**Core Project 1.1: Matrices over Finite Fields**

## Question 1

The function $findinverses(p)$, found on page 6, completes the task using the given algorithm. Note if p is not prime, it produces an error.

```
>> findinverses(7)
[1  4  5  2  3  6]

>> findinverses(13)
[1  7  9  10  8  11  2  5  3  4  6  12]

>> findinverses(4)
Please enter a prime
```

By exploiting $(x^{-1})^{-1} = x$ we can reduce run time. By starting with a blank array of length $p-1$, once we have calculated $x^{-1}$ we can set $inverses[x] = x^{-1}$ and $inverses[x^{-1}] = x$, and then skip calculating an inverse if the value is already filled. This will approximately half the time for large p.

## Question 2

In each "b" loop we are taking the residue of $ab$. One way of doing this is taking $ab - p\left\lfloor \frac{ab}{p} \right\rfloor$, a finite fixed number of operations. We repeat this $p-1$ times (maximum) per $b$ loop. We repeat the $b$ loop $p-1$ times in the $a$ loop. Hence we have a complexity of $p^2$ overall.

## Question 3

The function $ref(A, p)$, shown on page 7, takes a matrix A, and prime p, and returns the row echelon form (ref) of it. It uses subfunctions $S(A, k, q, i)$, $D(A, i, q, inverse)$ and $T(A, i, j)$ to perform row subtraction, division, and transposition respectively (see page 6). We have:

$$ref(A_1, 11) = \begin{bmatrix} 1 & 0 & 3 & 2 & 7 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} (mod\, 11) \qquad ref(A_1, 19) = \begin{bmatrix} 1 & 0 & 5 & 3 & 12 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 1 & 4 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} (mod\, 19)$$

$$ref(A_2, 23) = \begin{bmatrix} 1 & 18 & 21 & 10 & 4 & 16 \\ 0 & 1 & 4 & 0 & 15 & 10 \\ 0 & 0 & 1 & 9 & 14 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (mod\, 23)$$

Since row operations do not affect rank, we can deduce that the rank of each matrix is the same as the rank of the ref form. Since rank = row rank, the number of linearly independent rows is the rank. The leading $1's$ force linear independence, hence the number of non zero rows is the rank:

Calling the row vectors of ref(A1,11) $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \boldsymbol{v_4}$, and letting $a\boldsymbol{v_1} + b\boldsymbol{v_2} + c\boldsymbol{v_3} + d\boldsymbol{v_4} = \boldsymbol{0} \Rightarrow a = b = c = d = 0$ by looking at the coefficients of the first, second, and fourth and fifth entries. This method applies to any matrix in ref form.

Thus rank of $A_1$ is 4 over both GF(11) and GF(19). The rank of $A_2$ is 3 over GF(23). The non zero rows form a basis for the row space in each case, since they are linearly independent, and don't alter the row space of the original matrix.

## Question 4

The function $kernel(A, p)$, shown on page 8, takes a matrix A and prime p, and returns a basis for the kernel as the column vectors of the output matrix.

We first set $B = ref(A)$. A and B have the same kernel, since if $\boldsymbol{x} \in ker(A)$ then $\boldsymbol{v_i x} = 0$ where $\boldsymbol{v_i}$ are the row vectors of A. Hence, for any linear combination; $\sum(\lambda_i \boldsymbol{v_i})\boldsymbol{x} = 0$. All row operations do is rewrite the rows as linear combinations. Hence $ker(A) \subseteq ker(B)$ and similarly $ker(B) \subseteq ker(A)$, hence they are equal.

We calculate the $l$ values for each row, giving us the pivots, with the total number of them being the rank. By rank nullity, we calculate the nullity, and return the 0 vector if this is 0.

The pivot values determine which $x_i$ are "fixed", the rest being "free". Loop through all the fixed i, setting the vector $\boldsymbol{x^i}$ to be all 0's, but a 1 in the i'th position. In each case, solve for all fixed components (ie solve the system Bx=0) which has a unique solution, via $x_{l(i)} = \sum_{j=l(i)+1}^{columns} -B_{ij}x_j$ for all i (calculated with i decreasing). Once we know all the fixed components, we add the vector $\boldsymbol{x^i}$ to the kernel matrix and move onto the next free $x_i$.

This certainly gives us the same number of vectors in the kernel as the nullity. As long as they are linearly independent they span the kernel and we are done. This is guaranteed. Consider $\sum \lambda_j \boldsymbol{x^j} = \boldsymbol{0}$. In the i'th component, $(\boldsymbol{x^i})_i = 1$ and all other $(\boldsymbol{x^j})_i = 0$, hence $\lambda_i = 0 \ \forall i$. Hence linearly independent.

We have:

$$ref(B_1, 13) = \begin{bmatrix} 1 & 8 & 11 & 7 & 4 \\ 0 & 1 & 0 & 9 & 6 \\ 0 & 0 & 1 & 11 & 3 \\ 0 & 0 & 0 & 1 & 11 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} (mod\, 13) \qquad kernel(B_1, 13) = \begin{bmatrix} 7 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} (mod\, 13)$$

$$ref(B_1, 17) = \begin{bmatrix} 1 & 10 & 14 & 9 & 5 \\ 0 & 1 & 1 & 6 & 10 \\ 0 & 0 & 1 & 6 & 3 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} (mod\, 17) \qquad kernel(B_1, 17) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} (mod\, 17)$$

$$ref(B_2, 23) = \begin{bmatrix} 1 & 10 & 14 & 1 & 3 & 2 \\ 0 & 1 & 20 & 3 & 16 & 11 \\ 0 & 0 & 1 & 17 & 7 & 5 \\ 0 & 0 & 0 & 1 & 20 & 18 \\ 0 & 0 & 0 & 0 & 1 & 14 \end{bmatrix} (mod\, 23) \qquad kernel(B_2, 23) = \begin{bmatrix} 6 \\ 6 \\ 9 \\ 9 \\ 9 \\ 1 \end{bmatrix} (mod\, 23)$$

The columns of the output form a basis of the kernel (when non zero). Note the kernel of $B_1$ over GF(17) is trivial, and of rank 1 in the other 2 cases.

## Question 5

Given a row space U of $F^n$, let A be a matrix such that the set of rows of A form a basis of U. Then we have dim(U) = rank(A). We now use the rank-nullity theorem on A:

$$\begin{aligned} &rank(A) + nullity(A) = n \\ \Rightarrow\ &dim(U) + dim(U^\circ) = n \end{aligned}$$

## Question 6

Reading ahead, it seems like a good idea to write some functions to "annihilate" for us. We require two, one taking a row space input and giving a column space output, $artc(A, p)$, and vice versa, $actr(A, p)$, which can both be found on page 9. First, we find the row space of $A_1$, U (expressed as a matrix whose rows form a basis of the space). We can do this by:

$$U = ref(A_1, 19) = \begin{bmatrix} 1 & 0 & 5 & 3 & 12 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 1 & 4 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} (mod\, 19)$$

Next we calculate $U^\circ$, which is just the kernel. The function $artc(U, 19)$ simply takes this kernel.

$$U^\circ = artc(U, 19) = \begin{bmatrix} 6 \\ 13 \\ 16 \\ 18 \\ 1 \end{bmatrix} (mod\, 19)$$

We now verify that $(U^\circ)^\circ = U$. We use $actr(U^\circ, 19)$, which transposes, takes the kernel, then transposes again. Taking a transposition changes the problem from finding all row vectors $\boldsymbol{t}$ such that $\boldsymbol{ts} = 0$ for any column vector $\boldsymbol{s} \in U^\circ$ to finding all column vectors $\boldsymbol{t^T}$ such that $\boldsymbol{s^T t^T} = 0$ for any column vector $\boldsymbol{s^T}$ in the row space $(U^\circ)^T$. This is the same problem as the row case, hence we can now take the kernel, and transpose back, allowing us to work with familiar rows rather than columns.

$$(U^\circ)^\circ = actr(U^\circ, 19) = \begin{bmatrix} 3 & 0 & 0 & 0 & 1 \\ 16 & 0 & 0 & 1 & 0 \\ 10 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} (mod\, 19)$$

This doesn't look much like U, but we can perform row operations without changing the row space, so apply ref.

$$(U^\circ)^\circ = ref((U^\circ)^\circ, 19) = \begin{bmatrix} 1 & 0 & 0 & 0 & 13 \\ 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} (mod\, 19)$$

Now put U into this form, through more row operations.

$$U = \begin{bmatrix} 1 & 0 & 5 & 3 & 12 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 1 & 4 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} r_2 - 7r_3 \\ \longrightarrow \\ r_1 - 5r_3 \\ \longrightarrow \end{matrix} \begin{bmatrix} 1 & 0 & 0 & -17 & -23 \\ 0 & 1 & 0 & -26 & -39 \\ 0 & 0 & 1 & 4 & 7 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} r_2 + 26r_4 \\ \longrightarrow \\ r_1 + 17r_4 \\ \longrightarrow \end{matrix} \begin{matrix} r_3 - 4r_4 \\ \longrightarrow \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & -6 \\ 0 & 1 & 0 & 0 & -13 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Thus $U = (U^\circ)^\circ \pmod{19}$, as required.

## Question 7

The function $q7(A, B, p)$, found on page 9, takes matrices A and B and prime p as inputs, and returns the required bases, as the non zero rows of the outputs. The bases U and W are as before, and computed by simply using ref; $U = ref(A)$, $W = ref(B)$.

$U + W = \{u + w | u \in U, w \in W\}$ so is clearly spanned by the union of the bases of U and W. Hence we can compute a basis for U+W by appending the matrix W below U to create a new matrix C with the same number of columns but more rows. Then the non zero rows of ref(C) gives a basis.
Since we do not have an easy way of computing the intersection of bases, we exploit the second identity

$$(U \cap W)^\circ = U^\circ + W^\circ$$

We calculate $U^\circ$ and $W^\circ$, append matrices horizontally to get a new matrix D (as above, but now we have a column space hence horizontal), then apply *actr* to D and finally take the non zero rows of the ref form.

Running $q7(A, B, p)$ we get the below bases. The 0 rows have been omitted below.

Modulo 11 $A = A_1, B = B_1$:

$$U = \begin{bmatrix} 1 & 0 & 3 & 2 & 7 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad W = \begin{bmatrix} 1 & 7 & 4 & 6 & 9 \\ 0 & 1 & 3 & 4 & 0 \\ 0 & 0 & 1 & 4 & 10 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad U{+}W = \begin{bmatrix} 1 & 0 & 3 & 2 & 7 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 1 & 8 & 6 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad U{\cap}W = \begin{bmatrix} 1 & 9 & 0 & 7 & 0 \\ 0 & 1 & 7 & 9 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Modulo 19 $A = A_3, B = kernel(A_3)^T$:

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 4 & 5 & 12 & 0 \\ 0 & 0 & 1 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 11 & 13 \\ 0 & 0 & 0 & 0 & 1 & 17 & 6 \end{bmatrix} \quad W = \begin{bmatrix} 1 & 14 & 9 & 8 & 6 & 0 & 18 \\ 0 & 1 & 0 & 2 & 0 & 4 & 14 \end{bmatrix}$$

$$U + W = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 4 & 5 & 12 & 0 \\ 0 & 0 & 1 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 11 & 13 \\ 0 & 0 & 0 & 0 & 1 & 17 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad U \cap W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Modulo 23 $A = A_3, B = kernel(A_3)^T$:

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 14 & 15 & 19 & 0 \\ 0 & 0 & 1 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 14 & 16 \\ 0 & 0 & 0 & 0 & 1 & 5 & 8 \end{bmatrix} \quad W = \begin{bmatrix} 1 & 5 & 17 & 0 & 15 & 0 & 1 \\ 0 & 1 & 0 & 3 & 0 & 5 & 17 \end{bmatrix}$$

$$U + W = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 14 & 15 & 19 & 0 \\ 0 & 0 & 1 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 14 & 16 \\ 0 & 0 & 0 & 0 & 1 & 5 & 8 \\ 0 & 0 & 0 & 0 & 0 & 1 & 7 \end{bmatrix} \qquad U \cap W = \begin{bmatrix} 1 & 17 & 17 & 13 & 15 & 14 & 21 \end{bmatrix}$$

Note we have the relationship $dim(U+W) + dim(U \cap W) = dim(U) + dim(W)$. To see this, let $v_1, \ldots, v_a$ be a basis for $U \cap W$, $v_1, \ldots, v_a, u_1, \ldots, u_b$ be a basis for U, and $v_1, \ldots, v_a, w_1, \ldots, w_c$ be a basis for W. Then clearly $B = v_1, \ldots, v_a, u_1, \ldots, u_b, w_1, \ldots, w_c$ spans U+W. For linear independence, let

$$\alpha_1 v_1 + \ldots + \alpha_a v_a + \beta_1 u_1 + \ldots + \beta_b u_b + \gamma_1 w_1 + \ldots + \gamma_c w_c = 0$$

Then

$$x = \alpha_1 v_1 + \ldots + \alpha_a v_a + \beta_1 u_1 + \ldots + \beta_b u_b = -(\gamma_1 w_1 + \ldots + \gamma_c w_c).$$

Note the LHS $\in U$ and RHS $\in W$ hence $x \in U \cap W$. Hence

$$x = -(\gamma_1 w_1 + \ldots + \gamma_c w_c) = \delta_1 v_1 + \ldots + \delta_a v_a$$

so by linear independence of the set $v_1, \ldots, v_a, w_1, \ldots, w_c$ $\delta_i = 0$ and $\gamma_i = 0$. This implies

$$x = \alpha_1 v_1 + \ldots + \alpha_a v_a + \beta_1 u_1 + \ldots + \beta_b u_b = 0$$

so by linear independence of the set $v_1, \ldots, v_a$ $\alpha_i = 0$ and $\beta_i = 0$. Hence the whole set is linearly independent and the above relation holds.

## Question 8

In the modulo 19 case, we have $U + W$ full rank, with $U \cap W = \mathbf{0}$ ie $U + W = RowSpace(A_3) \oplus Ker(A_3)^T = GF(19)^7$. We can write the row space as the image of the transpose, so have $Im((A_3)^T) \oplus Ker(A_3)^T = GF(19)^7$.

Over $\mathbb{R}$ or $\mathbb{C}$ we have for any linear map $\alpha$, $Im(\alpha^*) = Ker(\alpha)^\perp \Leftrightarrow Im(A^\dagger) \oplus Ker(A)^\dagger = \mathbb{F}^n$, where * denotes the adjoint, $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$ and A is the matrix of the linear map $\alpha$ with respect to some orthonormal basis. Translating this to our language we get the (surprising) result that; for any row space U, $W = kernel(U)$ written as a row space;
i) The row space U+W is the whole space
ii) U and W have trivial intersection

Here in GF(p), however, the notion of an adjoint doesn't make sense since we cannot define an inner product, so we don't get this result generally, as seen in the modulo 23 case.

## Programs

### Division

(i) *findinverses.m*

```
function [inverse] = findinverses(p)

%initialise inverse
inverse=[];

%check if prime
if isprime(p)==0
    inverse='Please enter a prime';
    return
end

%loop to calculate the inverse
for a = 1:p-1
    for b=1:p-1
        if mod(a*b,p)==1
            inverse=[inverse b];
            break %out of the b loop
        end
    end
end
```

### Gaussian Elimination

(i) *T.m*

```
function [A] = T(A,i,j)
    %transposes rows i and j in a matrix A
    temp=A(i,:);
    A(i,:)=A(j,:);
    A(j,:)=temp;
end
```

(ii) *S.m*

```
function [A] = S(A,k,q,i)
    %Subtract from row k, q times row i
    A(k,:)=A(k,:)-q*A(i,:);
end
```

(iii) *D.m*

```
function [ A ] = D( A,i,q,inverse )
    %divides row i of A by q
    A(i,:)=A(i,:)*inverse(q);
end
```

(iv) $ref.m$

```
function [A] = ref( A,p )
%Algorithm: Start with input matrix A. Take i=j=1. If A_ij=0 try to
%find another row k in A with A_kj not 0. If this exists swap row 1 with
%row k. If not take i=2 and repeat...
%If no suitable pivot we are 0 matrix - done.
%Once we have a pivot we divide whole row by pivot.
%Then subtract multiples of this row from other rows to ensure whole column is 0's
%Increase i,j by 1 and repeat until j reaches number of columns or i
%reaches number of rows

rows=size(A,1);
columns=size(A,2);
i=1;
j=1;
inverse=findinverses(p);

while j<= columns

%find the pivot row and swap it in to place
    if mod(A(i,j),p)==0
        while j<=columns

            for k = i:rows
                if mod(A(k,j),p)~=0
                    A=T(A,k,i);
                    break % out of inner while loop - now have a pivot
                end
            end

            if mod(A(i,j),p)==0
                j=j+1; %keeps adding 1 to j until we find a non zero term
            else
                break
            end
        end

    %we are done once the inner while finishes and j increases past columns
        if j>columns
            A=mod(A,p);
            break
        end
    end

    A=mod(A,p);

%divide pivot row to get 1
    A=D(A,i,A(i,j),inverse);
    A=mod(A,p);
%subtract rows below
    for k= i+1:rows
        A=S(A,k,A(k,j),i);
    end

% we now add 1 to i and j and loop this until we reach the bottom right
% corner

    i=i+1;
    j=j+1;
```

```
    if j>columns | i>rows
        A=mod(A,p);
    break %out of the first while and hence end the program

    end
end
```

**Kernels and Annihilators**

(i) *calculatel.m*

```
function [l] = calculatel(M)
%determines locations of pivot elements in the ref matrix
for i = 1:size(M,1) %loop through rows
    for j = 1:size(M,2) %loop through columns
        if M(i,j)==1 %find the first 1 in the row
            l(i)=j;
            break %out of the for loop
        end
        if j==size(M,2) %if we get to the end of the column, and havn't found a 1
            l(i)=0;
        end
    end

end
```

(ii) *kernel.m*

```
function [ker] = kernel(A,p)
%The algorithm is described in the write up

[B] = ref(A,p);
l = calculatel(B); %members of l are the "fixed" components
rows=size(B,1);
columns=size(B,2);
%r=rank, n=nullity.
r=nnz(l);
%nnz counts non zero terms in the array l
n=columns-r;

if n==0
    ker=zeros(rows,1);
    return
end

d=0; %d counts how many lin indep elements sof kernel we have so far

for c=columns:-1:1 %loop through columns from bottom

    if ismember(c,l) %if fixed do nothing
    else %if component free
        d=d+1;
        x=zeros(columns,1); %initialise vector x
        x(c)=1; %set the free variable to 1

        for i=rows:-1:1 % solve the silmutaneous equations, starting from the bottom
```

```
            if l(i)==0 | l(i)==columns % do nothing
            % l(i)=0 means free component, we have already set them all to 0 by the algorithm
            % l(i)=columns means we have the equation x_i=0, so no
            % calculation needed

            else % for the fixed x_i which we must calculate

                for j=l(i)+1:columns % formula to calculate  them
                    x(l(i))=x(l(i))-B(i,j)*x(j);
                end
            end
        end

        ker(:,d)=x; %append vector x to end of kernel matrix.
    end

end
ker=mod(ker,p);
```

(iii) *artc.m*

```
function [B] = artc(A,p)
    %annihilate row to column
    %this one is simple; we just take the kernel of A
    B=kernel(A,p)
end
```

(iv) *actr.m*

```
function [ C ] = actr( A,p )
    %annihilate column to row
    %we first transpose so we can work with rows, then take the kernel, then
    %tranpose again
    B=transpose(A)
    B=kernel(B,p)
    C=transpose(B)
end
```

(v) *q7.m*

```
function [] = q7( A,B,p )

U=ref(A,p);
W=ref(B,p);

%for a basis of U+W we simply construct a new matrix of U on top of W, then
%ref it

C=[U;W];
UaddW=ref(C,p);

%for UintersectW we use the second given identity%
aU=artc(U,p);
aW=artc(W,p);

%now adding these 2 columns spaces requires adding horizontally

aUaddaW=[aU,aW];
```

```
%now we take the actr of this to obtain UintersectW

UintersectW=ref(actr(aUaddaW,p),p);

disp(['U has basis the (nonzero) rows of'])
disp(U)
disp(['W has basis the (nonzero) rows of'])
disp(W)
disp(['U+W has basis the (nonzero) rows of'])
disp(UaddW)
disp(['U intersect W has basis the (nonzero) rows of'])
disp(UintersectW)
end
```