Analysis of The Extended Euclidean Algorithm & Galois Fields: A Matlab Approach

Nikeel Ramharakh (2433669)
13^{th} May 2024

Data and Information Management (ELEN3015A)

I. Section 1: Outcomes of Laboratory 2

1) The **`[g,a,b] = Extended_Euclidean_Int`** function implements the Extended Euclidean Algorithm to compute the greatest common divisor (gcd) of two integers, v and u in the equation $g = av + bu$. It initializes coefficients a and b as 1 and 0 respectively, along with supplementary coefficients $a2$ and $b2$ as 0 and 1. Through a while loop that continues until u becomes 0, it updates v and u based on their remainders in an iterative manner and recalculates coefficients a and b through the use of temporary variables $atemp$ and $btemp$. These calculations all follow the logic of the Table Method to solve problems using the Extended Euclidean Algorithm. Finally, it returns the gcd, which is the last non-zero remainder once the while loop condition is met. This algorithm is efficient for finding the gcd of two integers and simultaneously determining coefficients a and b such that $gcd(v, u) = av + bu$. The robustness of the algorithm was tested against the inbuilt Matlab **`[g,a,b] = gcd(v,u)`** function. A flow diagram of the algorithm is found below in Fig. 1.

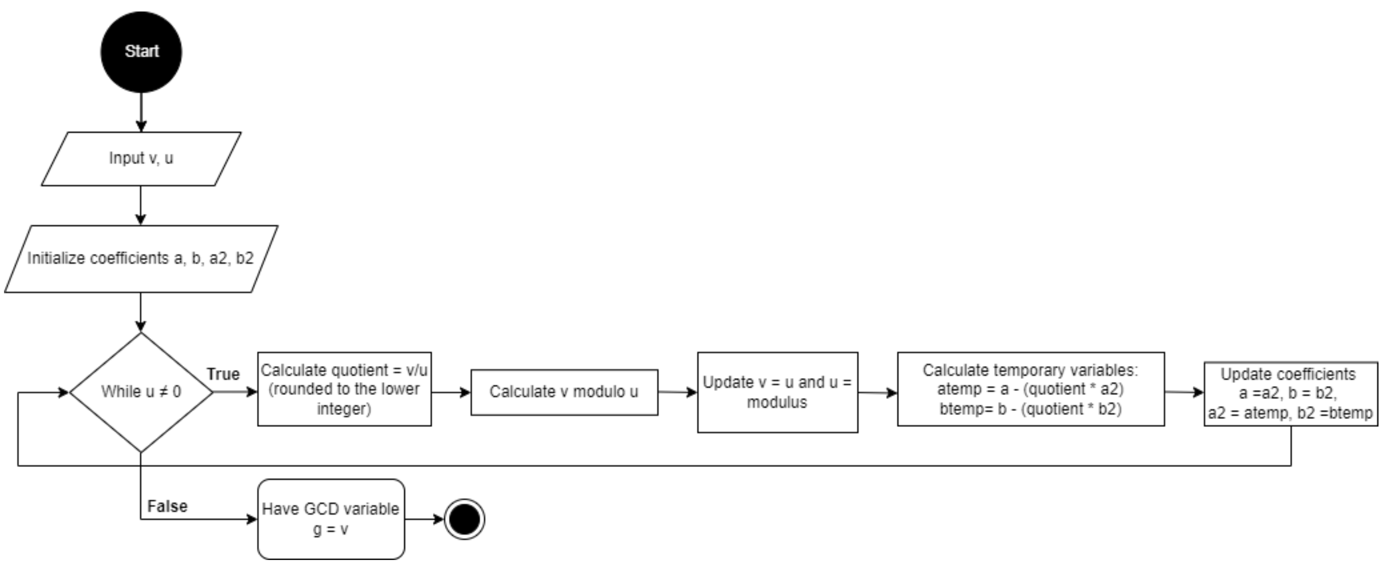

Fig. 1: Flow Diagram Depicting The Extended Euclidean Algorithm Employed

A handwritten attempt was made to further verify the algorithm used beyond comparing to the inbuilt Matlab function with u and v integer values being 18 and 48 respectively.

$$\begin{aligned} \text{Given:} \quad & g = av + bu \\ \& \quad & u = 18, \quad v = 48 \\ & \mathrm{mod}(48, 18) = 12 \\ & \mathrm{mod}(18, 12) = 6 \\ \therefore \quad & \mathrm{gcd}(v, u) = 6 \end{aligned}$$

Utilizing this gcd to determine coefficients a and b can be done by using the Extended Euclidean Algorithm as follows:

$$\begin{aligned}
\text{Given:} \quad 6 &= a(48) + b(18) \\
48 &= 2(18) + (12) \\
18 &= 1(12) + (6) \\
\\
6 &= (18) - 1(12) \\
6 &= (18) - (48 - 2(18)) \\
6 &= -1(48) + 3(18) \\
\therefore \quad a &= -1, \quad b = 3
\end{aligned}$$

Testing of the robustness of the algorithm was undertaken to verify the efficacy and reliability of the results. TABLE I compares the results obtained when using the inbuilt Matlab function and the **`[g,a,b] = Extended_Euclidean_Int`**.

TABLE I: Rigorous Testing of the Employed Algorithm

(v, u)	`[g,a,b] = gcd(v,u)`	`[g,a,b] = Extended_Euclidean_Int`
$(48, 18)$	$(6, -1, 3)$	$(6, -1, 3)$
$(5, 2)$	$(1, 1, -2)$	$(1, 1, -2)$
$(3, 10)$	$(1, -3, 1)$	$(1, -3, 1)$
$(14, 28)$	$(14, 1, 0)$	$(14, 1, 0)$
$(15, 35)$	$(5, -2, 1)$	$(5, -2, 1)$
$(21, 42)$	$(21, 1, 0)$	$(21, 1, 0)$
$(25, 50)$	$(25, 1, 0)$	$(25, 1, 0)$
$(27, 63)$	$(9, -2, 1)$	$(9, -2, 1)$
$(30, 72)$	$(6, 5, -2)$	$(6, 5, -2)$
$(33, 81)$	$(3, 5, -2)$	$(3, 5, -2)$
$(9, 17)$	$(1, 2, -1)$	$(1, 2, -1)$

2) The construction of TABLE II is summarized as follows assuming equivalence between α and x:

TABLE II: Depiction of a $GF(2^4)$ field by the primitive polynomial $p(x) = x^4 + x + 1$

Codeword	Polynomial in x	Power of α
0000	0	-
1000	1	1
0100	x	α
0010	x^2	α^2
0001	x^3	α^3
1100	$1 + x$	α^4
0110	$x + x^2$	α^5
0011	$x^2 + x^3$	α^6
1101	$1 + x + x^3$	α^7
1010	$1 + x^2$	α^8
0101	$x + x^3$	α^9
1110	$1 + x + x^2$	α^{10}
0111	$x + x^2 + x^3$	α^{11}
1111	$1 + x + x^2 + x^3$	α^{12}
1011	$1 + x^2 + x^3$	α^{13}
1001	$1 + x^3$	α^{14}

$$\begin{array}{llll}
x^5 = x^4 \cdot x & x^6 = x^5 \cdot x & x^7 = x^6 \cdot x & x^8 = x^7 \cdot x \\
x^5 = (x + 1) \cdot x & x^6 = (x^2 + x) \cdot x & x^7 = (x^3 + x^2) \cdot x & x^8 = (x^3 + x + 1) \cdot x \\
x^5 = x^2 + x & x^6 = x^3 + x^2 & x^7 = x^3 + x + 1 & x^8 = x^2 + 1
\end{array}$$

$$
\begin{aligned}
x^9 &= x^8 \cdot x & x^{10} &= x^9 \cdot x & x^{11} &= x^{10} \cdot x & x^{12} &= x^{11} \cdot x \\
x^9 &= (x^2+1) \cdot x & x^{10} &= (x^3+x) \cdot x & x^{11} &= (x^2+x+1) \cdot x & x^{12} &= (x^3+x^2+x) \cdot x \\
x^9 &= x^3+x & x^{10} &= x^2+x+1 & x^{11} &= x^3+x^2+x & x^{12} &= x^3+x^2+x+1
\end{aligned}
$$

$$
\begin{aligned}
x^{13} &= x^{12} \cdot x & x^{14} &= x^{13} \cdot x \\
x^{13} &= (x^3+x^2+x+1) \cdot x & x^{14} &= (x^3+x^2+1) \cdot x \\
x^{13} &= x^3+x^2+1 & x^{14} &= x^3+1
\end{aligned}
$$

- $\alpha^4 + \alpha^9$
  Since addition of alpha terms can be computed by XORing the Codewords:

$$
\begin{array}{cccc}
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
\hline
1 & 0 & 0 & 1 \\
\hline
\end{array}
$$

$$\therefore \alpha^4 + \alpha^9 = \alpha^{14}$$

- $\alpha^4 \cdot \alpha^9$
  Here, two forms of multiplication are performed to reinforce the validity of results.

$$
\begin{aligned}
\alpha^4 \cdot \alpha^9 &= \alpha^{(4+9)} & \qquad (1+x) \cdot (x+x^3) \\
&= \alpha^{13} & = x^3+x^2+1
\end{aligned}
$$

$$\therefore \alpha^4 \cdot \alpha^9 = \alpha^{13}$$

- α^4/α^9
  When performing the division α^4/α^9 in a Galois field of a $GF(2^4)$ modulo arithmetic is involved due to the cyclic nature of the field. Since there are fifteen distinct alpha terms in the field, any exponential operation greater than fifteen can be reduced by modulo fifteen to bring it within the range of the field.

  α^4/α^9 can be simplified as $\alpha^{(4-9) \mod 15}$. Since, $4 - 9 = -5$ and $-5 \mod 15$ is equals 10. Therefore, α^4/α^9 simplifies to α^{10}.

$$\therefore \alpha^4/\alpha^9 = \alpha^{10}$$

  All results were verified against the Matlab commands corresponding to each of the above calculations. The Matlab results are given as powers of α and matches the results of the above calculations.

3) Polynomial multiplication $P(x) \cdot Q(x)$ is illustrated as follows:

$$
\begin{aligned}
P(x) &= 1 + \alpha^3 x^2 + \alpha^3 x^6 + \alpha^7 x^7 \\
Q(x) &= \alpha^4 x^2 + \alpha^{10} x^6 \\
P(x) \cdot Q(x) &= (1 + \alpha^3 x^2 + \alpha^3 x^6 + \alpha^7 x^7) \cdot (\alpha^4 x^2 + \alpha^{10} x^6) \\
&= \alpha^4 x^2 + \alpha^{10} x^6 + \alpha^7 x^4 + \alpha^{13} x^8 + \alpha^7 x^8 + \alpha^{13} x^{12} + \alpha^{11} x^9 + \alpha^{17} x^{13} \\
&= \alpha^4 x^2 + \alpha^{10} x^6 + \alpha^7 x^4 + x^8(\alpha^{13} + \alpha^7) + \alpha^{13} x^{12} + \alpha^{11} x^9 + \alpha^{17} x^{13}
\end{aligned}
$$

$\alpha^{13} + \alpha^7$ is determined by using the XOR operation

$$
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
\hline
0 & 1 & 1 & 0 \\
\hline
\end{array}
$$

$$= \alpha^4 x^2 + \alpha^{10} x^6 + \alpha^7 x^4 + x^8 \alpha^5 + \alpha^{13} x^{12} + \alpha^{11} x^9 + \alpha^{17} x^{13}$$

Since $17 \mod 15 = 2$, the cyclic equivalence of α^{17} is α^2.
$$\therefore P(x) \cdot Q(x) = \alpha^4 x^2 + \alpha^{10} x^6 + \alpha^7 x^4 + x^8 \alpha^5 + \alpha^{13} x^{12} + \alpha^{11} x^9 + \alpha^2 x^{13}$$

This corresponds to the Matlab representation of [-Inf -Inf 4 -Inf 7 -Inf 10 -Inf 5 11 -Inf -Inf 13 2]

4) Appropriate Matlab commands used to calculate polynomial division of $(x^{15} - 1)/(x^4 + \alpha^{13} x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10})$ can be viewed with the attached .m-file. In order to test if the answer from the Matlab code is correct, a decision was made to multiply the answer by the divisor to determine if the same dividend can be yielded.

$$\begin{aligned}
Quotient &= \alpha^5 + \alpha^{13} x + \alpha^{11} x^2 + \alpha^6 x^3 + \alpha^{10} x^4 + \alpha^{13} x^5 + \alpha x^6 + \alpha x^7 + \alpha^{13} x^8 + x^9 \\
Divisor &= x^4 + \alpha^{13} x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10} \\
Quotient \cdot Divisor &= (\alpha^5 + \alpha^{13} x + \alpha^{11} x^2 + \alpha^6 x^3 + \alpha^{10} x^4 + \alpha^{13} x^5 +^6 +^7 + \alpha^{13} x^8 + x^9) \cdot \\
&\quad (x^4 + \alpha^{13} x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10}) \\
&= [0 - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf - Inf 0]
\end{aligned}$$

Utilizing the **gfconv(Quotient,Divisor,field)** function, the correct Dividend is obtained, being $(x^{15} - 1)$.

5) In order to perform the Extended Euclidean Algorithm which calculates the polynomials $g(x)$, $a(x)$ and $b(x)$ for polynomials $v(x)$ and $u(x)$ such that
$$g(x) = a(x)v(x) + b(x)u(x),$$
where $g(x)$ is the greatest common divisor of $v(x)$ and $u(x)$, it would require manipulation of the **[g,a,b] = Extended_Euclidean_Int** function. Firstly, the input variables v and u are no longer integers and rather are polynomials which can be represented in Matlab form. An example of this representation is using the polynomials of the dividend and divisor of Outcome 4, which appear as follows:

dividend - [0 -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf 0] which will be used as $v(x)$
divisor - [10 3 6 13 0 -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf] which will be used as $u(x)$

Once the $u(x)$, $v(x)$ and $field$ terms have been passed into the function, the structure of the function should follow that of the **[g,a,b] = Extended_Euclidean_Int** function with subtle differences. These subtle differences are due to the polynomial input variables giving rise for new challenges. Initialization of the a, b, $a2$, $b2$ base coefficients take place as before, however when dealing with computation of polynomials in Galois Fields, α powers are used in Matlab. Thus, where there was a 0 in the previous function, there shall be a $-Inf$ in the new function. Likewise, with a 1 in the previous function, there will be a 0 in the new function. The condition for the while loop follows this change. It is important to integrate the calculations used in the **[g,a,b] = Extended_Euclidean_Int** function, but utilizing polynomial arithmetic as in previous outcomes. In this case, the quotient and remainder are calculated with the use of the **[quotient,modulus] = gfdeconv(v,u,field)** function. Additionally, an if statement is used as a zero as the modulo should be a Galois Field element. Thus, the if statement converts a zero into $-Inf$ such that the while loop can now terminate. Since powers of α in binary are the basis of Galois Field computation, subtraction and addition should be dealt with the same. That is why the **gfadd()** function is used for the calculations for the $atemp$ and $btemp$ variables. The **gfconv()** function is also utilized in place of multiplication. The rest of the operations follow suit of the **[g,a,b] = Extended_Euclidean_Int** function, until the condition of the while loop is met, the code will then terminate with the g variable being the last non-zero polynomial.

TABLE III: Table Method Solving Extended Euclidean Algorithm for Polynomials

Quotient	v(x)	u(x)	Remainder	a	b	atemp	a2	b2	btemp
$Quotient$	$(x^{15}-1)$	$(x^4+\alpha^{13}x^3+\alpha^6x^2+\alpha^3x+\alpha^{10})$	-Inf	0	-Inf	0	-Inf	0	$Quotient$
-	$(x^4+\alpha^{13}x^3+\alpha^6x^2+\alpha^3x+\alpha^{10})$	-Inf	-	-Inf	0	-	0	$Quotient$	-

The strategy devised to verify the result of the **`[g,a,b] = Extended_Euclidean_GF(v,u,field)`** function will be use of the Table Method to solve for the polynomials with,

$$Quotient = \alpha^5+\alpha^{13}x+\alpha^{11}x^2+\alpha^6x^3+\alpha^{10}x^4+\alpha^{13}x^5+\alpha x^6+\alpha x^7+\alpha^{13}x^8+x^9$$

The Table Method ends when $u(x) = -Inf$ as division by zero is not permissible. It is thereby conclusive that the greatest common divisor $gcd(Dividend = v(x), Divisor = u(x))$ of two polynomials is the largest-degree polynomial dividing both. In this case, the second row of $v(x)$ is the $g(x)$. The $a(x)$ and $b(x)$ are $-Inf$ and 0 respectively as confirmed by the employed polynomial algorithm and Table Method.

Appendix

Appendix A - Outcome 1

```
function [g,a,b] = Extended_Euclidean_Int(v,u)
    % Initializaton of the base coefficients
    a = 1;
    b = 0;
    a2 = 0;
    b2 = 1;

    while u ~= 0
        % Calculations of quotient and remainder
        quotient = floor(v/u);
        modulus = mod(v,u);

        % Computation based on TABLE METHOD using new shifted variables
        atemp = a - (quotient*a2);
        btemp = b - (quotient*b2);

        % Shift in variables to account for the new row for the TABLE
        % METHOD
        a = a2;
        b = b2;
        a2 = atemp;
        b2 = btemp;
        v = u;
        u = modulus;
    end
    % Update the GCD to be the last non-zero remainder
    g = v;
end
```

Appendix

Appendix B - Outcomes 2-4

```
clear all
clc
%%
% Test data with given u and v integers
v = 48;
u = 18;
%% *Outcome 1 - Extended Euclidean Algorithm*
[g1,a1,b1] = Extended_Euclidean_Int(v,u);
[g2,a2,b2] = gcd(v,u);
disp('-------------------------------------------------------------------')
disp('Result of Outcome 1')
disp('ExtendedEuclideanInt function: (g,a,b)')
disp(['(',num2str(g1),',',num2str(a1),',',num2str(b1),')']);
disp('Inbuilt Matlab gcd function: (g,a,b)')
disp(['(',num2str(g2),',',num2str(a2),',',num2str(b2),')']);
disp('-------------------------------------------------------------------')
%% *Outcome 2*
% Defiing the GF(2^4) field
field = gftuple([-1:2^4-2]',4,2);

% Declaration of alpha^4 and alpha^9 variables
alpha4 = 4;
alpha9 = 9;

% alpha^4 + alpha^9
sum = gfadd(alpha4,alpha9,field);
disp('Result of Outcome 2')
disp('Answer for alpha^4 + alpha^9')
disp(['= ',num2str(sum)])

% alpha^4 * alpha^9
mulli = gfmul(alpha4,alpha9,field);
disp('Answer for alpha^4 * alpha^9')
disp(['= ',num2str(mulli)])

% alpha^4 / alpha^9
quot= gfdiv(alpha4,alpha9,field);
disp('Answer for alpha^4 / alpha^9')
disp(['= ',num2str(quot)])
disp('-------------------------------------------------------------------')
%% *Outcome 3*
% P(x) * Q(x)
% Declaration of P and Q polynomials
p = [0 -Inf 3 -Inf -Inf -Inf 3 7];
q = [-Inf -Inf 4 -Inf -Inf -Inf 10 -Inf];

% Computation
product = gfconv(p,q,field);
disp('Result of Outcome 3')
disp('Answer for P(x) * Q(x)')
disp(['= ','[',num2str(product),']'])
disp('-------------------------------------------------------------------')
%% *Outcome 4*
```

```
% Declaration of Dividend and Divisor
dividend = [0 -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf 0];
divisor = [10 3 6 13 0 -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf];

[quotient,remainder] = gfdeconv(dividend,divisor,field);

disp('Result of Outcome 4')
disp('(x^15 - 1)/(x^4 + alpha^13x^3 + alpha^6x^2 + alpha^3x + alpha^10)')
disp(['Quotient = ','[',num2str(quotient),']'])
disp(['Remainder = ','[',num2str(remainder),']'])
disp('-----------------------------------------------------------------')
%% *Outcome 5*
[g,a,b] = Extended_Euclidean_GF(dividend,divisor,field);

disp('Result of Outcome 5')
disp('Extended_Euclidean_GF function: (g,a,b)')
disp(['(','[',num2str(g),']',',',num2str(a),',',num2str(b),')']);
disp('-----------------------------------------------------------------')
%%
```

Appendix

Appendix C - Outcome 5

```
function [g,a,b] = Extended_Euclidean_GF(v,u,field)
    % Initializaton of the base coefficients using alpha power values due
    % to this using polynomials
    a = 0;
    b = -Inf;
    a2 = -Inf;
    b2 = 0;

    while u ~= -Inf
        % Calculations of quotient and remainder
        [quotient,modulus] = gfdeconv(v,u,field);

        % If statement used to account for a 0 remainder which in alpha terms is
            either -1 or -Inf
        if modulus == 0
           modulus = -Inf;
        end

        % Computation based on TABLE METHOD using new shifted variables.
        % Since subtraction is the same as addition in GF(2^4) gfadd is utilized
        multiplication_01 = gfconv(b,quotient,field);
        atemp = gfadd(a,multiplication_01,field);
        multiplication_02 = gfconv(b2,quotient,field);
        btemp = gfadd(a2,multiplication_02,field);

        % Shift in variables to account for the new row for the TABLE
        % METHOD
        a = b;
        b = atemp;
        a2 = b2;
        b2 = btemp;
        v = u;
        u = modulus;
    end
    % Update the GCD to be the last non-zero remainder
    g = v;
end
```