HILL CIPHERS: A LINEAR ALGEBRA PROJECT WITH *MATHEMATICA*

Murray Eisenberg
University of Massachusetts
Mathematics and Statistics Department
Lederle Graduate Research Tower
Amherst, MA 01003-4515
murray@math.umass.edu

Abstract A project on Hill ciphers is discussed here that is suitable for a beginning linear algebra course. In the project, students use the computer algebra system MATHEMATICA to implement the Hill encipherment method and to "crack" a Hill cipher. This paper explains Hill ciphers as an application of linear algebra over $\mathbb{Z}_n$; describes the project from the student's point of view; and discusses the software implementation needed for the project.

Introduction Students' being able to learn math on their own is a desirable instructional goal. Software such as MATHEMATICA can facilitate achieving that goal. How this can be done is illustrated by a project on Hill ciphers—an engaging application of linear algebra to cryptology—in which students express the concepts in the form of MATHEMATICA functions which they then use to "crack" a Hill cipher, that is, to discover its key. Aside from its intrinsic interest, the topic of Hill ciphers requires students to learn some new linear algebra, namely, the situation in which the scalar field is finite.

Hill ciphers hardly have the currency of secret-key block ciphers such as Blowfish, or of public-key systems such as RSA or elliptic-curve methods. Yet they do have interest at least for historical reasons: they constitute the first general method for successfully applying algebra—specifically, linear algebra—to polygraphic ciphers.[1]

This project could be implemented in essentially any computer algebra system or general-purpose programming language (although to implement it completely as described, the CAS or language must be able to load packages that the student is unable to read). In fact, the author originally implemented the project in the array-based programming language APL, and more recently reimplemented it in Kenneth Iverson's programming language J. Implementing the project in a programming language such as C, Pascal, or Java that is not array-oriented would be more tedious for both instructor and students.

About Hill ciphers Letter-by-letter substitution ciphers easily succumb to frequency analysis and so are notoriously unsecure. Polygraphic ciphers, by contrast, in which each list of n consecutive letters of the plaintext—an n**-graph**—is replaced by another n-graph according to some key, can be more challenging to break. The first systematic yet simple polygraphic ciphers using more than two letters per group are the Hill

[1]See Kahn [6], especially pp. 404–408.

ciphers, first described by Lester Hill [4] in 1929.[2] For a polygraphic substitution, changing just one or two plaintext letters can completely change the corresponding ciphertext! That is one reason that Hill ciphers are so difficult to crack.

A **Hill** n**-cipher** works as follows. Start with an m-character **alphabet** and code each character with a unique integer in $\{0, 1, 2, \ldots, m-1\}$. The alphabet could consist of just the usual 26 letters in English or, as here, those letters supplemented with the 3 punctuation characters . and ? and ␣ (where ␣ denotes a blank space). For simplicity, the coding is done here by numbering the 29 characters in order as shown in Table 1. Next, choose as **key matrix** an $n \times n$ matrix A with entries in $\{0, 1, 2, \ldots, m-1\}$ that is invertible modulo m.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	?	␣
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Table 1: Numerical coding of 29-letter alphabet

Given a ciphertext string, encode it to the corresponding vector $\mathbf{v}$ of integers in the set $\{0, 1, 2, \ldots, m-1\}$. If the length of $\mathbf{v}$ is not an integral multiple k of the key size n, then **pad** $\mathbf{v}$ as needed by repeating its last entry. Partition $\mathbf{v}$ into k column vectors $\mathbf{v}_j$. Form each of the products $A\mathbf{v}_j$; equivalently, form the matrix product $A[\mathbf{v}_1 \mid \mathbf{v}_2 \mid \cdots \mid \mathbf{v}_k]$. Reduce the result modulo m. Reassemble the consecutive columns of this product into a new vector $\mathbf{w}$. Finally, decode $\mathbf{w}$ into the corresponding ciphertext string.

The following Hill 3-cipher illustrates the procedure for our 29-character alphabet with its encoding given in Table 1. The key matrix is:

$$A = \begin{bmatrix} 17 & 5 & 20 \\ 23 & 9 & 3 \\ 11 & 2 & 12 \end{bmatrix}$$

The plaintext is the following 10-character message (including the space and period):

WANT␣HELP.

First, encode the plaintext to the corresponding numbers:

22 0 13 19 28 7 4 11 15 26

Second, group these into trigraphs, repeating the final number twice to fill out the fourth group:

22 0 13   19 28 7   4 11 15   26 26 26

Third, form the 3×4 matrix having these groups of three numbers as its columns. Fourth, apply the key:

$$A \begin{bmatrix} 22 & 19 & 4 & 26 \\ 0 & 28 & 11 & 26 \\ 13 & 7 & 15 & 26 \end{bmatrix} \equiv \begin{bmatrix} 25 & 23 & 17 & 19 \\ 23 & 14 & 4 & 11 \\ 21 & 1 & 14 & 12 \end{bmatrix} \pmod{29}$$

[2]Hill [5] subsequently generalized his ciphers to a more complex scheme that used block matrices.

Fifth, decode the numbers

$$25\ \ 23\ \ 21\ \ 23\ \ 14\ \ 1\ \ 17\ \ 4\ \ 14\ \ 19\ \ 11\ \ 12$$

from the unraveled columns of the last matrix to obtain the letters of the ciphertext:

Z X V X O B R E O T L M

A Hill cipher is relatively immune from attack if its key size n is large enough to preclude frequency analysis of n-graphs. But it is easy to crack if you have "captured" enough plaintext along with the corresponding ciphertext, for then the method of the following theorem applies. (This method was not explained in Hill [4] but is formulated, for example, by Anton and Rorres [1]; a proof is included in the expository article [2].)

Cracking Theorem. *Suppose the alphabet length m is prime. Let $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ be n plaintext vectors for a Hill n-cipher having (unknown) key matrix A, and let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n$ be the corresponding ciphertext vectors. Suppose these plaintext vectors are* linearly independent *over $\mathbb{Z}_m$. Form the matrices $P = \left[\ \mathbf{p}_1 \mid \mathbf{p}_2 \mid \ldots \mid \mathbf{p}_n\ \right]$ and $C = \left[\ \mathbf{c}_1 \mid \mathbf{c}_2 \mid \ldots \mid \mathbf{c}_n\ \right]$ having the plaintext vectors and ciphertext vectors, respectively, as their columns. Then the same sequence of elementary row operations that reduces C^T to the identity matrix reduces P^T to the* transpose $(A^{-1})^T$ *of the inverse key matrix A^{-1}.*

The situation described by the theorem, which in practice would not be wholly unrealistic, is the one that arises in the project.

What the student does To start, the student reads a 19-page article [2] about Hill ciphers that describes the underlying mathematics regarding arithmetic modulo m and linear algebra over $\mathbb{Z}_m$; the Hill encipherment scheme; and the Cracking Theorem.

Next, the student defines in MATHEMATICA a main function `encipher` that takes as arguments a key matrix and plaintext string and returns as result the corresponding ciphertext string. He prepares auxiliary functions to do the requisite conversions between text to numbers and to arrange lists into matrices and vice versa. Guided by a MATHEMATICA notebook, the student next "validates" his `encipher`: he loads an instructor-prepared MATHEMATICA package that automatically generates various key matrix and plaintext pairs, evaluates the student's `encipher` at those pairs as arguments, and compares the results with what they should be.

Now the student is ready for the project problems. He loads another instructor-prepared MATHEMATICA package that creates data—key matrices, plaintexts, and ciphertexts—and stores them in named variables. As with the test pairs for validation, this data is individualized for the student through randomly generated key matrices and randomly selected entries from a database of texts. The package prints a list of three problems to be solved:

1. Given ciphertext and the *inverse* of the key matrix, decipher the text.
2. Given ciphertext and the key matrix, decipher the text.
3. Given the key matrix size and "captured" ciphertext along with corresponding plaintext, if possible "crack" the cipher—determine the inverse key.

Results Of all the computer-related linear algebra projects the author has assigned over the past twenty years, this one unquestionably has generated the most favorable student response. Nearly all students have been able to complete it, even if they were unable to get completely correct the details of padding (which can now be handled automatically with MATHEMATICA Version 4). And with many students one can almost see the "light bulb lighting up" as they realize that the function for decipherment is the same as that for encipherment, but with the key matrix replaced by its inverse.

Over many years the author has observed that students typically are quite persistent in carrying out computer projects to completion (and one can but dream that they would be just as persistent with problems that are not computer-related!). And of all these computer projects, with this one on Hill ciphers students been the most persistent. Perhaps the challenge of cracking a cipher is too enticing to ignore.

Implementation The suite of files needed for a student to carry out the project, available at the author's Web site [3], consists of:

- The expository article [2] about Hill ciphers, in Adobe `.pdf` format. (Given the primary purpose of the project, little would seem to be gained by making this article available in the interactive form of a MATHEMATICA notebook.)
- The MATHEMATICA notebook "About Hill" that guides the student through validating `encipher` and obtaining test data for the project problems.
- The MATHEMATICA packages that validate the student's `encipher` and generate the data for the project problems, respectively. The former includes a short database of plaintext strings used. The latter package does a hidden import of a third encoded package, which produces a database of plaintext strings—literary quotations, pithy sayings, etc. These packages are encoded.

These files are ready for student use once they are downloaded into a directory to which MATHEMATICA has direct access. For security reasons, only the encoded forms of the packages are available on the Web site. Instructors wishing to alter the packages—for example, to change the database of texts—should apply directly to the author for copies of the unencoded source MATHEMATICA notebooks; then the packages can be recreated in ASCII form and encoded again by MATHEMATICA.

In both packages, each key matrix is generated on the fly by iterating the process of forming a random matrix and checking it for invertibility modulo 29, with a default matrix being used if too many iterations are required.

Designing the packages involved two subtleties in MATHEMATICA programming. The first is conditional evaluation within a package: When the package is loaded, a function in it tests the value of a global variable (such as the student's name or ID number) to see if it has been defined and, if so, has the proper format; the package's "context" is created only if the global variable passes the test. The second involves indirectly creating symbolic names and assigning to them the values used for validation and problem data. For details, see the package source notebooks.

Extensions and complications The project, as currently implemented in MATHEMATICA, is about as simple as Hill ciphers can be. The project could be modified in one or more of the following ways:

- The student writes his own functions—or modifies instructor-provided ones that work over the field $\mathbb{R}$—to do row reduction and matrix inversion modulo the alphabet length m, instead of using the option `Modulus -> m` to MATHEMATICA's built-in `RowReduce` and `Inverse` functions.[3] Such new functions require finding inverses of integers modulo m. And this could be done with the brute-force method of using a lookup table or, for a mathematically more instructive approach, by programming and applying the euclidean algorithm.
- Allow, as did Hill himself, nonprime alphabet length m (for example, $m = 26$). Then include the problem of checking whether a given key matrix for such an alphabet is in fact invertible.
- Use affine transformations, as Hill did in his original paper [4], instead of just linear transformations.
- Make the cipher more realistic by using some arbitrary transposition cipher first.

The project could also be enhanced by including detection of key matrix size or by using frequency analysis to break the cipher even in the absence of a captured ciphertext-plaintext pair. Of course, such enhancement would take the project beyond the realm of a typical linear algebra course.

Acknowledgment The author thanks Allan Hayes and Tom Wickham-Jones for assistance with programming issues that arose in designing the MATHEMATICA packages for validation and problem generation.

References

[1] Howard Anton and Chris Rorres, *Elementary Linear Algebra: Applications Version,* 6 ed., Wiley, New York, 1991. See sec. 11.17.

[2] Murray Eisenberg, *Hill ciphers and modular linear algebra,* mimeographed notes, University of Massachusetts, 1998, 19 pages.

[3] Murray Eisenberg, *Hill ciphers*, web site, University of Massachusetts, URL `http://www.math.umass.edu/~murray/Hill_ciphers/`.

[4] Lester S. Hill, *Cryptography in an algebraic alphabet,* Amer. Math. Monthly **36** (1929) 306–312.

[5] Lester S. Hill, *Concerning certain linear transformation apparatus of cryptography,* Amer. Math. Monthly **38** (1931) 135–154.

[6] David Kahn, *The Codebreakers: The Story of Secret Writing,* Weidenfeld and Nicolson, London, 1967. See especially pp. 404–410.

[3] MATHEMATICA Version 2 does not allow the option `Modulus -> m` to `RowReduce`.