1.1 Matrices over Finite Fields

Throughout this report p is assumed to be prime.

Question 1:

The printout of the program is attached at the end of the report. The program takes a prime number p as input and outputs and stores an array 'Inv' of length *p - 1*. A typical test output is shown below(for $p = 11$ case).

```
Inv =

     1     6     4     3     9     2     8     7     5    10
```

Figure 1, a typical output.

To speed up this procedure we can skip all values that are already other numbers' inverses when computing a number's inverse. This still gives the inverse of each number since there is a 1-1 correspondence between numbers and their inverses *mod* p but upper bounds the number of operations for number i by *p - i*. This modification speeds up the procedure by a factor of 2 because there are at most $\sum_{i=1}^{p-1} i = \frac{(p-2)(p-1)}{2}$ operations.

Question 2:

Note that for each number there are at most *p - 1* operations, and there are *p-1* numbers to test. Hence the complexity is p^2.

Question 3:

The printout of the program for Question 3 is attached at the end of the report. The output of the program is shown below.

```
A =

     1     0     3     2     7
     0     1     7     2    10
     0     0     0     1     8
     0     0     0     0     1
```

Figure 2, $p = 11$ case

```
A =

     1     0     5     3    12
     0     1     7     2    10
     0     0     1     4     7
     0     0     0     1     1
```

Figure 3, $p = 19$ case

```
A =

     1    18    21    10     4    16
     0     1     4     0    15    10
     0     0     1     9    14     7
     0     0     0     0     0     0
```

Figure 4, $p = 23$ case

Since row operations do not alter the row space of a matrix, we see that matrix A_1 has rank 4 when $p = 11$ or 19 and matrix A_2 has rank 3 when $p = 23$. The bases for row spaces of matrices A_1 and A_2 for different p can be read off from their row echelon forms:

$$B_{A_1,11} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 3 \\ 2 \\ 7 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 1 \\ 7 \\ 2 \\ 10 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 8 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}^T \right\} \quad B_{A_1,19} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 5 \\ 3 \\ 12 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 1 \\ 7 \\ 2 \\ 10 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 4 \\ 7 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}^T \right\}$$

$$B_{A_2,23} = \left\{ \begin{bmatrix} 1 \\ 18 \\ 21 \\ 10 \\ 4 \\ 16 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 1 \\ 4 \\ 0 \\ 15 \\ 10 \end{bmatrix}^T, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 9 \\ 14 \\ 7 \end{bmatrix}^T \right\}, \text{where } T \text{ denotes tranpose.}$$

Question 4:

The printout of the program for Question 4 is attached at the end of the report. The program takes a matrix and a prime number p as input and outputs a basis of the kernel of the matrix *mod* p. The outputs of the program for different examples are shown below:

7 2 1 2 1	0 0 0 0 0	6 6 9 9 9 1
Figure 5, $p = 13$ case	Figure 6, $p = 17$ case	Figure 7, $p = 23$ case

My program for this question works as follows:
1: Transform the matrix into row echelon form.
2: Eliminate all rows with zero entries.
3: From the result in step 2, use Matlab's symbolic expression feature to express all pivotal variables in terms of other variables inductively.
4: Assign specific values to non-pivotal variables to obtain a basis. (Let one variable = 1 and others = 0.)

Question 5:

The sum of the dimension of U and the dimension of U^o is equal to the number of columns of the matrix.

Question 6:

For U the row space of A_1 mod 19, U^o is the kernel of A_1. Use the program for question 4 we find it to be $\left\{ \begin{bmatrix} 6 & 13 & 16 & 18 & 1 \end{bmatrix}^T \right\}$.

The annihilator of U^o, $(U^o)^o$, is the set of all row vectors $\mathbf{t}$ such that $\mathbf{ts} = 0$, where $\mathbf{s}$ is the basis vector we just found. By taking the transpose of this relation we have $\mathbf{s}^T\mathbf{t}^T = 0$, i.e. $\mathbf{t}^T$ is in the kernel of the matrix $\begin{bmatrix} 6 & 13 & 16 & 18 & 1 \end{bmatrix}$.

Using the program for question 4 we find that $(U^o)^o$ has a basis

$$B_{(U^o)^o,19} = \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T, \begin{bmatrix} 10 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}^T, \begin{bmatrix} 16 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T, \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}^T \right\}$$

To show that A_1 spans the row space of $B_{(U^o)^o,19}$, we put these basis vectors and rows in the row echelon form of A_1 in one matrix and perform Gaussian elimination: (using program from question 1)

$$\begin{bmatrix} 1 & 0 & 5 & 3 & 12 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 1 & 4 & 7 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 10 & 0 & 1 & 0 & 0 \\ 16 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Gaussian Elimination}} \begin{bmatrix} 1 & 0 & 5 & 3 & 12 \\ 0 & 1 & 7 & 2 & 10 \\ 0 & 0 & 1 & 4 & 7 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

From the result of Gaussian elimination we can see that all basis vectors of $(U^o)^o$ are in the row space of A_1. Since $(U^o)^o$ and A_1 have the same dimension we conclude that they are equal.

Question 7:

The printout of the program for question 7 is attached at the end of the report. The program takes 2 matrices and a prime number p as inputs and outputs their row spaces, the intersection and the sum of their row spaces. (For input matrices P and Q, A is the row space of P, B is the row space of Q, C is the intersection of A and B, and D is the sum of A and B in the output.)

The inputs for different examples are:
Example 1: $P = A_1$, $Q = B_1$, $p = 11$.
Example 2: $P = A_3$, $Q = D^T$, $p = 19$, where D consists of basis vectors of the kernel of A_3.
Example 3: The same as in Example 2, except $p = 23$.

My program for this question works as follows:
1. Transform the input matrices into row echelon forms to obtain bases for their

row spaces.
2. Join the two matrices obtained in step 1 together to obtain a new matrix consisting of a spanning set of $U + W$, where U and W are row spaces of the input matrices.
3. Perform Gaussian elimination on the new matrix to obtain a basis for $U + W$
4. Find bases for the kernels of the two input matrices and transpose them. Apply step 1, 2, 3 to obtain a matrix H consisting of a transposed basis for $U^o + W^o$. The kernel of the matrix H is $(U^o + W^o)^o = U \cap W$, which can be found by using the program for question 4.

The outputs of the program for different examples are shown below:

```
A =

     1     0     3     2     7
     0     1     7     2    10
     0     0     0     1     8
     0     0     0     0     1


B =

     1     7     4     6     9
     0     1     3     4     0
     0     0     1     4    10
     0     0     0     0     1
     0     0     0     0     0


C =

     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
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0


D =

[ 7, 5, 1, 0, 0]
[ 8, 6, 0, 1, 0]
[ 0, 0, 0, 0, 1]
```

Figure 8, example 1

```
A =

     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


B =

     1    10     9     0     6     3     0
     0     1     0     2     0     4    14


C =

     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    14
     0     0     0     0     0     0     1


D =

     0     0     0     0     0     0     0
```

Figure 9, example 2

```
A =

     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


B =

     1     9    17    12    15    20     0
     0     1     0     3     0     5    17


C =

     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
     0     0     0     0     0     0     0


D =

[ 11, 3, 3, 5, 4, 16, 1]
```

Figure 10, example 3

The relation between the dimensions of U, W, $U + W$ and $U \cap W$ is $\dim(U) + \dim(W) - \dim(U \cap W) = \dim(U + W)$, where dim denotes dimension.

Question 8:

There exists a non-zero kernel vector that is a linear combination of basis vectors for row spaces.

Reference

[1] *Matlab Documentation*
URL https://www.mathworks.com/help/matlab/
[2] *Simon Wadsley, Linear Algebra*
URL https://www.dpmms.cam.ac.uk/~sjw47/LecturesM16.pdf
[3] *Computational Projects Assessors Committee, CATAM IB Manual 2018*

Source Code

Code for Question 1:

```
function [Inv]=InverseModP(p)
Inv = zeros(1,p-1);
for i = 1:p-1
for j = 1:p-1
if mod(i*j,p)==1
Inv(i)=j;
end
end
end
```

Code for Question 3:

```
function [F]=OneStep(B,p,Inv)% one step of gaussian elimination: reduce a submatrix such that the (1,1) entry is 1 and there is only 1 non-zero entry in the first column
d = size(B);
r = d(1);
c = d(2);
s=0;
E=B;
if B == zeros(r,c)
```

```
F=B;
else
for i=1:c
s=i;
if B(1:r,1)==zeros(r,1)
x = size(B);
B = B(1:r,2:(x(2)));
else
break
end
end
if B(1,1)==0
for k=2:r
if B(k,1) =0
z=B(k,:);B(k,:)=B(1,:);B(1,:)=z;
B(1,:)=mod(B(1,:)*Inv(B(1,1)),p);
break
end
end
else
B(1,:)=mod(B(1,:)*Inv(B(1,1)),p);
end
for j=2:r
B(j,:)=mod(B(j,:)-(B(j,1))*(B(1,:)),p);
end
E(1:r,s:c)=B;
F=E;
end
end
```

```
function [R]=GaussE(A,p)% apply OneStep function to submatrices of the original matrix
d = size(A);
r = d(1);
c = d(2);
t = InverseModP(p);
for i = 1:r
try %Use try to deal with empty matrix!
A(i:r,i:c)=OneStep(A(i:r,i:c),p,t);
R = A;
catch
break
end
end
```

Code for Question 4:

```
function [B]=KernelBasis(A,p)
R = GaussE(A,p);%Reduced matrix
d = size(R);
r = d(1);
c = d(2);
u = 0;
for i = 1:r
if R(i,1:c)==zeros(1,c)
u = u;
else
u = u + 1;
end
end
R = R(1:u,1:c);%ignore all 0 rows
d = size(R);
r = d(1);
c = d(2);
for i = 1:c
x(i)=sym(['x',num2str(i)]);
end
z = x;
for i = r:-1:1 % find algebraic expressions of non-pivotal vars.
for j = 1:c
if R(i,j)==0
continue
else
if j == c
x(c)=0;
else
s = 0;
for k = j+1:c
s = s - R(i,k)*x(k);
end
x(j) = s;
break
end
end
end
end
q = zeros(1,c);
for i = 1:r % storing indices of non-pivotal vars
for j = 1:c
if R(i,j)==0
continue
```

```
else
q(j) = 1;
break
end
end
end
for i = 1:c
for j = 1:c
C(i,j)=sym(['x',num2str(1),num2str(j)]);
end
end
D=C;
v = transpose(C(1,1:c));
if q == ones(1,c)
C = zeros(c,1);
else
for i = 1:c
if q(i) == 1
continue
else %q(i)==0
z = subs(x,x(i),1);%substitute
for j = 1:c
if (q(j) == 0) & (j  = i)
z = subs(z,z(j),0);
end
C(i,1:c)=z;
end
end
end
end
C = transpose(C);
if q == ones(1,c)
C = zeros(c,1);
else
for i=1:c
d = size(C);
f = d(2);
for j = 1:f
if C(1:c,j)==v
C(:,j)=[];
break
end
end
end
end
try
```

```
C(1,1)%test if C is an empty matrix and use 'try' to deal with the case where
C is empty
B=mod(C,p);
catch
B = zeros(c,1);
end
end%there is some redundancy in the last 40 lines but this program works for
all questions.
```

Code for Question 7:

```
function [A,B,C,D] = SumIntersection(Q,R,p)%A,B are row spaces of Q and R,
C = Q + R, D = Q intersection R
A = GaussE(Q,p);
B = GaussE(R,p);
c = size(A);
d = size(B);
G = zeros(c(1)+d(1),c(2));%Initialize a zero matrix
for i = 1:c(1)
G(i,1:c(2)) = A(i,1:c(2));
end
for i = 1:d(1)
G(c(1) + i,1:c(2)) = B(i,1:c(2));
end
C = GaussE(G,p);
clear c
clear d
N = KernelBasis(Q,p);
M = KernelBasis(R,p);
N = transpose(N);
M = transpose(M);
c = size(N);
d = size(M);
H = zeros(c(1)+d(1),c(2));
for i = 1:c(1)
H(i,1:c(2)) = N(i,1:c(2));
end
for i = 1:d(1)
H(c(1) + i,1:c(2)) = M(i,1:c(2));
end
S = KernelBasis(H,p);
D = transpose(S);
end
```