* Mixing with matrices

Master Course, Spring term 2025

Fernando Chamizo https://matematicas.uam.es/~fernando.chamizo/

Contents. Elements of a cryptosystem. Hill cipher. Some matrix constructions.

2.1 Basics of cryptography

In cryptography we have a set of possibles *messages*, commonly called *plaintexts*, $\mathcal{P}$ that we are going to consider to be a set of integer numbers (very often representing classes to some modulo). Commonly this requires an *encoding scheme* passing from our raw data to numbers. For instance, ASCII [7] encodes each character (letters, digits and something else) into a number of 8 bits, an integer in $[0,256)$. If our $\mathcal{P}$ contains all the numbers with 800 bits we can break any text into pieces of 100 characters and consider each of them as an element of $\mathcal{P}$.

The idea is to use a *key* in a set $\mathcal{K}$ to encrypt a message through an *encryption function*

$$e : \mathcal{K} \times \mathcal{P} \longrightarrow \mathcal{C}$$

for some image set $\mathcal{C}$ whose elements are called *ciphertexts*. Of course, we want to recover the original information at some point, so we need a *decryption function*

$$d : \mathcal{K} \times \mathcal{C} \longrightarrow \mathcal{P} \qquad \text{such that} \quad d\big(x, e(x,y)\big) = y.$$

A *cipher* is an algorithm for encryption and decryption, a recipe for e and d, an a *cryptosystem* is the tuple $\{\mathcal{P}, \mathcal{C}, \mathcal{K}, e, d\}$.

Actually, the previous definition is closer to that of a so-called *symmetric cipher* [3, 1.7.1], but we will not enter into these details now.

A very simple cipher, called *Caesar cipher*, consists in shifting characters. Using ASCII, take $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}/256\mathbb{Z}$, identified with the standard representatives $\{0, 1, \dots, 255\}$, $e(k,n) = k+n$ and $d(k,n) = n-k$. For instance, if $k = 2$, with the usual encoding, the cipher encrypts `A` as `C` and `F` as `H`.

The main target in cryptography is to find cryptosystems in which recovering the plaintext from the ciphertext is very difficult without knowing the key. In rough mathematical terms, we look for *trap functions*, easy to apply and difficult to invert without extra information. In this sense, Caesar cipher is quite weak because the key is fairly easy to guess, for instance trying all the possibilities or analyzing the frequencies of the characters.

A stronger cipher is *Hill cipher*, the main topic here. In this case $\mathcal{P}$ and $\mathcal{C}$ are the linear space $(\mathbb{Z}/p\mathbb{Z})^n$ with p a prime and $\mathcal{K} = \mathrm{GL}_n(\mathbb{Z}/p\mathbb{Z})$, the set of nonsingular matrices with entries

in $\mathbb{Z}/p\mathbb{Z}$. The "G" and "L" stand for "general" and "linear". Strictly speaking $\mathcal{P}$ and $\mathcal{C}$ are not sets of numbers, but it is not a big deal because a vector in $(\mathbb{Z}/p\mathbb{Z})^n$ can be considered as a list of numbers that can be taken as the digits of a very big number in base p. The encryption and decryption functions are

$$e(M,\vec{v}) = M\vec{v} \qquad \text{and} \qquad d(M,\vec{v}) = M^{-1}\vec{v}.$$

For instance, $p = 257$ covers all the possibilities for an ASCII based encoding of single characters. Taking

$$M = \begin{pmatrix} 60 & 202 & 237 & 247 \\ 104 & 89 & 31 & 49 \\ 60 & 100 & 69 & 233 \\ 132 & 216 & 40 & 1 \end{pmatrix} \in \mathrm{GL}_4(\mathbb{Z}/257\mathbb{Z}), \qquad \text{with inverse} \quad \begin{pmatrix} 1 & 187 & 29 & 24 \\ 237 & 116 & 64 & 21 \\ 166 & 58 & 230 & 226 \\ 118 & 111 & 133 & 219 \end{pmatrix},$$

we can encrypt any text of $4k$ characters grouping them into blocks (vectors) of 4 characters. This is a sagemath function to do it:

```
def ee(text):
        """
        Encode and encrypt
        """
        if (len(text)%4)!=0: return 'The length must be multiple of 4'
        M = matrix(IntegerModRing(257), 4,4, [ 60, 202, 237, 247,  104,  89,
                ↪ 31,  49,   60, 100,  69, 233,  132, 216,  40,   1])
        L = []
        for k in range(0,len(text),4):
                v = M*vector([ord(_) for _ in list(text[k:k+4])])
                L += v.list()
        return L
```

This function encodes and encrypt the text "`Modular form`" as the list

```
[228, 249, 23, 221, 87, 38, 148, 222, 243, 64, 111, 218].
```

To decrypt and decode, we proceed with a similar function:

```
def dd(alist):
        """
        Decode and decrypt
        """
        if (len(alist)%4)!=0: return 'The length must be multiple of 4'
        Mi = matrix(IntegerModRing(257), 4,4, [ 1, 187, 29, 24,  237,  116,
                ↪ 64,  21,   166, 58,  230, 226,  118, 111,  133,  219])
        text = ''
        for k in range(0,len(alist),4):
                v = Mi*vector( alist[k:k+4] )
                text += ''.join( [chr(ZZ(v[_])) for _ in range(4)]  )
        return text
```

Hill cipher has mainly academic interest. One of its drawbacks is that it is vulnerable to *plaintext attacks*. This means that with some pairs (p, c) of plaintexts and corresponding ciphertexts, we can recover the key. In plain words, once we know the decryption of some messages we will able to decrypt all. It reduces to simple linear algebra: Given n linearly independent vectors in $(\mathbb{Z}/p\mathbb{Z})^n$ and knowing their images by a matrix $M \in \mathrm{GL}_n(\mathbb{Z}/p\mathbb{Z})$, which represents an endomorphism, it is possible to determine M.

2.2 Building easy matrices

Let us use Hill cipher as an excuse to play with matrix algebra and to recall some theorems.

The quick creation of encryption and decryption functions leads to the construction of nontrivial integers matrices having easy-to-compute integer inverses. These matrices can be reduced modulo p to encrypt and decrypt.

Clearly, $M \in \mathcal{M}_{n\times n}(\mathbb{Z})$ and $M^{-1} \in \mathcal{M}_{n\times n}(\mathbb{Z})$ if and only if $\det(M) = \pm 1$. Passing from a sign to another is as simple as changing the sign of a column, then the problem becomes to construct nontrivial elements of $\mathrm{SL}_n(\mathbb{Z})$ easy to invert. In $\mathrm{SL}_2(\mathbb{Z})$ the problem does not exist because any element is easy to invert:

$$\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z}) \qquad \Longrightarrow \qquad \gamma^{-1} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

For a and c coprime chosen randomly Euclid's algorithm gives very quickly b and d to complete an $\mathrm{SL}_2(\mathbb{Z})$ matrix.

With an algebraic device it is possible to lift n-uples of $\mathrm{SL}_2(\mathbb{Z})$ matrices to $\mathrm{SL}_{2^n}(\mathbb{Z})$ very easy to invert. For $A, B \in \mathcal{M}_{n\times n}(\mathbb{Z})$ the *Kronecker product* of A and B is defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{pmatrix}$$

The definition extends to any unit commutative ring, in particular to $\mathbb{Z}/p\mathbb{Z}$, and, with obvious modifications to matrices of any dimensions. As the notation suggests, it is related to the tensor product [4, 4.2.7].

Proposition 2.1. *The Kronecker product in $\mathcal{M}_{n\times n}$ is bilinear and associative. It also verifies*

$$\det(A \otimes B) = \big(\det(A)\det(B)\big)^n \qquad \textit{and} \qquad (A \otimes B)(C \otimes D) = (AC) \otimes (BD).$$

The second property implies that if A and B are invertible then $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.

The following sagemath code produces large matrices and their inverses inexpensively with this idea.

```
# Number of matrices
m = 3
# Size of the entries
N = 100
# Prime
p = 103

ML = [] # Matrix list
for k in range(m):
        a, c = 0,0
        while gcd(a,c) != 1: a, c = randint(1, N), randint(1, N)
        # build an SL_2(Z) matrix and add it to ML
        ML.append( matrix(2,2,[a,-xgcd(a,c)[2],c,xgcd(a,c)[1]]) )

A, Ai = ML[0], ML[0].inverse()
ML = ML[1:]
for item in ML:
        A = A.tensor_product(item)
```

```
        Ai = Ai.tensor_product(item.inverse())

Ap, Aip =  MatrixSpace(IntegerModRing(p), 2^m, 2^m)(A),
        ↪ MatrixSpace(IntegerModRing(p), 2^m, 2^m)(Ai)
```

The output is `A` and `Ai`, a pair of an $\mathrm{SL}_{2^{\mathtt{m}}}(\mathbb{Z})$ matrix and its inverse, and `Ap` and `Aip`, the same matrices modulo `p`.

A more flexible method to produce matrices in $\mathrm{SL}_n(\mathbb{Z})$ is to recall the LU decomposition from numerical analysis [1]. Choosing random $n \times n$ lower and upper triangular matrices L and U with ones in the diagonal, it is obvious that $LU \in \mathrm{SL}_n(\mathbb{Z})$. A possible sagemath implementation is

```
def rand_triang(d):
        """
        random upper triangular of dimension d
        """
        L = []
        for k in srange(d^2):
                if (k%d) < floor(k/d): L.append(0)
                elif (k%d) == floor(k/d): L.append(1)
                else: L.append( randint(-N, N) )
        return matrix(d,d,L)

# Dimension
d = 6
# Size of the entries
N = 5

U, L = rand_triang(d), rand_triang(d).transpose()
```

The point is that L and U can be easily inverted by forward and backward substitution.

Even if one has an efficient method to create pairs of integer matrices M and M^{-1}, a natural question is if it is possible to use the same matrix to encrypt and decrypt. The answer is positive considering

$$M = ADA^{-1} \quad \text{with} \quad A \in \mathrm{SL}_n(\mathbb{Z}) \quad \text{and} \quad D \text{ diagonal with } d_{ii} \in \{-1, 1\}$$

because $M^2 = I$, it is its own inverse.

If n is not very large and we choose randomly d_{ii}, we take the risk of getting M with some zeros or even $M = I$. A variant is to look for matrices such that $M^m = I$, in this way, $M^{-1} = M^{m-1}$. To fulfill $M^m = I$ the eigenvalues must be powers of $\zeta = e^{2\pi i/m}$. If ζ is one of them, to optimize the dimension the characteristic polynomial must be $\pm C_m$ with C_m the *cyclotomic polynomial* $\prod_{\gcd(k,m)=1}(x - \zeta^k)$ which is known to be an irreducible monic element of $\mathbb{Z}[x]$ of degree $\varphi(m)$, the minimal polynomial of ζ, [6]. The *Frobenius normal form* [5, §6.3] makes trivial to find $F \in \mathcal{M}_{d\times d}(\mathbb{Z})$ having C_m as its characteristic polynomial, with $d = \varphi(m)$. It is enough to appeal to this result:

Lemma 2.2. *Consider the polynomial* $P = x^n + a_{n-1}x^{n-1} + \cdots + a_0$. *Then*

$$F = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & 0 & \dots & 0 & -a_1 \\ 0 & 1 & 0 & \dots & 0 & -a_2 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -a_{n-1} \end{pmatrix} \qquad \textit{verifies} \qquad \det(F - \lambda I) = P(\lambda).$$

The choice $M = AFA^{-1}$ with $A \in \mathrm{SL}_d(\mathbb{Z})$ as before makes the job. Once we have chosen A, the following lines in sagemath

```
p = ZZ['x'].cyclotomic_polynomial(m)
F = companion_matrix(p)
M = A*F*A.inverse()
```

implement the method. By the way, this circle of ideas can be exploited to show that the possible orders in $\mathrm{SL}_2(\mathbb{Z})$ are 1, 2, 3, 4, 6 and ∞ (see [2, Th.2.7] for an elementary proof).

A final remark is that the reduction modulo p, $\mathrm{SL}_d(\mathbb{Z}) \longrightarrow \mathrm{SL}_d(\mathbb{Z}/p\mathbb{Z})$, can reduce the order. In connection with this, obviously, any element of infinite order becomes of finite order.

Exercises of lecture 2

Exercise 1. Write a short sagemath code applying the encryption function of Caesar cipher for a given key $k \in \mathbb{Z}/256\mathbb{Z}$ to the characters of any text stringgiving as output as a list with elements in $\mathbb{Z}/256\mathbb{Z}$

Exercise 2. Write a short sagemath code to recover the original text string when k and the output of the previous exercise are taken as inputs.

Exercise 3. Consider a Hill cipher with $p = 11$ and $n = 2$. If the plaintext $(1, 3)$ is encrypted as $(9, 4)$ and $(2, 5)$ as $(1, 1)$, find the key.

Exercise 4. Prove the two formulas in Proposition 2.1.

Exercise 5. Prove Lemma 2.2.

Exercise 6. If T is a nonsingular $n \times n$ triangular matrix with ones in the diagonal, show $T^{-1} = I + R + R^2 + \cdots + R^{n-1}$ with $R = I - T$.

Exercise 7. Implement in sagemath an efficient algorithm to compute U^{-1} for U any nonsingular upper triangular matrix.

References

[1] K. E. Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, Inc., New York, second edition, 1989.

[2] K. Conrad. $\mathrm{SL}_2(\mathbb{Z})$. `https://kconrad.math.uconn.edu/blurbs/grouptheory/SL(2,Z).pdf`, 2023. Expository papers.

[3] J. Hoffstein, J. Pipher, and J. H. Silverman. *An introduction to mathematical cryptography*. Undergraduate Texts in Mathematics. Springer, New York, 2008.

[4] A. I. Kostrikin and Y. I. Manin. *Linear algebra and geometry*, volume 1 of *Algebra, Logic and Applications*. Gordon and Breach Science Publishers, Amsterdam, 1997. Translated from the second Russian (1986) edition by M. E. Alferieff.

[5] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12 of *Texts in Applied Mathematics*. Springer-Verlag, New York, second edition, 1993. Translated from the German by R. Bartels, W. Gautschi and C. Witzgall.

[6] S. H. Weintraub. Several proofs of the irreducibility of the cyclotomic polynomials. *Amer. Math. Monthly*, 120(6):537–545, 2013.

[7] Wikipedia contributors. ASCII — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=ASCII&oldid=1268373038`, 2025. [Online; accessed 24-January-2025].