Omar Essa

Loyola University of New Orleans

April 18, 2024

ABSTRACT

We look at a particular semigroup of matrices and examine the idempotents, nilpotents, and zero divisors.

INTRODUCTION

In what follows, we denote S to be an arbitrary semigroup.

Definition: A semigroup S is a set with a single binary associative operation.

Example: Let N be the set of positive integers; then N forms a semigroup under addition, denoted as (N, +).

Example: The set (N, .) is a semigroup under multiplication.

Example: A square matrix M is an nxn table of numbers. NOTE: DEFINE MATRIX MULTIPLICATION (?) (Matrix multiplication is the multiplication of two nxn tables of numbers that involves taking the total sum of the dot products of each row of one matrix with each column of the other.) The set of nxn matrices over the real numbers forms a semigroup under matrix multiplication.

Definition: An element s in S is idempotent if s^2 = s.

Example: The matrix M = (10

1. is an idempotent because M^2 = M.

Definition: An element s in S is nilpotent of index n if s^n = 0 for some integer n > 1.

Example: The matrix M = (01

1. is nilpotent of index 2.

Definition: A left (right) zero divisor s in S is an element such that st = 0 (ts = 0) for some t in S.

Example: Let M = (00 Let N = (10; then MN = 0. Hence, M is a left zero divisor and N is a right zero divisor.

10) 00)

COMMENT: THIS PAPER WILL FOCUS MORE ON NON-SQUARE MATRICES

NON-SQUARE MATRICES

In what follows, we denote M and N to be nxm matrices multiplied to each other and P to be an mxn sandwich matrix in between to make the multiplication possible.

Describe project: In this project, we performed matrix multiplication between two nxm matrices M and N with the help of an mxn matrix P in between them that we called a sandwich matrix. Our requirements for M and N are that they should have a single non-zero element, and our requirements for P are that it should have at least one non-zero element in each row and column. Our goal was to perform the multiplication on as many different nxm matrices as possible to try to find certain patterns that arise with regard to the nxm matrices being nilpotents or idempotents, depending on each of the products. Because of the patterns that we were searching for, we focused on results that were either zero matrices or the same nxm matrices that were multiplied.

Definition: Matrix multiplication is the multiplication of an nxm table of numbers M with another nxm table of numbers N using an mxn table of numbers P in the middle that involves taking the total sum of the dot products of each row of one matrix with each column of the other.

Example: To multiply a 2x3 M = (100000) by another 2x3 N = (010000), we first find

(100 times P = (11 equals (11, which is then multiplied by (010, which equals (010

000) 11 00) 000) 000)

11)

Main Results: Every element is either idempotent or nilpotent.

(Write proof)

The number of nilpotents = the number of zeros in the sandwich matrix.

PROOF: Let A = E\_{i, j}. Then APA = E\_{i, j} P\_{j, i} E\_{i, j}}.

If P\_{j, i} = 0, then APA = 0, and A is nilpotent.

The number of idempotents = the number of ones in the sandwich matrix.

PROOF: Let A = E\_{i, j}. Then APA = E\_{i, j} P\_{j, i} E\_{i, j}}.

If P\_{j, i} = 1, then APA = 1.

You can determine whether an element is nilpotent or idempotent by looking at the sandwich matrix.

PROOF:

Multiplication that involves nxm matrices will result in the same number of nilpotents and idempotents as multiplication that involves mxn matrices

PROOF:

Further Questions: What patterns are there among the multiplied nxm matrices and their products regarding zero divisors?

How do the n and m dimensions affect the number of possible zero divisors, nilpotents, and idempotents?

PROGRAM AND OUTPUT

Describe program and say what results we got: We created a C++ program in Visual Studio to perform matrix multiplication at a much quicker rate than we would be able to manually on paper. We were able to implement the above requirements in our program using an incremental approach. We generated all possible mxn sandwich matrices first. Then we only took the ones that fit our requirements of at least 1 non-zero entry in every row and column. We then generated the nxm matrices following the requirements of 1 non-zero entry. After we were able to get that working, we began the process of multiplying the matrices and printing out the results. Finally, we printed out summaries with the matrices multiplied, the products, and counts of regular zero divisors, nilpotents, and idempotents so that we can see how the dimensions of matrices and the entries of matrices affect the products and classifications. The results we got are that the number of nilpotents was equal to the number of zeros in the sandwich matrix, and the number of idempotents was equal to the number of ones in the sandwich matrix. We also gained more evidence that every nxm matrix was either a nilpotent or idempotent and the nxm matrix being a nilpotent or idempotent depends on the sandwich matrix.

Program itself: (maybe paste program here, holding off for now so we can look at the above)

Zero Divisor Matrix Thesis

The world of zero divisors is an intriguing and, sometimes, complicated world within abstract algebra and modular arithmetic. Problems that involve zero divisors tend to include a set or ring of integers defined as Z, and those integers are mod some n. A set is a collection of distinct, well-defined objects that form a group, and a ring is an algebraic structure that consists of a set with addition and multiplication operations. For example, a problem with Z8 would include 2 and 4 as zero divisors because 2 times 4 mod 8 is 0, which makes them zero divisors because they can each be multiplied by another number and end up with a final result of 0 after calculating their product mod 8. For our purposes, we took a linear algebra approach to zero divisors. We took two “outer” matrices of the same dimensions and multiplied them together. However, this is usually not possible or is more difficult without a “sandwich” matrix in between the “outer” matrices. The “sandwich” matrix is one that has the opposite dimensions of the “outer” matrices. This allows the multiplication of the two “outer” matrices. We searched for zero divisors within the “outer” matrices, and what makes one a zero divisor is that it can be multiplied by a particular “sandwich” matrix and opposite “outer” matrix and result in a zero matrix. We also ended up focusing especially on nilpotents and idempotents. A nilpotent is an “outer” matrix that results in a zero matrix when it is multiplied by itself, and an idempotent is an “outer” matrix that results in itself when it is multiplied by itself. Through the use of a computer program and many different whiteboards, we were able to find that every “outer” matrix is either a nilpotent or an idempotent but can also result in a zero matrix when it is not multiplied by itself and the number of zeros in the “sandwich” matrix that the “outer” matrices are multiplied by corresponds with the number of “outer” matrices are nilpotents when multiplied by that “sandwich” matrix.

Our approach to the problem began with understanding zero divisors and our requirements. One requirement was that the “outer” matrices either had the same dimensions as the “sandwich” matrix or opposite dimensions to the “sandwich” matrix. This would mean that if the “outer” matrices have 2 rows and 3 columns, then the “sandwich” matrix has 3 rows and 2 columns and vice versa. All three matrices can also have 2 rows and 2 columns. The next requirement, for research purposes, is that the “outer” matrices must only have 1 non-zero entry. We decided to stick with binary matrices that only include 1s and 0s, but the patterns should apply to matrices with other numbers, too because the 0s are what make the patterns. The “sandwich” matrix must have at least 1 non-zero entry in each row and column. A final requirement that we had for our research was that, although regular zero divisors are good to observe, our main focus was on nilpotents and idempotents, which we were able to find patterns about. We were able to implement these requirements in our program using an incremental approach. We generated all possible “sandwich” matrices first. Then we only took the ones that fit our requirements of at least 1 non-zero entry in every row and column. We then generated the “outer” matrices following the requirements of 1 non-zero entry. After we were able to get that working, we began the process of multiplying the matrices and printing out the results. Finally, we printed out summaries with the matrices multiplied, the products, and counts of regular zero divisors, nilpotents, and idempotents so that we can see how the dimensions of matrices and the entries of matrices affect the products and classifications.