentuk polinomial rf yang dibentuk dari hasil kali semua polinomial di FS dan perulangannya. Lanjut ke langkah 17. Jika tidak, lanjut ke langkah 19.
17. Ubah $k \coloneqq f/rf$ atas $\mathbb{Z}_p$, k merupakan polinomial bukan *square-free* yang hanya memiliki satu faktorisasi.
18. Tentukan $tk = k'$ atas $\mathbb{Z}_p$.
    Hitung $gk = \gcd(k, tk)$ atas $\mathbb{Z}_p$.
    Hitung $hk = k/gk$ atas $\mathbb{Z}_p$
19. Bentuk $FS \coloneqq FS \cup \left\{\left(hk, \frac{\deg(k)}{\deg(hk)}\right\}\right.$. Lanjut ke langkah 22.
20. $i = i + 1$. Jika $h \neq 1$, kembali ke langkah 10. Jika tidak, lanjut ke langkah 21.
21. Tentukan $gk = k'$ atas $\mathbb{Z}_p$
22. Jika $gk = 0$,maka polinomial menjadi
    $kp(x) \coloneqq k\left(x^{\frac{1}{p}}\right)$.
    Kemudian $FS = FS \cup \{(fp, \deg(k) / \deg(kp)\}$.
    Lanjut ke langkah 23.
23. Selesai.

Dengan menggabungkan Algoritma 1 dan 2, selanjutnya dibentuk suatu algoritma baru yang menjadi penyelesaian dari faktorisasi untuk polinomial *square-free* dan bukan *square-free* atas $\mathbb{Z}_p$ yang akan dibahas pada bagian berikutnya.

2. Algoritma Faktorisasi Polinomial *Square-Free* dan bukan *Square-Free*

Pada bagian ini, dibentuk algoritma baru yang merupakan gabungan Algoritma 1 dan 2 sebagai penyelesaian dari faktorisasi untuk polinomial *square-free* dan bukan *square-free* atas $\mathbb{Z}_p$.

Algoritma 3. Algoritma Faktorisasi Polinomial *Square-Free* dan bukan *Square-Free*
Input: f polinomial atas lapangan $\mathbb{Z}_p$ berderajat d.
Output: kf adalah koefisien pangkat tertinggi f.
$Fac = \{(fac_i, n_i) | fac$ faktorisasi tak tereduksi dari $f, n_i \in \mathbb{N} \ni f(x) = \prod fac_i^{n_i}(x)\}$.

1. Bentuk $Fac = \{\}$.
2. kf adalah koefisien pangkat tertinggi dari f.
3. Bentuk ulang polinomial f menjadi
   $f \coloneqq \left((kf)^{-1}(f\ )\right) (\text{mod } p)$.
4. Tentukan FS yakni himpunan semua faktorisasi *square-free* dari f dengan menggunakan Algoritma 2.
5. Tentukan m adalah banyaknya faktorisasi *square-free* dari f dari langkah 4.
6. $i = 1$.
7. Tentukan Vf yang merupakan himpunan faktorisasi tak tereduksi polinomial *square-free* dari himpunan $FS = \{(g_i, e_i)\}$ anggota ke- i dengan menggunakan Algoritma 1.
8. Jika $|Vf| = 1$, maka
   $Fac = Fac \cup \{(Vf, e_i\}$.
   Lanjut ke langkah 10. Jika tidak, lanjut ke langkah 9.
9. Tentukan k yang merupakan banyaknya himpunan dari Vf dengan $k \neq 1$.
10. Untuk $j = \{1, \cdots, k\}$

$Fac = Fac \cup \{((\text{anggota ke} - j \text{ dari } Vf), e_i\}$.
11. $i = i + 1$. Jika $i \neq m$,kembali ke langkah 7.
12. Selesai.

Konsep dasar dari algoritma 3 adalah mencari faktor *square-free* dari polinomial f dan banyak perulangannya. Selanjutnya, setiap faktor *square-free* tersebut difaktorisasi lagi dengan menggunakan algoritma Berlekamp. Untuk lebih jelas mengenal Algoritma 3, perhatikan contoh berikut ini.

Contoh 1. Faktorisasi polinomial berikut

$$f(x) = 2x^{13} + x^{12} + x^{10} + 3x^9 + 2x^8 + 2x^6 + 3x^5 + 2x^4 + 2x^3 + x + 4$$

atas $\mathbb{Z}_5$.

Solusi:

Karena koefisien pangkat tertinggi adalah 2, maka polinomial menjadi

$$f(x) = 2^{-1}f(x) \pmod 5$$
$$= x^{13} + 3x^{12} + 3x^{10} + 4x^9 + x^8 + x^6 + 4x^5 + x^4 + x^3 + 3x + 2 \pmod 5$$

Turunan pertama f terhadap x

$$g(x) = 3x^{12} + x^{11} + x^8 + 3x^7 + x^5 + 4x^3 + 3x^2 + 3 \pmod 5$$

Kemudian,

$$k = \gcd(f, g)$$
$$= x^7 + 3x^6 + 4x^4 + 2x^3 + 3x^2 + 2x + 4.$$

Lalu diperoleh

$$h(x) = (f/k) \pmod 5$$
$$= x^6 + 4x^3 + 3x + 3 \pmod 5.$$

dan derajat dari h, $dh = 6$.

Karena $d \pmod{dh} \neq 0$, maka kemudian untuk $i = 1$ diperoleh:

Iterasi $i = 1$

$$y = \gcd(k, h) = x^6 + 4x^3 + 3x + 3 \pmod 5.$$

Lalu diperoleh $fac = h/y = 1$ atas $\mathbb{Z}_5$. Karena $fac = 1$, maka fac bukan faktorisasi yang dimaksud. Kemudian ubah $h = y$ dan $k = k/y$ atas $\mathbb{Z}_5$ sehingga

$$k = x + 3.$$

Ubah $i = i + 1 = 2$.

Iterasi $i = 2$

$$y = \gcd(k, h) = x + 3 \pmod 5.$$

Lalu diperoleh

$$fac = h/y$$
$$= x^5 + 2x^4 + 4x^3 + 2x^2 + 4x + 1 \pmod 5.$$

Karena $fac \neq 1$, maka

$$FS = \{((x^5 + 2x^4 + 4x^3 + 2x^2 + 4x + 1), 2)\}.$$

Kemudian ubah $h = y$ dan $k = k/y$ atas $\mathbb{Z}_5$ sehingga

$$k = 1.$$

Karena $k = 1$, maka bentuk

$$rf = (x^5 + 2x^4 + 4x^3 + 2x^2 + 4x + 1)^2 \pmod 5.$$

Ubah $k := f/rf$ atas $\mathbb{Z}_5$ sehingga

$$k = x^3 + 4x^2 + 2x + 2$$

Kemudian $tk(x) = k'(x) = 3x^2 + 3x + 2$.

Lalu $gk = \gcd(k, tk) = x^2 + x + 4$ sehingga diperoleh $hk = k/gk = x + 3$ atas $\mathbb{Z}_5$.

Akibatnya, faktorisasi *square-free* dari f

$$FS := FS \cup \{(x + 3, 3\}$$
$$:= \left\{\begin{pmatrix}(x^5 + 2x^4 + 4x^3 + 2x^2 + 4x + 1), 2\\ (x + 3, 3)\end{pmatrix}\right\}.$$

Banyaknya faktorisasi *square-free* dari f adalah $m = 2$.

Langkah selanjutnya adalah menentukan faktorisasi tak tereduksi dari polinomial di FS. Karena $x + 3$ tak tereduksi, cukup mencari faktorisasi dari $x^5 + 2x^4 + 4x^3 + 2x^2 + 4x + 1$ dengan Algoritma 1.

Matriks Berlekamp Q adalah

$$Q = \begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 4 & 1 & 3 & 1 & 3\\ 3 & 4 & 3 & 2 & 4\\ 1 & 2 & 0 & 0 & 1\\ 0 & 2 & 3 & 1 & 0\end{bmatrix}$$

sehingga

$$(Q - I)^T = \begin{bmatrix}0 & 4 & 3 & 1 & 0\\ 0 & 0 & 4 & 2 & 2\\ 0 & 3 & 2 & 0 & 3\\ 0 & 1 & 2 & 4 & 1\\ 0 & 3 & 4 & 1 & 4\end{bmatrix}.$$

Dengan operasi baris elementer atas $\mathbb{Z}_5$ diperoleh

$$\begin{bmatrix}0 & 1 & 0 & 3 & 0\\ 0 & 0 & 1 & 3 & 0\\ 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}.$$

Dan ruang null dari $(Q - I)^T$ adalah

$$\left\{\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\2\\2\\1\\0\end{bmatrix}\right\}$$

sehingga $h_1 = 1$ dan $h_2 = x^3 + 2x^2 + 2$.

Selanjutnya, proses cukup untuk h_2 saja.

Kemudian dihitung $\gcd(f, h_2 - j)$ untuk $j = 0,1, \ldots, 4$.

$\gcd(f, h_2 - 0) = 1$,
$\gcd(f, h_2 - 1) = x^2 + 2$,
$\gcd(f, h_2 - 2) = x^3 + 2x^2 + 2x + 3$,
$\gcd(f, h_2 - 3) = 1$, dan
$\gcd(f, h_2 - 4) = 1$.

Sehingga $Vf := \{x^2 + 2, x^3 + 2x^2 + 2x + 3\}$.

Karena semua polinomial di Vf tak tereduksi, maka proses untuk Algoritma 1 selesai.

Karena $|Vf| \neq 1$, maka

$$Fac = \{((x^2+2),2),((x^3+2x^2+2x+3),2)\}$$

Karena $x+3$ tak tereduksi, maka himpunan faktorisasi tak tereduksi f adalah

$$Fac = Fac \cup \{(x+3,3)\} = \left\{ \begin{matrix} ((x^2+2),2), \\ ((x^3+2x^2+2x+3),2), \\ (x+3,3) \end{matrix} \right\}$$

dengan koefisien pangkat tertinggi adalah 2.

3. Implementasi Algoritma sebagai Fungsi Matlab.

Penelitian ini menggunakan Matlab 2012a untuk mengimplementasikan algoritma-algoritma yang telah dibahas sebelumnya. Alasan dari penelitian ini memilih Matlab adalah karena merupakan program yang cukup umum dipakai dan dapat sebagai produk berupa bahan ajar pada mata kuliah Pemrograman (Syaharuddin & Mandailina, 2017). Karena implementasi ini sangat berguna dalam mata kuliah Teori Pengkodean. Salah satunya dalam mengkontruksi kode siklik (Ding, 2017).

Pada dasarnya, ada beberapa algoritma yang bukan merupakan fungsi bawaan Matlab 2012a namun tidak dapat peneliti tampilkan. Fungsi-fungsi Matlab itu adalah sebagai berikut.

- gfinv(a,p) : untuk mencari invers perkalian dari bilangan tak nol $a \in \mathbb{Z}_p$.
- gfgauss(A,p) : untuk menentukan bentuk eliminasi gauss dari matriks A atas $\mathbb{Z}_p$.
- nullgf(A,p) : untuk menentukan basis ruang null dari matriks A atas $\mathbb{Z}_p$.
- vektorpol(v) : untuk mengubah suatu bentuk vector dari koefisien polinomial menjadi polinomial
- gfgcd(f,g,p) : untuk mencari faktor persekutuan terbesar dari dua polinomial f dan g atas $\mathbb{Z}_p$.
- doubfac(f,p) : untuk Mendeteksi Polinomial yang faktorisasinya hanya satu polinomial square-free dan pergandaannya.

Berikut ini beberapa fungsi matlab yang lain yang telah implementasikan.

1. Fungsi Matlab untuk Mendeteksi Faktorisasi *Square-free* dari Polinomial atas $\mathbb{Z}_p$

```
function [Fs]=sf(f,p) %f polinomial monik
S=[];
R=[];
d=length(f)-1;
g=mod(polyder(f),p)%b
syms x
pf=vektorpol(fliplr(f));
if g==0
    pfp=subs(pf,x,x^(1/p));
    kfp=sym2poly(pfp) ;
    Fs=[S; pfp,((length(f)-1)/(length(kfp)-
1))];
else
    k=gfgcd(f,g,p)%k
    k=mod(gfinv(k(length(k)),p)*k,p);
    [h,s]=gfdeconv(fliplr(f),k,p);%k
    lh=length(h)-1;
    if rem(d,lh)==0
        ph=mod(vektorpol(h)^(d/lh),p);
        kph=sym2poly(ph);%b
        if ph==pf
        [Fs]=[S;vektorpol(h),(d/lh)];
        end
    else
        w=fliplr(h);
        w1=gfinv(w(1),p);
        w=mod(w1*w,p);
        %menidentifikasi semua faktor w
        dw=mod(polyder(w),p)%b
        gw=gfgcd(w,dw,p);
        i=1;
        while length(w)~=1
            y=gfgcd(fliplr(k),w,p);
           [fac,s2]=gfdeconv(fliplr(w),y,p);
            f1=gfinv(fac(length(fac)),p);
            fac=mod(f1*vektorpol(fac),p);
            if fac==1
               fac=[];
               S=[S;fac];
               R=[R (fac)^i];
            else
                S=[S;fac,i];
                R=[R (fac)^i];
            end
           w=fliplr(y);
           [k,s3]=gfdeconv(k,y,p);
           k=mod(gfinv(k(length(k)),p)*k,p);
           if k==1
           rf=mod(expand(prod(R)),p);
           rf=sym2poly(rf);
           k=gfdeconv(fliplr(f),fliplr(rf),p);
                [T]=doubfac(fliplr(k),p);
                S=[S;T];
            break
            else
            i=i+1;
            end
    end
syms x
k=fliplr(k);
dk=mod(polyder(k),p);
pk=vektorpol(fliplr(k));
if dk==0
fk=subs(pk,x,x^(1/p));
vfk=sym2poly(fk);
Fk=[fk^((length(k)-1)/(length(vfk)-1))];
Fs=[S; fk,((length(k)-1)/(length(vfk)-1))];
else
```

```
    Fs=S;
    end
    end
end
```

2. Fungsi Matlab untuk menentukan Matriks Berlekamp dari Polinomial atas $\mathbb{Z}_p$

```
% f koefisien polinomial dan p bilangan prima
function Bf=matriksberl(f,p)
f=gfinv(f(1),p)*f;
f=mod(f,p);
d=length(f)-1;
 t=p*d+1;
Bf=[];
i=1;
while i<=d
h=zeros(1,t);
h(p*((d-i)+1)+1)=1;
[r,s]=deconv(h,f);
Bf=[Bf;s];
i=i+1;
end
Bf=mod(Bf,p);
Bf=Bf(:,t-d+1:t);
Bf=fliplr(Bf);
```

3. Fungsi Matlab untuk mendeteksi Faktor Persekutuan Terbesar dari Polinomial dan Polinomial yang dibentuk dari Ruang *Null* Matriks Berlekamp

```
function [b,Fc]=fbs(f,p) %masukkan koef
polinomial dari pangkat tertinggi
b=f(1);
c=gfinv(b,p); %koefisien dari f
fo=mod(c*f,p); % bentuk monik
% menghitung matriks berlekamp
Bf=matriksberl(fo,p);
Bfi=mod(Bf-eye(size(Bf)),p);
Bfi=Bfi';
% ruang null dari (Bf-I)^T
Z = nullgf(Bfi,p);
Z=Z';
[u,v]=size(Z);
H=fliplr(Z(2:u,:));
Fc=[];
Gc=[];
for i=1:u-1
       j=0;
      while j<p
          hj=H(i,:)-[zeros(1,v-1) j];
           hj=mod(hj,p);
           gc=gfgcd(fo,hj,p);
           if length(gc)==1
               G=[];
               Gc=[Gc,G];
           else
           G=vektorpol(gc);
           if gc(length(gc))==0
               G=G;
           else
           G=mod(gfinv(gc(length(gc)),p)*G,p);
           end
           Gc=[Gc,G];
           end
           Fc=union(Fc,Gc);
        j=j+1;
      end
    end
    if length(Fc)>1
    for k=1:length(Fc)
        kfc1=sym2poly(Fc(k));
        r=k+1;
        while r<=length(Fc)
            kfc2=sym2poly(Fc(r));
            gkf=gfgcd(kfc1,kfc2,p);
            if length(gkf)==1
                Hk=[];
           Fc=[Hk Fc];
            else
                if length(kfc1)>length(kfc2)
              hg=gfdeconv(fliplr(kfc1),gkf,p);
                    Hk=vektorpol(hg);
                    Fc=[Hk Fc];
                else

hg=gfdeconv(fliplr(kfc2),gkf,p);
                    Hk=vektorpol(hg);
                    Fc=union(Hk,Fc);
                end
            end
          r=r+1;
        end
    end
    elseif length(Fc)==1
th=gfdeconv(fliplr(f),fliplr(sym2poly(Fc)),p);
        th=vektorpol(th);
        Fc=union(th,Fc);
    else
        Fc=[];
    end
```

4. Fungsi Matlab untuk mendeteksi faktorisasi tak tereduksi Algoritma Berlekamp

```
function [kf,If]=berlekamploop(f,p) %masukkan
polinomial dari pangkat tertinggi
kf=f(1);
[kf,Fc]=fbs(f,p);
if length(Fc)==0
    f=fliplr(f);
    If=vektorpol(f);
else
d=length(f)-1;
If=[];
for i=1:d^p
    f1=sym2poly(Fc(1));
    [b1,fc]=fbs(f1,p);
    fc=fc;
    if length(fc)==0
        If=union(If,Fc(1));
        Fc=setdiff(Fc,Fc(1));
    else
        Fc=setdiff(Fc,Fc(1));
        Fc=union(fc,Fc);
    end
    if length(Fc)==0
            break
     else
            continue
    end
end
end
```

5. Fungsi Matlab untuk Algoritma 3

```
function [kf,Fac]=gabungan(f,p)
Fac=[];
kf=f(1);
```

```
d=length(f)-1;
c=gfinv(kf,p); %koefisien dari f
f=mod(c*f,p);
Fs=sf(f,p);
[m,n]=size(Fs);
i=1;
while i<=m
    fs=sym2poly(Fs(i,1));
    V=ddfp(fs,p);
    vf=sym2poly(V(1));
    dvf=doufac1(vf,p);
    fv=sym2poly(dvf(1));
    vf=sffmy(fv,p);
    vf=sym2poly(vf(1));
    [b1,Vf]=berlekamploop(vf,p);
    if length(Vf)==1
    Fac=[Fac;Vf(1),V(1,2)*Fs(i,2)*dvf(1,2)];
    else
     k=length(Vf);
     for j=1:k
     Fac=[Fac;Vf(j),V(1,2)*Fs(i,2)*dvf(1,2)];
     end
    end
    i=i+1;
end
```

Adapun untuk fungsi yang dijalankan untuk menyelesaikan persoalan adalah fungsi Matlab gabungan(f,p). Untuk menjalankan fungsi-fungsi tersebut dengan Matlab setiap fungsi disimpan dengan nama {nama_fungsi}.m dalam satu folder. Berikut hasil *running* program yang ditampilkan pada jendela *command window* dari Contoh 1.

```
f=[ 2     1     0     1     3     2     0
2     3     2     2     0     1     4];
p=5;
[kf,Fac]=gabungan(f,p)
Fs =
[ x^5 + 2*x^4 + 4*x^3 + 2*x^2 + 4*x + 1, 2]
[                                x + 3, 3]
Bf =
     1     0     0     0     0
     4     1     3     1     3
     3     4     3     2     4
     1     2     0     0     1
     0     2     3     1     0
Z =
     1     0
     0     2
     0     2
     0     1
     0     0
kf =
     2
Fac =
[              x^2 + 2, 2]
[ x^3 + 2*x^2 + 2*x + 3, 2]
[                x + 3, 3]
```

Berdasarkan hasil keluaran *running* program diperoleh hasil akhir himpunan pasangan terurut Fac dengan tiga anggota. Untuk setiap pasangan terurut tersebut adalah urutan pertama merupakan faktorisasi polinomial tak tereduksi dan urutan kedua adalah penggandaan polinomial tersebut untuk setiap pasangan terurut. Hasil ini sama dengan solusi contoh 1.

E. SIMPULAN DAN SARAN

Algoritma Berlekamp merupakan salah satu metode untuk menfaktorisasi polinomial atas lapangan $\mathbb{Z}_p$, namun hanya dibatasi untuk polinomial *square-free*. Dengan menggabungkan Algoritma Berlekamp dan Algoritma Faktorisasi *square-free* diperoleh suatu algoritma baru untuk faktorisasi polinomial *square-free* dan bukan *square-free* atas $\mathbb{Z}_p$. Untuk penelitian lanjutan, peneliti menyarankan untuk faktorisasi polinomial *square-free* dan bukan *square-free* atas $\mathbb{Z}_{p^k}$ dengan $k \in \mathbb{Z}$.

UCAPAN TERIMA KASIH

Penelitian ini didanai oleh Direktorat Riset dan Pengabdian Masyarakat (DPRM) Kementerian Riset Teknologi dan Pendidikan Tinggi melalui Program Penelitian Dosen Pemula tahun pendanaan 2019. Terima kasih juga untuk LPPM Universitas Quality Berastagi yang telah memfasilitasi peneliti hingga penelitian ini selesai.

REFERENSI

Abdel-Ghaffar, K. (2012). Counting Matrices over Finite Field Having a Given Number of Rows of Unit Weight. *Linear Algebra and its Applications, 436*, 2665-2669.

Aransay, J., & Divasón, J. (2016). Formalisation of the computation of the echelon form of a matrix in Isabelle/HOL. *Formal Aspects of Computing, 28*, 1005–1026.

Batoul, A., Guenda, K., & Gulliver, T. A. (2015). Repeated-Root Isodual Cyclic Codes over Finite Fields. *Springer International Publishing Switzerland 2015*, 119–132.

Butar-butar, J. L. (2018). Menentukan Grup Galois Pada Polinomial Rasional Dengan Pengembangan Metode Stauduhar. *Jurnal Curere, 2*(2), 184-193.

Cao, Y., & Gao, Y. (2014). Repeated root cyclic F_q-linear codes over F_(q^l). *Journal Finite Fields and Their Applications, 31*, 202–227.

Childs, L. N. (2009). *A Concrete Introduction to Higher Algebra Third edition.* New York: Springer.

Couveignes, J.-M., & Lercier, R. (2013). Fast Construction of Irreducible Polynomials over Finite Fields. *Israel Journal of Mathematics, 194*, 77–105.

Ding, C. (2017). A sequence construction of cyclic codes over finite fields. *Cryptography and Communications, 10*(2), 319–341.

Divasón, J., Joosten, S., Thiemann, R., & Yamada, A. (2019). A Verified Implementation of the Berlekamp–

Zassenhaus Factorization Algorithm. *Journal of Automated Reasoning, 63*, 1-37. doi:https://doi.org/10.1007/s10817-019-09526-y

Divasón, J., Joosten, S., Thiemann, R., & Yamada, A. (2017). A Formalization of the Berlekamp–Zassenhaus Factorization Algorithm. *CPP 2017 Proceeding of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, (pp. 17-29).

Grout, J. (2010). The minimum rank problem over finite fields. *Electronic Journal of Linear Algebra, 20*, 691-716.

Hanif, S., & Imran, M. (2011). *Factorization Algorithms for Polynomials over Finite Fields.* Linnæus University, Departement of Mathematics. Retrieved April 19, 2019, from http://www.diva-portal.org/smash/get/diva2:414578/FULLTEXT01.pdf

Hanif, Sajid. (2011). *Factorization Algorithms for Polynomials over Finite Fields.* Master Thesis, Departement of Mathematics.

Iliev, A., & Kyurkchiev, N. (2018). A Note on Euclidean and Extended Euclidean Algorithms for Greatest Common Divisor for Polynomials. *International Journal of Pure and Applied Mathematics, 118*(3), 713-721.

Irwan, M. (2017). Pengantar Matlab Untuk Sistem Persamaan Linear. *Jurnal MSA, 5*(2), 48-53.

Lee, M. M.-D. (2013). *Factorization of multivariate polynomials.* Dissertation, Technischen Universitat Kaiserslautern, Mathematics, Kaiserslautern. Retrieved July 28, 2019, from https://d-nb.info/1036637972/34

Mursyidah, H. (2017). Algoritma Polinomial Minimum untuk Membentuk Matriks Diagonal dari Matriks Persegi. *Aksioma Jurnal Pendidikan Matematika FKIP Univ. Muhammadiyah Metro, 6*(2), 282-293.

Oktafia, R., Gemawati , & Endang. (2014). Faktorisasi Polinomial Aljabar dengan Menggunakan Metode Euclidean dan Faktor Persekutuan Terbesar. *Jurnal Online Mahasiswa FMIPA UNRI*, 1-5.

Saropah. (2012). Akar-Akar Polinomial Separable Sebagai Pembentuk Perluasan Normal Pada Ring Modulo. *Jurnal Cauchy, 2*(3), 148-153.

Syaharuddin, & Mandailina, V. (2017). Pengembangan Modul Pemrograman Komputer Berbasis Matlab. *Jurnal Teori dan Aplikasi Matematika, 1*(1), 1-4.

Temnhurne, J., & Sathe, S. R. (2016). New Modified Euclidean and Binary Greatest Common Divisor Algorithm., 62(6), 852-858. *IETE Journal of Research, 62*(6), 852-858.

JTAMV10N2-9

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		