Principles of Linear Algebra With *Maple*™
Modular Arithmetic and Matrix Cryptography

Kenneth Shiskowski and Karl Frinkle

Contents

1 Modular Arithmetic and Matrix Cryptography **1**
1.1 Modular Arithmetic 1
1.2 Matrix Encryption 5

Chapter 1

Modular Arithmetic and Matrix Cryptography

1.1 Modular Arithmetic

In this section we will partition the set of integers $\mathbb{Z}$ into non-empty disjoint subsets $\mathbb{Z}_N = \{[0]_N, [1]_N, [2]_N, \ldots, [N-1]_N\}$ based on the remainder you get when you divide them all by a base positive integer $N \geq 2$. $\mathbb{Z}_N$ is called the set of mod N equivalence classes of $\mathbb{Z}$. Since every integer w can be written uniquely as $w = Q \cdot N + R$, for integer quotient Q and non-negative integer remainder R with $0 \leq R < N$, we can partition the integers $\mathbb{Z}$ by their remainder when they are divided by N so that $[0]_N$ is the set of all integers with 0 remainder for R (which is all multiples of N) while $[N-1]_N$ is the set of all integers with $N-1$ remainder for R. So $[R]_N = \{\, Q \cdot N + R \mid Q \in \mathbb{Z}\}$, for integers R with $0 \leq R \leq N-1$.

It should also be noted that for any integer w, we have $[w]_N = [R]_N$ if R is the remainder when you divide w by N. So

$$[0]_N = [N]_N = [2N]_N = \ldots$$
$$[1]_N = [N+1]_N = [2N+1]_N = \ldots$$

and similarly for the other elements of $\mathbb{Z}_N$. We say that $[w]_N$ (which is the integers equivalent to w mod N) is the equivalence class of $\mathbb{Z}$ generated by the integer w and it consists of all the integers with the same remainder R as w has when divided by N.

The amazing and immensely useful nature of $\mathbb{Z}_N$ lies in that we can do arithmetic with the elements of this set, that is, we can add, subtract and multiply their elements together to get another element of $\mathbb{Z}_N$, and when N is prime we can divide by the non-zero elements of $\mathbb{Z}_N$. This last fact makes mod-

ular arithmetic in $\mathbb{Z}_p$, for p a prime, the place to work for many cryptographic purposes as we will see shortly.

Example 1.1.1. (The Example of $\mathbb{Z}_5$ for Base $N = 5$.) Let's now do an example for base $N = 5$. Then the partition of $\mathbb{Z}$ mod 5 is the collection of subsets of $\mathbb{Z}$ given by $\mathbb{Z}_5 = \{[0]_5, [1]_5, [2]_5, [3]_5, [4]_5\}$ where $\mathbb{Z}_5$ is read $\mathbb{Z}$ mod 5, and

$$\begin{aligned}
[0]_5 &= \{\, Q \cdot 5 \mid Q \in \mathbb{Z}\} \\
[1]_5 &= \{\, Q \cdot 5 + 1 \mid Q \in \mathbb{Z}\} \\
[2]_5 &= \{\, Q \cdot 5 + 2 \mid Q \in \mathbb{Z}\} \\
[3]_5 &= \{\, Q \cdot 5 + 3 \mid Q \in \mathbb{Z}\} \\
[4]_5 &= \{\, Q \cdot 5 + 4 \mid Q \in \mathbb{Z}\}\,.
\end{aligned}$$

In particular, we have that

$$[3]_5 = \{\ldots, -12, -7, -2, 3, 8, 13, 18, \ldots\}$$

and so

$$\ldots = [-12]_5 = [-7]_5 = [-2]_5 = [3]_5 = [8]_5 = [13]_5 = [18]_5 = \ldots$$

Example 1.1.2. (Addition and Multiplication in Modular Arithmetic in $\mathbb{Z}_5$.) Let's do the addition and multiplication tables for arithmetic in $\mathbb{Z}_5$. The way you add two elements $[u]_5$ and $[v]_5$ is that $[u]_5 + [v]_5 = [u + v]_5$ and you similarly can multiply them by $[u]_5 \cdot [v]_5 = [u \cdot v]_5$. In other words, in order to add or multiply two elements $[u]_5$ and $[v]_5$ of $\mathbb{Z}_5$, you must first add or multiply u and v and then find their equivalence class mod 5 containing this sum or product of u and v. So

$$[2]_5 + [4]_5 = [2 + 4]_5 = [6]_5 = [1]_5$$

and

$$[2]_5 \cdot [4]_5 = [2 \cdot 4]_5 = [8]_5 = [3]_5.$$

This addition and multiplication of equivalence classes mod 5 gives the same answer regardless of what elements of the two equivalence classes you use to perform the addition or multiplication. So if we use $[2]_5 = [17]_5$ and $[4]_5 = [-21]_5$, then

$$[2]_5 + [4]_5 = [17]_5 + [-21]_5 = [17 - 21]_5 = [-4]_5 = [-4 + 5]_5 = [1]_5$$

and

$$[2]_5 \cdot [4]_5 = [17]_5 \cdot [-21]_5 = [17 \cdot (-21)]_5 = [-357]_5 = [-357 + 360]_5 = [3]_5.$$

Now let's have *Maple* do some mod 5 arithmetic for us using the command *modp*, and also produce the addition and multiplication tables mod 5 in two matrices. *Maple* does not use the square bracket and subscript notation as we do since it treats our equivalence classes mod 5 as integers.

```
> modp(17, 5);
```

$$2$$

```
> modp(-21, 5);
```

$$4$$

```
> modp(modp(17, 5)+modp(-21, 5), 5);
```

$$1$$

```
> modp(modp(17, 5)*modp(-21, 5), 5);
```

$$3$$

```
> with(linalg):
> AdditionTableMod5 := matrix(6, 6, 0):
> AdditionTableMod5[1, 1] := "+":
> for j from 2 to 6 do
    AdditionTableMod5[1, j] := [j-2]:
  end do:
> for j from 2 to 6 do
    AdditionTableMod5[j, 1] := [j-2]:
  end do:
> for j from 2 to 6 do
    for k from 2 to 6 do
      AdditionTableMod5[j, k] := modp(j-2+(k-2), 5):
    end do:
  end do:
> print(AdditionTableMod5);
```

$$\begin{bmatrix} \text{"+"} & [0] & [1] & [2] & [3] & [4] \\ [0] & 0 & 1 & 2 & 3 & 4 \\ [1] & 1 & 2 & 3 & 4 & 0 \\ [2] & 2 & 3 & 4 & 0 & 1 \\ [3] & 3 & 4 & 0 & 1 & 2 \\ [4] & 4 & 0 & 1 & 2 & 3 \end{bmatrix}$$

```
> MultTableMod5 := matrix(6, 6, 0):
```

```
> MultTableMod5[1, 1] := "*":
> for j from 2 to 6 do
      MultTableMod5[1, j] := [j-2]:
  end do:
> for j from 2 to 6 do
      MultTableMod5[j, 1] := [j-2]:
  end do:
> for j from 2 to 6 do
      for k from 2 to 6 do
        MultTableMod5[j, k] := modp((j-2)*(k-2), 5):
      end do:
  end do:
> print(AdditionTableMod5);
```

$$\begin{bmatrix} \text{“*”} & [0] & [1] & [2] & [3] & [4] \\ [0] & 0 & 0 & 0 & 0 & 0 \\ [1] & 0 & 1 & 2 & 3 & 4 \\ [2] & 0 & 2 & 4 & 1 & 3 \\ [3] & 0 & 3 & 1 & 4 & 2 \\ [4] & 0 & 4 & 3 & 2 & 1 \end{bmatrix}$$

Example 1.1.3. (Subtraction and Division in Modular Arithmetic in $\mathbb{Z}_5$.) Now we can do the inverse operations to addition and multiplication in $\mathbb{Z}_5$, subtraction and division. For any element $[u]_5$ of Z_5, its additive inverse is $[-u]_5$ since

$$[u]_5 + [-u]_5 = [u - u]_5 = [0]_5$$

where $[0]_5$ is the zero element of $\mathbb{Z}_5$. So in order to subtract $[u]_5$ in $\mathbb{Z}_5$, we instead add $[-u]_5$, in other words,

$$[v]_5 - [u]_5 = [v]_5 + [-u]_5 = [v - u]_5.$$

As an example,

$$[2]_5 - [4]_5 = [2 - 4]_5 = [-2]_5 = [3]_5.$$

Unfortunately, division in $\mathbb{Z}_5$ is more complicated than the other 3 arithmetic operations since we have no fractions in Z_5. If we want to divide by $[u]_5 \neq [0]_5$, then we need to find an element $[v]_5$ where $[u]_5 \cdot [v]_5 = [1]_5$ so that $[v]_5$ is the multiplicative inverse of $[u]_5$. In other words, if we want to divide by $[u]_5$, we instead multiply by its multiplicative inverse $[v]_5$. As an

example, if we want to divide by $[2]_5$ we instead multiply by $[3]_5$ since $[3]_5$ is $[2]_5$'s multiplicative inverse because $[2]_5 \cdot [3]_5 = [1]_5$. Thus,

$$\frac{1}{[2]_5} = [2]_5^{-1} = [3]_5.$$

In general, it is a non-trivial task to compute multiplicative inverses in modular arithmetic, happily *Maple* can do it for us as long as the base N is a prime since only in $\mathbb{Z}_p$, for p a prime, does every non-zero element have a multiplicative inverse. Note below that *Maple* tells us that

$$\frac{1}{[18]_{83}} = [18]_{83}^{-1} = [60]_{83}.$$

```
> modp(1/2, 5);
                                3
> modp(1/18, 83);
                               60
> modp(18*60, 83);
                                1
> type(83, prime);
                              true
```

1.2 Matrix Encryption

In this section we want to use modular arithmetic mod a prime p to encode messages to numerical values mod p along with multiplication by square matrices to do encryption and decryption of messages. First, we must create an alphabet for our messages consisting of a prime p number of elements. For this we will use the English alphabet of 26 capital letters along with the punctuation marks of period, comma, colon, question mark and a blank space to separate words. This gives us an alphabet of 31 symbols for 31 being prime. If you want to include the digits 0 through 9, then you must enlarge your alphabet by these 10 symbols to get an alphabet of 41 symbols which is also prime. We will not include these 10 digits in our alphabet below, but you should do so as an exercise.

Our next step is to assign the symbols of our alphabet to the elements of $\mathbb{Z}_{31}$ where $A = [0]_{31}$, $B = [1]_{31}, \ldots, Z = [25]_{31}$, period(.)= $[26]_{31}$, comma(,)= $[27]_{31}$, blank space( )= $[28]_{31}$, question mark(?)= $[29]_{31}$, and a colon(:)= $[30]_{31}$. This will allow us to change our messages into numerical lists of values taken

from $\mathbb{Z}_{31}$. We will also need to do the reverse assignment to turn our numerical messages back into the English alphabet. Our message written in $\mathbb{Z}_{31}$ will be said to be encoded, but not yet encrypted.

Next we can break our message into blocks of the same length, say blocks of size K whose elements are from $\mathbb{Z}_{31}$. If your message is not of length an integer multiple of K, then use blank spaces to make this happen. It is these blocks of length K which will be encrypted by multiplying them (written as columns) by a square matrix E of size $K \times K$ whose entries are also from $\mathbb{Z}_{31}$. The decryption will be done by multiplying the encoded message blocks by E^{-1} which will exist if $[\det(E)]_{31} \neq [0]_{31}$. This encryption matrix E can be chosen at random and E can be changed at random in order to prevent someone from discovering the value of E and so breaking your code.

Example 1.2.1. As our first step we must have *Maple* assign values of $\mathbb{Z}_{31}$ to our alphabet of capital letters and symbols using the table command with the table called *PlainTextToMod31*, and also reverse this assignment with another table called *Mod31ToPlainText*.

> PlainTextToMod31 := table([“A” = 0, “B” = 1, “C” = 2, “D” = 3, “E” = 4, “F” = 5, “G” = 6, “H” = 7, “I” = 8, “J” = 9, “K” = 10, “L” = 11, “M” = 12, “N” = 13, “O” = 14, “P” = 15, “Q” = 16, “R” = 17, “S” = 18, “T” = 19, “U” = 20, “V” = 21, “W” = 22, “X” = 23, “Y” = 24, “Z” = 25, “.” = 26, “,” = 27, “ ” = 28, “?” = 29, “:” = 30]):

> PlainTextToMod31[“ ”];

28

> message1 := convert(“IS MATH THE KING OF THE SCIENCES, AND PHYSICS IS ITS QUEEN?”, list);

message1 := [“I”, “S”, “ ”, “M”, “A”, “T”, “H”, “ ”, “T”, “H”, “E”, “ ”, “K”, “I”, “N”, “G”, “ ”, “O”, “F”, “ ”, “T”, “H”, “E”, “ ”, “S”, “C”, “I”, “E”, “N”, “C”, “E”, “S”, “,”, “ ”, “A”, “N”, “D”, “ ”, “P”, “H”, “Y”, “S”, “I”, “C”, “S”, “ ”, “I”, “S”, “ ”, “I”, “T”, “S”, “ ”, “Q”, “U”, “E”, “E”, “N”, “?”]

> encodedmessage1 := [seq(PlainTextToMod31[message1[k]], k =1..nops(message1))];

encodedmessage1 := [8, 18, 28, 12, 0, 19, 7, 28, 19, 7, 4, 28, 10, 8, 13, 6, 28, 14, 5, 28, 19, 7, 4, 28, 18, 2, 8, 4, 13, 2, 4, 18, 27, 28, 0, 13, 3, 28, 15, 7, 24, 18, 8, 2, 18, 28, 8, 18, 28, 8, 19, 18, 28, 16, 20, 4, 4, 13, 29]

Now we use a table called *Mod31ToPlainText* to decode our messages from mod 31 back to plain text in our alphabet.

> Mod31ToPlainText := table([0 = “A”, 1 = “B”, 2 = “C”, 3 = “D”, 4 = “E”, 5 = “F”, 6 = “G”, 7 = “H”, 8 = “I”, 9 = “J”, 10 = “K”, 11 = “L”, 12 = “M”, 13 = “N”, 14 = “O”, 15 = “P”, 16 = “Q”, 17 = “R”, 18 = “S”, 19 = “T”, 20 = “U”, 21 = “V”, 22 = “W”, 23 = “X”, 24 = “Y”, 25 = “Z”, 26 = “.”, 27 = “,”, 28 = “ ”, 29 = “?”, 30 = “:”])

> Mod31ToPlainText[9];

“J”

> decodedmessage1 := [seq(Mod31ToPlainText[encodedmessage1[k]], k = 1 .. nops(encodedmessage1))];

decodedmessage1 := [“I”, “S”, “ ”, “M”, “A”, “T”, “H”, “ ”, “T”, “H”, “E”, “ ”, “K”, “I”, “N”, “G”, “ ”, “O”, “F”, “ ”, “T”, “H”, “E”, “ ”, “S”, “C”, “I”, “E”, “N”, “C”, “E”, “S”, “,”, “ ”, “A”, “N”, “D”, “ ”, “P”, “H”, “Y”, “S”, “I”, “C”, “S”, “ ”, “I”, “S”, “ ”, “I”, “T”, “S”, “ ”, “Q”, “U”, “E”, “E”, “N”, “?”]

> plaintextdecodedmessage1 := cat(seq(convert(decodedmessage1[k], symbol), k = 1 .. nops(encodedmessage1)));

plaintextdecodedmessage1 := *IS MATH THE KING OF THE SCIENCES, AND PHYSICS IS ITS QUEEN?*

Now we need to decide on a block size K to break our encoded messages into so that we can multiply these blocks by the encryption matrix E which is $K \times K$ and whose entries are in $\mathbb{Z}_{31}$. This matrix E must have an inverse matrix E^{-1} mod 31 (we need $[\det(E)]_{31} \neq [0]_{31}$) so that we can decrypt our messages by multiplication by E^{-1}. Let $K = 3$ for simplicity. We will have *Maple* pick the 3×3 matrix E randomly.

> with(linalg):

> E := matrix(3, 3, rand(0 .. 30));

$$E := \begin{bmatrix} 28 & 22 & 14 \\ 25 & 12 & 5 \\ 1 & 26 & 11 \end{bmatrix}$$

> modp(det(E), 31);

10

> nops(message1);

59

> modp(nops(message1), 3);

2

Our message called message1 is not of length a multiple of 3 since it has 59 characters or symbols. So we must place a blank space at the end of our message so that it has length 60.

> newmessage1 := convert(“IS MATH THE KING OF THE SCIENCES, AND PHYSICS IS ITS QUEEN? ”, list);

newmessage1 := [“I”, “S”, “ ”, “M”, “A”, “T”, “H”, “ ”, “T”, “H”, “E”, “ ”, “K”, “I”, “N”, “G”, “ ”, “O”, “F”, “ ”, “T”, “H”, “E”, “ ”, “S”, “C”, “I”, “E”, “N”, “C”, “E”, “S”, “,”, “ ”, “A”, “N”, “D”, “ ”, “P”, “H”, “Y”, “S”, “I”, “C”, “S”, “ ”, “I”, “S”, “ ”, “I”, “T”, “S”, “ ”, “Q”, “U”, “E”, “E”, “N”, “?”, “ ”]

> nops(newmessage1);

60

> encodednewmessage1 := [seq(PlainTextToMod31[newmessage1[k]], k = 1 .. nops(newmessage1))];

encodednewmessage1 := [8, 18, 28, 12, 0, 19, 7, 28, 19, 7, 4, 28, 10, 8, 13, 6, 28, 14, 5, 28, 19, 7, 4, 28, 18, 2, 8, 4, 13, 2, 4, 18, 27, 28, 0, 13, 3, 28, 15, 7, 24, 18, 8, 2, 18, 28, 8, 18, 28, 8, 19, 18, 28, 16, 20, 4, 4, 13, 29, 28]

> blockedencodednewmessage1 := transpose(matrix(20, 3, encodednewmessage1))

blockedencodednewmessage1 :=

$$\begin{bmatrix} 8 & 12 & 7 & 7 & 10 & 6 & 5 & 7 & 18 & 4 & 4 & 28 & 3 & 7 & 8 & 28 & 28 & 18 & 20 & 13 \\ 18 & 0 & 28 & 4 & 8 & 28 & 28 & 4 & 2 & 13 & 18 & 0 & 28 & 24 & 2 & 8 & 8 & 28 & 4 & 29 \\ 28 & 19 & 19 & 28 & 13 & 14 & 19 & 28 & 8 & 2 & 27 & 13 & 15 & 18 & 18 & 18 & 19 & 16 & 4 & 28 \end{bmatrix}$$

> blockedencryptednewmessage1 := map(modp, evalm(E&*blockedencodednewmessage1), 31);

blockedencryptednewmessage1 :=

$$\begin{bmatrix} 20 & 13 & 24 & 25 & 18 & 19 & 30 & 25 & 9 & 23 & 18 & 5 & 11 & 15 & 24 & 3 & 17 & 11 & 22 & 30 \\ 29 & 23 & 17 & 22 & 8 & 29 & 29 & 22 & 18 & 18 & 17 & 21 & 21 & 26 & 4 & 18 & 23 & 29 & 10 & 7 \\ 9 & 4 & 14 & 16 & 20 & 20 & 12 & 16 & 3 & 23 & 25 & 16 & 28 & 23 & 10 & 0 & 11 & 23 & 13 & 21 \end{bmatrix}$$

> with(ListTools):
> encryptedencodednewmessage1:=Flatten(convert(transpose(blockedencryptednewmessage1), listlist)) ;

encryptedencodednewmessage1 := [20, 29, 9, 13, 23, 4, 24, 17, 14, 25, 22, 16, 18, 8, 20, 19, 29, 20, 30, 29, 12, 25, 22, 16, 9, 18, 3, 23, 18, 23, 18, 17, 25, 5, 21, 16, 11, 21, 28, 15, 26, 23, 24, 4, 10, 3, 18, 0, 17, 23, 11, 11, 29, 23, 22, 10, 13, 30, 7, 21]

> encrypteddecodednewmessage1 := [seq(Mod31ToPlainText[encrypteddecodednewmessage1[k]], k = 1 .. nops(encrypteddecodednewmessage1))];

encrypteddecodednewmessage1 := [“U”, “?”, “J”, “N”, “X”, “E”, “Y”, “R”, “O”, “Z”, “W”, “Q”, “S”, “I”, “U”, “T”, “?”, “U”, “:”, “?”, “M”, “Z”, “W”, “Q”, “J”, “S”, “D”, “X”, “S”, “X”, “S”, “R”, “Z”, “F”, “V”, “Q”, “L”, “V”, “ ”, “P”, “.”, “X”, “Y”, “E”, “K”, “D”, “S”, “A”, “R”, “X”, “L”, “L”, “?”, “X”, “W”, “K”, “N”, “:”, “H”, “V”]

Now we want to use multiplication by E^{-1} to turn *blockedencryptednewmessage1* back into *blockedencodednewmessage1*, and then back to *newmessage1*.

> inverseE := map(modp, inverse(E), 31);

$$inverseE := \begin{bmatrix} 25 & 6 & 19 \\ 4 & 17 & 21 \\ 8 & 10 & 22 \end{bmatrix}$$

> map(modp, evalm(E &*inverseE), 31);

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

> map(modp, evalm(inverseE&*blockedencryptednewmessage1), 31);

$$\begin{bmatrix} 8 & 12 & 7 & 7 & 10 & 6 & 5 & 7 & 18 & 4 & 4 & 28 & 3 & 7 & 8 & 28 & 28 & 18 & 20 & 13 \\ 18 & 0 & 28 & 4 & 8 & 28 & 28 & 4 & 2 & 13 & 18 & 0 & 28 & 24 & 2 & 8 & 8 & 28 & 4 & 29 \\ 28 & 19 & 19 & 28 & 13 & 14 & 19 & 28 & 8 & 2 & 27 & 13 & 15 & 18 & 18 & 18 & 19 & 16 & 4 & 28 \end{bmatrix}$$

> Flatten(convert(transpose(%), listlist));

[8, 18, 28, 12, 0, 19, 7, 28, 19, 7, 4, 28, 10, 8, 13, 6, 28, 14, 5, 28, 19, 7, 4, 28, 18, 2, 8, 4, 13, 2, 4, 18, 27, 28, 0, 13, 3, 28, 15, 7, 24, 18, 8, 2, 18, 28, 8, 18, 28, 8, 19, 18, 28, 16, 20, 4, 4, 13, 29, 28]

> [seq(Mod31ToPlainText[%[k]], k = 1 .. nops(%))];

[“I”, “S”, “ ”, “M”, “A”, “T”, “H”, “ ”, “T”, “H”, “E”, “ ”, “K”, “I”, “N”, “G”, “ ”, “O”, “F”, “ ”, “T”, “H”, “E”, “ ”, “S”, “C”, “I”, “E”, “N”, “C”, “E”, “S”, “,”, “ ”, “A”, “N”, “D”, “ ”, “P”, “H”, “Y”, “S”, “I”, “C”, “S”, “ ”, “I”, “S”, “ ”, “I”, “T”, “S”, “ ”, “Q”, “U”, “E”, “E”, “N”, “?”, “ ”]

> cat(seq(convert(%[k], symbol), k = 1 .. nops(%)));

IS MATH THE KING OF THE SCIENCES, AND PHYSICS IS ITS QUEEN?