Catam 1.1 Finite Field Matrices

December 2022

1 Question 1

Our goal is to write a program to find the units of non-zero numbers modulo p. We know that any and all coprime numbers are units, and their inverse is unique.

A simple program has been written via a brute force algorithm, simply checking each possible inverse in turn. The program is called "UnitInverses.m" found on page 8. A few test cases have been tabluated below:

p	inverseArray
11	$[1, 6, 4, 3, 9, 2, 8, 7, 5, 10]$
5	$[1, 3, 2, 4]$
29	$[1, 15, 10, \ldots, 19, 14, 28]$

Table 1: Inverses modulo p

One simple way to reduce time complexity in half is to first note that if we know $a^{-1} = b$, we also know $b^{-1} = a$, and thus do not need to compute the values of b. Thus reducing the number of needed computations by half in a crude sense.

An amended program implementing this idea called "UnitInversesFast.m" can be found on page 8.

2 Question 2

Let us suppose that basic arithmetic operations and comparison of numbers has a time of one iteration of the computer CPU cycle. Also, let's exclude the final line as it is not part of the computations.

In order to execute any program of this form we must initialise the inverse array, p operations, which is negligible so may be ignored. To multiply a number with it's candidate inverse, take mod and compare to 1, this will clearly take 3 operations, finally to assign our inverse to the inverse array will take 1 operation. So in core of the nested loop we are bounded above by 4 operations, and bounded below by 3.

To find an upper bound on the number of clock cycles let us assume that we never break at any point in the nested loops, i.e. we must fully complete every loop before stopping. Clearly this is bounded above by $(p-1)*(p-1)*4 = 4(p-1)^2$ cycles. For a lower bound let us note that the final value of j must take all possible values of $1, 2, ...p-1$, thus we can create a lower bound of $(1+2+3...+p-1)*3$, that is summing over the number of operations needed for each possible for loop. This is $\frac{3}{2}p(p-1)$.

Finally we can truly say that our program has time complexity $O(p^2)$ as for large p we have:

$$\frac{1}{4} \leq \frac{p^2}{4(p-1)^2} \leq \frac{p^2}{f(p)} \leq \frac{2p^2}{3p(p-1)} \leq 1$$

3 Question 3

A program has been written to compute the row echelon form of a finite field matrix, it has the name "GaussianElimination.m" and may be found on page 9.

Let us quickly note that in row echelon form one basis of the row space is simply the non-zero rows, thus some computations yield:

Field	Initial Matrix	Row Echelon Matrix	Row Space Basis
F_{11}	$\begin{pmatrix} 0 & 1 & 7 & 2 & 10 \\ 8 & 0 & 2 & 5 & 1 \\ 2 & 1 & 2 & 5 & 5 \\ 7 & 4 & 5 & 3 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 3 & 0 & 0 \\ 0 & 1 & 7 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$(1,0,3,0,0)$ $(0,1,7,0,0)$ $(0,0,0,1,0)$ $(0,0,0,0,1)$
F_{23}	$\begin{pmatrix} 0 & 1 & 7 & 2 & 10 \\ 8 & 0 & 2 & 5 & 1 \\ 2 & 1 & 2 & 5 & 5 \\ 7 & 4 & 5 & 3 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 19 & 14 \\ 0 & 1 & 0 & 22 & 19 \\ 0 & 0 & 1 & 7 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$(1,0,0,19,14)$ $(0,1,0,22,19)$ $(0,0,1,7,2)$
F_5	$\begin{pmatrix} 6 & 16 & 11 & 14 & 19 & 4 \\ 7 & 9 & 1 & 1 & 21 & 0 \\ 8 & 2 & 9 & 12 & 17 & 7 \\ 2 & 19 & 2 & 19 & 19 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 3 & 3 & 0 \\ 0 & 1 & 0 & 3 & 3 & 2 \\ 0 & 0 & 1 & 3 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$(1,0,0,3,3,0)$ $(0,1,0,3,3,2)$ $(0,0,1,3,3,2)$

Table 2: Row Echelon Computations

We can very clearly see the ranks of these matrices are $4, 3, 3$ respectively.

4 Question 4

A progam has been written to find a basis of the kernels of given matrices, it is called "KernelBasis.m" and can be found on page 11.

The algorithm uses the non-trivial columns of the row echelon form, i.e. the columns that have not been intentionally normalised to a basis vector. It sets the component for said column in some new vector to 1, and then counters the effect by using the normalised columns to make all components zero. The vectors will be linearly independent as each new vector we compute has the lowest non-zero component in column vector form. To see that these vectors form a basis is simple. # columns = # non-zero rows ($Rank$) + $Null$. And the number of vectors we produce is exactly # columns - # non-zero rows.

Listed below are some computations:

Field	Initial Matrix	Row Echelon Matrix	Kernel Basis
F_2	$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 0 & 3 & 0 & 1 & 0 \\ 1 & 5 & 7 & 1 & 0 & 12 \\ 5 & 5 & 0 & 3 & 1 & 7 \\ 2 & 1 & 2 & 4 & 0 & 5 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\{ \begin{pmatrix} 0 \\ 0 \\ -1 \\ -1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 0 \\ 1 \end{pmatrix} \}$
F_{13}	$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 0 & 3 & 0 & 1 & 0 \\ 1 & 5 & 7 & 1 & 0 & 12 \\ 5 & 5 & 0 & 3 & 1 & 7 \\ 2 & 1 & 2 & 4 & 0 & 5 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 & 0 & 4 \\ 0 & 0 & 0 & 1 & 0 & 11 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$	$\{ \begin{pmatrix} 0 \\ -5 \\ -4 \\ -11 \\ -1 \\ 1 \end{pmatrix} \}$
F_{29}	$\begin{pmatrix} 3 & 7 & 19 & 3 & 9 & 6 \\ 10 & 2 & 20 & 15 & 3 & 0 \\ 14 & 1 & 3 & 14 & 11 & 3 \\ 26 & 1 & 21 & 6 & 3 & 5 \\ 0 & 1 & 3 & 19 & 0 & 3 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 22 & 2 \\ 0 & 1 & 0 & 0 & 13 & 3 \\ 0 & 0 & 1 & 0 & 10 & 9 \\ 0 & 0 & 0 & 1 & 13 & 23 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\{ \begin{pmatrix} -22 \\ -13 \\ -10 \\ -13 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -2 \\ -3 \\ -9 \\ -23 \\ 0 \\ 1 \end{pmatrix} \}$

Table 3: Basis Computations

5 Question 5

It is a known result from Linear Algebra that for a vector space V. $dim(V) = dim(U) + dim(U^\circ)$.

6 Question 6

We consider this question to be wholly taken modulo 19. Let us apply the "KernelBasis.m" from Question 4 to matrix A_1.

Initial Matrix	Row Echelon Matrix	U Basis	U° Basis
$\begin{pmatrix} 0 & 1 & 7 & 2 & 10 \\ 8 & 0 & 2 & 5 & 1 \\ 2 & 1 & 2 & 5 & 5 \\ 7 & 4 & 5 & 3 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 13 \\ 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$	$(1,0,0,0,13)$ $(0,1,0,0,6)$ $(0,0,1,0,3)$ $(0,0,0,1,1)$	$\begin{pmatrix} -13 \\ -6 \\ -3 \\ -1 \\ 1 \end{pmatrix}$

Table 4: Row Echelon Computations

It is now clear to see that the row space U has annihilator U° given by the basis of the kernel.

As we define the annihilator or a column space completely symmetrically to that of the row space, to find the annihilator of U° we may simply plug in the matrix of the transposes of the basis vectors of U° to our program, then take the transpose of the output to correct to row space.

Initial Matrix	Row Echelon Matrix	$(U^\circ)^\circ$ Basis	Row Echelon of $(U^\circ)^\circ$
(-13, -6, -3, -1, 1)	(1, 18, 9, 3, 16)	$(-18,1,0,0,0)$ $(-9,0,1,0,0)$ $(-3,0,0,1,0)$ $(-16,0,0,0,1)$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 13 \\ 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$

Table 5: Row Echelon Computations

Clearly a basis for $(U^\circ)^\circ$ is the basis of U. Hence $U = (U^\circ)^\circ$

7 Question 7

Our goal is, given two row spaces, U, W to find bases of $U \cap W$, U, W and $U + W$. The last three cases are quite simple, we use Gaussian elimination on each matrix, and a concatenation of the two matrices respectively to find a row space basis for U, W and $U + W$. The intersection is slightly more tricky but we can use $(U \cap W)^\circ = U^\circ + W^\circ$ giving $U \cap W = (U^\circ + W^\circ)^\circ$. Thus combining our knowledge from earlier questions a basis of $U \cap W$ will be the basis of the kernel of $U^\circ + W^\circ$, each of which we have programs to compute.

Also we can use a result from Linear algebra that $dim(U + W) = dim(U) + dim(W) - dim(U \cap W)$. We will see that this relationship holds in all cases we run through.

A program to compute these bases has been written called "MultipleBases.m" found on page 13.

For our three cases: Case 1) $p = 2$ and matrices are given. Case 2) $p = 19$, A is given and B is a matrix for the kernel of A. Naturally we take the transpose of the kernel matrix as we need W to be a row space, not a column space. Case 3) Same as Case 2 except $p = 7$

Case	Basis of U	Basis of W
1	(1, 0, 0, 0, 1, 0) (0, 1, 0, 1, 1, 0) (0, 0, 1, 0, 0, 0) (0, 0, 0, 0, 0, 1)	(1, 0, 0, 0, 0, 1) (0, 1, 0, 0, 0, 1) (0, 0, 1, 0, 1, 1) (0, 0, 0, 1, 1, 1)
2	(1, 0, 0, 0, 0, 0, 6, 1) (0, 1, 0, 0, 0, 3, 14) (0, 0, 1, 0, 0, 16, 9) (0, 0, 0, 1, 0, 0, 8) (0, 0, 0, 0, 1, 17, 6)	(1, 0, 9, 18, 6, 1, 12) (0, 1, 0, 2, 0, 4, 14)
3	(1, 0, 0, 0, 0, 0, 0, 0) (0, 1, 0, 0, 0, 3, 2) (0, 0, 1, 0, 0, 0, 0) (0, 0, 0, 1, 0, 2, 4) (0, 0, 0, 0, 1, 0, 0)	(0, 1, 0, 0, 0, 3, 2) (0, 0, 0, 1, 0, 2, 4)

Table 6: Bases of U, W

Case	Basis of $U + W$	Basis of $U \cap W$
1	(1, 0, 0, 0, 0, 0) (0, 1, 0, 0, 0, 0) (0, 0, 1, 0, 0, 0) (0, 0, 0, 1, 0, 0) (0, 0, 0, 0, 1, 0) (0, 0, 0, 0, 0, 1)	(0, 1, 0, 1, 1, 0) (0, 0, 1, 0, 1, 1)
2	(1, 0, 0, 0, 0, 0, 0) (0, 1, 0, 0, 0, 0, 0) (0, 0, 1, 0, 0, 0, 0) (0, 0, 0, 1, 0, 0, 0) (0, 0, 0, 0, 1, 0, 0) (0, 0, 0, 0, 0, 1, 0) (0, 0, 0, 0, 0, 0, 1)	$\vec{0}$
3	(1, 0, 0, 0, 0, 0, 0) (0, 1, 0, 0, 0, 3, 2) (0, 0, 1, 0, 0, 0, 0) (0, 0, 0, 1, 0, 2, 4) (0, 0, 0, 0, 1, 0, 0)	(0, 1, 0, 0, 0, 3, 2) (0, 0, 0, 1, 0, 2, 4)

Table 7: Bases of $U + W, U \cap W$

8 Question 8

It can be shown that in $\mathbb{R}$ the annihilator of a row space is orthogonal to the row space itself.

Let $v \in U^\circ$, then $Av = 0$, thus by considering a basis of the row space $B = \{b_1, ...b_n\}$ under the usual inner product $\forall i \ \ < b_i, v > = 0$. Since v is arbitrary $U \perp U^\circ$

Now this immediately gives that $U \cap W$, for cases $2, 3$, must be $\{\vec{0}\}$ as if it were not we can find a non-zero vector with norm 0. Thus case 3 is strange for someone working in $\mathbb{R}$ as we have this exact scenario with a non-trivial intersection.

9 Code

9.1 UnitInverses.m

```
p = 11;

% setup array
inverseArray = (1:p-1);


%core loop to find inverses
for i = 1:p-1
    for j = 1:p-1
        if (mod(i*j, p) == 1)
            inverseArray(i) = j;
            break;
        end
    end
end
```

9.2 UnitInversesFast.m

```
p = 11;

% setup array
inverseArray = zeros(1, p-1);

%core loop to find inverses
for i = 1:p-1
    if (inverseArray(i) ~= 0)
        continue
    end

    for j = 1:p-1
        if (mod(i*j, p) == 1)
            inverseArray(i) = j;
            inverseArray(j) = i;
            break;
        end
    end
end
```

9.3 GaussianElimination.m

```
function B = GaussianElimination(M, r, c, rows, columns)
    % r, c initial input are 1, rows, columns are dimensions of matrix
    if(r > rows || c > columns)
        B = M;
        return;
    end
    % Assuming in row echelon form for all rows less than r and columns
    % less than c. Turn next row/column into row echelon form

    % These values prechosen and precalculated for each choice of p
    inverseArray = [1, 4, 5, 2, 3, 6];
    p = 7;

    M = mod(M, p);

    zeroFlag = 1;
    if (M(r,c) == 0)
        % Find non zero entry in column
        for i = (r+1):(rows)
            if (M(i,c) ~= 0)
                for j = c:(columns)
                    M(r,j) = mod(M(r,j) + M(i,j), p);
                end
                zeroFlag = 0;
                break;
            end
        end
        if (zeroFlag == 1)
            B = GaussianElimination(M, r, c+1, rows, columns);
            return;
        end
    end


    % Basic Case
    M(r,c) = mod(M(r,c), p);
    if (M(r,c) == 0)
        inv = 0;
    else
        inv = inverseArray(M(r,c));
    end

    % Normalise top row
    for j = c:(columns)
```

```
        M(r,j) = mod(M(r,j) * inv, p);
    end

    % Subtract out
    for i = (r+1):(rows)
        a = M(i,c);
        for j = c:(columns)
            d = a * M(r, j);
            M(i,j) = mod(M(i,j) - d, p);
        end
    end

    % Complete row echelon below diagonal for the rest of matrix
    M = GaussianElimination(M, r+1, c+1, rows, columns);

    % Finally kill terms in rows above r in column c
    for i = (1):(r-1)
        a = M(i,c);
        for j = c:(columns)
            d = a * M(r, j);
            M(i,j) = mod(M(i,j) - d, p);
        end
    end
    B = M
end
```

9.4 KernelBasis.m

```
function S = KernelBasis(M, rows, columns)
    % First locate the special "1s"
    r = 1;
    rank = 0;

    Basis = [];

    diagOnes = [];
    for j = 1:columns
        if (M(r,j) == 1)
            rank = rank + 1;
            diagOnes(r) = j;
            r = r + 1;
            if (r > rows)
                break
            end
        end
    end

    vecNumber = 0;

    % Make sure there is a zero vector to be make other programs work
    Basis(1:columns, 1) = 0;

    for j = 1:columns
        if (ismember(j, diagOnes))
            continue;
        end
        vecNumber = vecNumber + 1;

        % Construct new basis vector
        vec(1:columns) = 0;
        vec(j) = 1;
        for i = 1:rows
            if (M(i,j) == 0)
                continue;
            end

            vec(diagOnes(i)) = -M(i,j);
        end

        for j = 1:columns
            Basis(j, vecNumber) = vec(j);
        end
```

```
    end


    disp(diagOnes);
    S = Basis;
end
```

The associated Program "Run1.m" was run to yield results

```
A = [3 7 19 3 9 6;
    10 2 20 15 3 0;
    14 1 3 14 11 3;
    26 1 21 6 3 5;
    0 1 3 19 0 3];

rows = 5;
columns = 6;

B = GaussianElimination(A, 1, 1, rows, columns);

C = KernelBasis(B, rows, columns);
```

9.5 MultipleBases.m

```
function [M1, M2, M3, M4] = MultipleBases (A, B, rowsA, rowsB, cols)
    M1 = GaussianElimination(A, 1, 1, rowsA, cols);
    M2 = GaussianElimination(B, 1, 1, rowsB, cols);

    C = vertcat(A, B);
    M3 = GaussianElimination(C, 1, 1, rowsA + rowsB, cols);

    D1 = KernelBasis(M1, rowsA, cols);
    D2 = KernelBasis(M2, rowsB, cols);

    E1 = transpose(D1);
    E2 = transpose(D2);

    F = vertcat(E1, E2);
    G = GaussianElimination(F, 1, 1, size(F,1), cols);
    H = transpose(KernelBasis(G, size(F,1), cols));
    M4 = GaussianElimination(H, 1, 1, size(H,1), size(H,2));
end
```

With associated program to run it with "Run2.m":

```
A = [1 0 0 0 3 0 0;
    0 5 0 1 6 3 0;
    0 0 5 0 2 0 0;
    2 4 0 0 0 5 1;
    4 3 0 0 6 2 6;];

C = GaussianElimination(A, 1, 1, 5, 7);

B = transpose(KernelBasis(C, 5, 7));

[M1, M2, M3, M4] = MultipleBases(A, B, size(A, 1), size(B, 1), size(A, 2));
```