

Dodgson's Determinant: A Qualitative Analysis

**A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY**

Amy Dannielle Schmidt

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE**

PROFESSOR JOHN R. GREENE

May, 2011

Acknowledgements

I would like to take this opportunity to thank Dr. John Greene. As an algebra professor, he made a great impact on my growth as a young mathematician. I am grateful for his guidance in my research and his advice on career-shaping decisions.

Dr. Dalibor Froncek, Dr. Harlan Stech, Angela Sharp, and Chad Pierson have also been great mentors throughout my undergraduate and graduate careers at the University of Minnesota Duluth. I appreciate their advice and encouragement.

I would also like to thank Dr. Guihua Fei and Dr. John Pastor for serving on my thesis review committee and my friends and family for supporting me over the years.

Abstract

Condensation, developed by Charles Dodgson, is an uncommon method for calculating the determinant. It is generally considered to be numerically unstable due to its iterative nature. While we do not attempt to prove whether or not the algorithm is stable, we conduct a qualitative stability analysis. We compare the algorithm's performance to that of row reduction on contrived and random matrices. We have written two modified condensation algorithms for 3×3 and 4×4 matrices, which we include in our comparisons as well.

We also briefly investigate the relationship between the condition number of a matrix and the performance of these algorithms when used to calculate its determinant.

Contents

Acknowledgements	i
Abstract	ii
List of Tables	v
1 Introduction	1
1.1 Background	1
1.2 Statement of Problem	5
1.3 Literature Review	6
2 Preliminaries	7
2.1 Terminology	7
2.2 Precision	7
2.3 Floating-Point Representation Errors	9
2.4 Order of Magnitude	12
3 Methodology	13
3.1 Algorithms	13
3.2 Comparisons	16
4 Ill-Conditioned Matrices	18
5 3 by 3 Matrices	27
5.1 Contrived Matrices	27
5.2 Random Matrices	34

6	4 by 4 Matrices	36
6.1	Contrived Matrices	36
6.2	Random Matrices	39
7	Conclusion and Discussion	41
7.1	Summary	41
7.2	Future Work	42
	References	44
	Appendix A. <i>Mathematica</i> Code	45
A.1	Condensation	45
A.2	Modified Condensation	46
A.2.1	3 by 3 Matrices	46
A.2.2	4 by 4 Matrices	46
A.3	Row Reduction	47

List of Tables

4.1	Error comparisons for the determinants of Hilbert matrices of orders 2 through 15.	20
4.2	Error comparisons for the determinant of A_1 for $50 \leq x \leq 55$	21
4.3	Error comparisons for the determinant of A_2 for $50 \leq x \leq 55$	22
4.4	Error comparisons for the determinant of A_3 for $51 \leq x \leq 56$	22
4.5	Error comparisons for the determinant of A_4 for $50 \leq x \leq 55$	23
4.6	Error comparisons for the determinant of A_5 for $50 \leq x \leq 55$	24
4.7	Error comparisons for the determinant of A_6 for $50 \leq x \leq 55$	24
4.8	Error comparisons for the determinant of A_7 for $24 \leq x \leq 30$	26
4.9	Error comparisons for the determinant of A_8 for $0 \leq x \leq 5$	26
5.1	Error comparisons for the determinant of B_1 for $50 \leq x \leq 57$	28
5.2	Error comparisons for the determinant of B_2 for $39 \leq x \leq 54$	29
5.3	Error comparisons for the determinant of B_3 for $50 \leq x \leq 56$	33
5.4	Error comparisons for the determinant of B_4 for $50 \leq x \leq 56$	34
5.5	Failure comparisons for the determinants of 3×3 matrices with random elements.	35
6.1	Error comparisons for the determinant of C_1 for $50 \leq x \leq 55$	37
6.2	Error comparisons for the determinant of C_2 for $50 \leq x \leq 60$	38
6.3	Error comparisons for the determinant of C_3 for $47 \leq x \leq 54$	39
6.4	Failure comparisons for the determinants of 4×4 matrices with random elements.	40

Chapter 1

Introduction

1.1 Background

Charles Dodgson is best known for writing the *Alice* books under the pen name Lewis Carroll. In addition to writing works of fiction, he was also a nineteenth century mathematician. One of his contributions was an iterative method for calculating the determinant of a matrix, first published in *Proceedings of the Royal Society of London* [4] in 1866. The method is sometimes referred to as Dodgson's Condensation Method, or condensation, because with each iteration a matrix is replaced by a smaller one until a 1×1 matrix, the determinant, is reached. Each smaller matrix contains the 2×2 connected minors of the previous iteration's matrix. The 2×2 connected minors are the determinants of each 2×2 submatrix consisting of adjacent elements of the larger matrix. Beginning with the second stage of iteration, each of these minors is divided by their central element from two stages previous. To illustrate this, consider the following examples.

In the case of a 3×3 matrix, M , where we denote the i^{th} stage of iteration by M_{i+1} , we have

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \\ 4 & 9 & 16 \end{pmatrix},$$

$$M_2 = \left(\begin{array}{c|c|c} \left| \begin{array}{cc} 1 & 1 \\ 2 & 3 \end{array} \right| & \left| \begin{array}{cc} 1 & 1 \\ 3 & 4 \end{array} \right| & \\ \left| \begin{array}{cc} 2 & 3 \\ 4 & 9 \end{array} \right| & \left| \begin{array}{cc} 3 & 4 \\ 9 & 16 \end{array} \right| & \end{array} \right) = \begin{pmatrix} 1 & 1 \\ 6 & 12 \end{pmatrix},$$

$$M_3 = \frac{1(12) - 1(6)}{3} = 2 = \det M,$$

where containment of a submatrix by vertical bars denotes its determinant. In the 4×4 case we have

$$M = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{pmatrix},$$

$$M_2 = \left(\begin{array}{c|c|c|c} \left| \begin{array}{cc} 1 & 2 \\ 1 & 3 \end{array} \right| & \left| \begin{array}{cc} 2 & 4 \\ 3 & 9 \end{array} \right| & \left| \begin{array}{cc} 4 & 8 \\ 9 & 27 \end{array} \right| & \\ \left| \begin{array}{cc} 1 & 3 \\ 1 & 4 \end{array} \right| & \left| \begin{array}{cc} 3 & 9 \\ 4 & 16 \end{array} \right| & \left| \begin{array}{cc} 9 & 27 \\ 16 & 64 \end{array} \right| & \\ \left| \begin{array}{cc} 1 & 4 \\ 1 & 5 \end{array} \right| & \left| \begin{array}{cc} 4 & 16 \\ 5 & 25 \end{array} \right| & \left| \begin{array}{cc} 16 & 64 \\ 25 & 125 \end{array} \right| & \end{array} \right) = \begin{pmatrix} 1 & 6 & 36 \\ 1 & 12 & 144 \\ 1 & 20 & 400 \end{pmatrix},$$

$$M_3 = \left(\begin{array}{c|c|c|c} \frac{1}{3} \left| \begin{array}{cc} 1 & 6 \\ 1 & 12 \end{array} \right| & \frac{1}{9} \left| \begin{array}{cc} 6 & 36 \\ 12 & 144 \end{array} \right| & & \\ \frac{1}{4} \left| \begin{array}{cc} 1 & 12 \\ 1 & 20 \end{array} \right| & \frac{1}{16} \left| \begin{array}{cc} 12 & 144 \\ 20 & 400 \end{array} \right| & & \end{array} \right) = \begin{pmatrix} 2 & 48 \\ 2 & 120 \end{pmatrix},$$

$$M_4 = \frac{2(120) - 2(48)}{12} = 12 = \det M.$$

We generalize this process for a given $n \times n$ matrix, A , as follows: Let A_0 be an $(n+1) \times (n+1)$ matrix whose entries are all 1, and let $A_1 = A$. Beginning with $k = 1$

repeat the process below until the result is a scalar, the determinant of A .

Iterative process: Calculate all the 2×2 connected minors of A_k and divide each by the corresponding interior element of A_{k-1} . The resulting matrix is A_{k+1} .

A proof of Dodgson's method in the 3×3 case is simple.

Proof. Let A be the 3×3 matrix,

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix},$$

and suppose $a_5 \neq 0$. By cofactor expansion [9] down the first column we have

$$\det A = a_1 a_5 a_9 - a_1 a_6 a_8 - a_4 a_2 a_9 + a_4 a_3 a_8 + a_7 a_2 a_6 - a_7 a_3 a_5.$$

By condensation we have

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}$$

$$\rightarrow A_2 = \begin{pmatrix} a_1 a_5 - a_2 a_4 & a_2 a_6 - a_3 a_5 \\ a_4 a_8 - a_5 a_7 & a_5 a_9 - a_6 a_8 \end{pmatrix}$$

$$\rightarrow A_3 = \frac{(a_1 a_5 - a_2 a_4)(a_5 a_9 - a_6 a_8) - (a_2 a_6 - a_3 a_5)(a_4 a_8 - a_5 a_7)}{a_5}$$

$$= \frac{a_1 a_5^2 a_9 - a_1 a_5 a_6 a_8 - a_2 a_4 a_5 a_9 + a_2 a_6 a_5 a_7 + a_3 a_5 a_4 a_8 - a_3 a_5^2 a_7}{a_5}$$

$$= a_1 a_5 a_9 - a_1 a_6 a_8 - a_2 a_4 a_9 + a_2 a_6 a_7 + a_3 a_4 a_8 - a_3 a_5 a_7,$$

the determinant of A . □

When $a_5 = 0$, this method breaks down. In this case, Dodgson suggested replacing a_5 with a nonzero element of the matrix by rotating columns or rows and then proceeding with condensation. (Note that for an $n \times n$ matrix, if n is odd, then a single row or column rotation involves an even number of row or column interchanges and so does not change the sign of the determinant.) If all elements of the matrix are zero, then the matrix is trivial and its determinant is zero.

General proofs for $n \times n$ matrices are written by Xia [11] and Bressoud [3].

Clearly the iterative process cannot continue if, at any stage but the last, a zero appears as an interior element. Dodgson suggested cyclical transposition of rows or columns to remove any offending zeros. This technique cannot be applied to a condensed matrix - a matrix resulting as a step in the iterative process - as it would not yield the correct value of the determinant. One must first determine where the interior zeros arise in condensation and which elements of the original matrix produced them. Second, rotate the original rows or columns to ensure that these zeros are in the first or last rows or columns of condensed matrices.

This procedure could inadvertently introduce new zeros at some stage of iteration, since it could change several of the 2×2 connected minors. In this case, cyclical transposition should be used again. In a random matrix, it is very unlikely for any element (including those on the interior) to be zero. Hence we focus our analysis on 3×3 and 4×4 matrices constructed with random, normally distributed elements. We also use matrices contrived to void the problem altogether. Zeros are much more likely to arise in matrices with elements from a small set of integers.

Another approach to avoiding division by zero is to replace the offending element with a variable and take the limit as the variable goes to 0. This technique is implemented as follows:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 4 \\ 4 & 9 & 16 \end{pmatrix} = \lim_{x \rightarrow 0} \begin{pmatrix} 1 & 1 & 1 \\ 2 & x & 4 \\ 4 & 9 & 16 \end{pmatrix},$$

$$\begin{aligned}
A_2 &= \lim_{x \rightarrow 0} \begin{pmatrix} x-2 & 4-x \\ 18-4x & 16x-36 \end{pmatrix}, \\
A_3 &= \lim_{x \rightarrow 0} \frac{(x-2)(16x-36) - (4-x)(18-4x)}{x}, \\
&= \lim_{x \rightarrow 0} \frac{12x^2 - 34x}{x} \\
&= \lim_{x \rightarrow 0} 12x - 34 \\
&= -34.
\end{aligned}$$

This process involving algebraic simplification prior to evaluation is not appropriate for comparison to numerical computations, which are used in our determinant-calculating algorithms and do not involve any such simplification. In this paper, we do not consider this technique further.

1.2 Statement of Problem

We compare the performance of Dodgson's method to that of row reduction. Row reduction, which involves $\frac{n^3}{3} + \mathcal{O}(n^2)$ steps, is three times faster than condensation, which requires $n^3 + \mathcal{O}(n^2)$ steps [11]. Nonetheless, Dodgson's method is still of theoretical interest. We are interested in the possible numerical stability of condensation. When using floating-point arithmetic, row reduction with partial pivoting is well-known to be numerically stable [10]. That is, loss of precision or roundoff errors are not magnified by the algorithm. Dodgson's method is generally considered to be unstable, since these errors are likely to be exacerbated by iteration. We do not intend to prove or disprove this notion but make qualitative comparisons of results produced by the two algorithms.

In Chapter 2 we cover preliminary topics, and in Chapter 3 we describe the techniques and algorithms we use in our analysis. In Chapter 4 we compare algorithms using ill-conditioned matrices. In Chapters 5 and 6 we use 3×3 and 4×4 contrived and random matrices.

1.3 Literature Review

Rice and Torrence [8] give a brief introduction to determinants, focusing Dodgson's method. They provide a description of the technique and cite a proof using adjoint matrices by Bressoud [3]. As teachers of mathematics, they find condensation to be very popular among their students for evaluating large determinants, despite its potential complications when it comes to interior zeros.

Ron Pandolfi, for his undergraduate honors research project, "Doubting Dodgson's Method of Determinants" [7], illustrates how condensation can yield large percent errors when using floating-point arithmetic. He uses a matrix whose entries differ by many orders of magnitude (see Section 2.4), which we investigate in further detail. Such matrices are ill-conditioned (see Chapter 4) and are an important part of numerical linear algebra [9].

Pandolfi includes the condensation algorithm written in *Matlab*, whereas we use *Mathematica*, and so we rely heavily on Wolfram Research's "Documentation Center" [1] for syntax support.

As a masters student in 1991, Xiangsheng Xia, also under the advisement of Professor John Greene at the University of Minnesota Duluth, wrote a new proof [11] of Dodgson's method and investigated the runtime costs of the algorithm.

Chapter 2

Preliminaries

2.1 Terminology

We use a few important terms related to matrices:

- The **transpose** of a matrix is the matrix produced by replacing each element positioned in the i^{th} row and j^{th} column by that in the j^{th} row and i^{th} column.
- A **cofactor**, denoted C_{ij} , of a matrix, A , is defined as

$$C_{ij} = (-1)^{i+j} \det(A(i|j))$$

where $A(i|j)$ is the matrix obtained by deleting the i^{th} row and j^{th} column of A , [5].

- The **adjoint** of a matrix, A , denoted $\text{adj}(A)$, is the transpose of the matrix whose elements are C_{ij} , where i and j denote the row and column position of each element.

2.2 Precision

The term precision is used to describe the number of digits used in storing numbers and performing calculations. These digits are most often base 2 or base 10. Base 2 digits, called bits, are zeros or ones, and base 10 digits, called decimal digits, are the integers zero through nine. In everyday use, we typically express numbers using decimal digits,

so we express π with 7 decimal digit precision as 3.141592. The amount of precision to be used can be implicitly or explicitly specified in *Mathematica*. The default precision for any number given with a decimal point and fewer than approximately 16 decimal digits is machine precision [1]. Such a number is called a machine number or a double and is represented by our machine using 64 bits. If we input a decimal number with more than about 16 digits, we implicitly specify the use of arbitrary precision using the number of digits given, or we can explicitly set the precision using the `SetPrecision` command. To declare x to be the first 20 digits of π , we input

$$x = \text{SetPrecision}[\pi, 20].$$

This command can also be used to specify the precision of integers or rational numbers, which default to arbitrary precision.

Arbitrary precision numbers are not restricted by 64 bits of memory, so numbers can be stored and manipulated at a higher level of precision than machine numbers. Arbitrary precision numbers require more memory than machine numbers and so calculations using them can require more runtime. We use arbitrary precision to calculate a more precise value of the determinant, which is used to calculate the percent errors (see Section 3.2) in the values returned by the algorithms we compare.

Computer memory is not limitless. The largest and smallest positive arbitrary precision numbers that can be represented on our computer system [1] are

$$1.233433712981650 \times 10^{323228458} \quad \text{and} \quad 6.423382276680400 \times 10^{-323228430},$$

respectively. There are more digits stored by the computer than those displayed above, which are displayed with 16 digits. The largest and smallest positive machine numbers representable are

$$1.7976931348623157 \times 10^{308} \quad \text{and} \quad 2.225073858507201 \times 10^{-308},$$

respectively.

2.3 Floating-Point Representation Errors

Most computers render real numbers using the base 2 floating-point representation

$$\sigma \times (.a_1a_2 \dots a_t)_2 \times 2^e$$

where the sign, σ , is -1 or 1 and the a_i 's are 0 or 1 . The a_i 's are the base 2 digits - multiples of 2 required to represent the number-, where $a_1 \neq 0$. The exponent, e , is an integer, and t represents the number of bits available in the machine's memory to store the a_i 's (the mantissa). The 64 bits used to store a machine number are allotted as $t = 52$ for the mantissa, one for the sign, and 11 are the exponent. To see the relationship between a base 10 (decimal) number and its base 2 representation, consider the a_i 's as multiples of subsequent powers of two, starting with the largest, then assign the appropriate value to e so that the decimal, or radix point, precedes a_1 . As an example we have

$$\begin{aligned} 11.25 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 1 \times (1011.01)_2 \times 2^0 \\ &= 1 \times (.101101)_2 \times 2^4. \end{aligned}$$

However, this example is misleading; most real numbers, x , cannot be represented exactly this way, because most numbers cannot be represented as decimal numbers that terminate. The limited memory space of the computer requires the representation to be an approximation of x given by truncating the representation [2]. Our machine uses rounding so that if the exact value of x is

$$x_{exact} = \sigma \times (.a_1a_2 \dots a_t a_{t+1} \dots)_2 \times 2^e,$$

then

$$x_{approximate} = \begin{cases} \sigma \times (.a_1a_2 \dots a_t)_2 \times 2^e, & \text{if } a_{t+1} = 0; \\ \sigma \times [(.a_1a_2 \dots a_t)_2 + (\underbrace{.00 \dots 0}_{t-1 \text{ zeros}}1)_2] \times 2^e, & \text{if } a_{t+1} = 1. \end{cases}$$

This introduces a small truncation error, which can be propagated by arithmetic operations. These errors can also be introduced in calculations involving numbers that can be represented exactly. To see this consider the unit round, $\epsilon = 2^{-52}$, of the computer. This is the smallest positive floating-point representable number such that $1 + \epsilon > 1$, using machine precision. Thus for any number, $\delta < \epsilon$, the machine evaluates $1 + \delta$ to 1 introducing a small roundoff error of δ .

We can verify this with the floating-point operations:

$$\epsilon = 1 \times (\underbrace{.100\dots0}_2) \times 2^{-51},$$

51 zeros

so if we choose $\delta = 2^{-53}$, we have

$$\delta = 1 \times (\underbrace{.100\dots0}_2) \times 2^{-52}.$$

51 zeros

To add or subtract floating-point numbers they must have the same exponent, so since

$$1 = 1 \times (\underbrace{.100\dots0}_2) \times 2^1$$

51 zeros

we need to adjust the representations of ϵ and δ so that their exponents are also 1, so we have

$$\epsilon = 1 \times (\underbrace{.00\dots01}_2) \times 2^1$$

52 zeros

and

$$\delta = 1 \times (\underbrace{.00\dots01}_2) \times 2^1.$$

53 zeros

We then calculate $1 + \epsilon$, and, because the 53rd bit of ϵ is 1, rounding yields

$$\begin{aligned} 1 + \epsilon &= 1 \times [(\underbrace{.100\dots0}_2) + (\underbrace{.00\dots01}_2)] \times 2^1 \\ &= 1 \times (\underbrace{.100\dots01}_2) \times 2^1 \\ &> 1. \end{aligned}$$

50 zeros

However, adding 1 and δ we have

$$1 + \delta = 1 \times [(\underbrace{.100\dots0}_2) + (\underbrace{.00\dots01}_2)] \times 2^1$$

51 zeros 53 zeros

$$\begin{aligned}
&= 1 \times (.1\underbrace{00\dots0}_{{51 \text{ zeros}}})_2 \times 2^1 \\
&= 1.
\end{aligned}$$

Thus $1 + \delta$ evaluates to 1 due to truncation. Similarly the machine rounds $1 - \delta$ to 1 for $\delta < 2^{-53}$, although subtraction is handled differently and can lead to significant errors.

When subtracting nearly equal quantities, we are likely to lose a lot of precision [2] and possibly accuracy. To subtract two quantities the machine uses a method called “two’s complement” [6]. This involves switching the bits of the subtrahend and adding one to the last bit. The operation is then performed, and the last bit to carry over is dropped. To see this and how precision and accuracy can be lost consider the following: Suppose x is a real number whose exact representation consists of 51 ones and a zero followed by an infinite sequence of ones, and y is a nearly equal real number that can be represented exactly (without truncation). Then their machine precision representations are

$$x = 1 \times (. \underbrace{11\dots1}_{{52 \text{ ones}}})_2 \times 2^0$$

and

$$y = 1 \times (. \underbrace{11\dots1}_{{51 \text{ ones}}})_2 \times 2^0.$$

The machine calculates their difference using two’s complement as follows:

$$x - y = 1 \times [(\underbrace{.11\dots1}_{{52 \text{ ones}}})_2 - (\underbrace{.11\dots10}_{{51 \text{ ones}}})_2] \times 2^0,$$

Switching the bits of y we have $(\underbrace{.00\dots0}_{{51 \text{ zeros}}})_2$, and adding one to the last bit gives us $(\underbrace{.00\dots0}_{{50 \text{ zeros}}})_2$. We proceed by adding this to x and dropping the last carried bit:

$$\begin{aligned}
&= 1 \times [(\underbrace{.11\dots1}_{{52 \text{ ones}}})_2 + (\underbrace{.00\dots010}_{{50 \text{ zeros}}})_2] \times 2^0 \\
&= 1 \times (\underbrace{.00\dots01}_{{51 \text{ zeros}}})_2 \times 2^0.
\end{aligned}$$

The exponent is then adjusted so that the radix point is in its proper place, and the machine pads the trailing 51 bits with zeros:

$$= 1 \times (.100 \dots 0)_2 \times 2^{51}.$$

In this calculation we lose 51 bits of precision, and the percent error (see Section 3.2) is almost 100%. In Chapters 4, 5, and 6, we see errors arise this way in each of the algorithms we are comparing.

2.4 Order of Magnitude

Due to the need for equal exponents when adding or subtracting floating-point numbers, their orders of magnitude are very important. Commonly the order of magnitude of a number is the number of powers of 10 in it. Since the machine uses powers of two in floating-point representation of numbers, we will define it with powers of 2. The order of magnitude, \mathcal{O} , of a number x , is

$$\mathcal{O}(x) = \lfloor \log_2 x \rfloor,$$

where $\lfloor \cdot \rfloor$ is the “floor” or “greatest integer”. For example,

$$\mathcal{O}(11) = \lfloor \log_2 11 \rfloor = \lfloor 3.459 \dots \rfloor = 3.$$

For our floating-point arithmetic computations, comparisons of the orders of magnitude of the numbers involved are more important the values themselves. Consider 1, ϵ , and δ : 1 and ϵ differ in order of magnitude by 52, and 1 and δ differ in magnitude by 53. The machine can evaluate $1 + \epsilon$ correctly but not $1 + \delta$, because of this difference. Similarly it can evaluate $1 - \delta$ but not $1 - \delta/2$ because of the two’s complement method and the difference in their orders of magnitude.

Chapter 3

Methodology

3.1 Algorithms

We use *Mathematica* in our experiments to compare three algorithms: row reduction with partial pivoting, condensation, and condensation with cyclical transposition of rows or columns. A “pivot” is an element in the main diagonal (from the upper left to lower right corner) of a matrix by which other elements are divided in algorithms used in matrix operations. Partial pivoting is the process of interchanging rows of a matrix in order to establish a “good” pivot element. A “good” pivot is one with a large absolute value, since it improves numerical stability [10].

Our row reduction algorithm (see Appendix A.3) incorporates standard partial pivoting. In a given column, starting with the first, we consider the elements in or below the pivot position. The element of largest absolute value is established as the pivot by interchanging its row with the pivot row. Since each interchange switches the sign of the resulting matrix’s determinant, the number of interchanges is monitored.

We then proceed by eliminating the values below the pivot by subtracting from their row a multiple of the pivot row. The multiplier is the quotient of the element we want to eliminate and the pivot element. For an $n \times n$ matrix, this process is repeated in the first $n - 1$ columns, while keeping track of sign changes of the determinant. The result is an upper triangular matrix, and its determinant is calculated by multiplying

the main diagonal entries. For example,

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \\ 4 & 9 & 16 \end{pmatrix} \xrightarrow{\text{one interchange}} \begin{pmatrix} 4 & 9 & 16 \\ 2 & 3 & 4 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\xrightarrow[\text{row}_2 - \frac{2}{4}\text{row}_1]{\text{row}_3 - \frac{1}{4}\text{row}_1} \begin{pmatrix} 4 & 9 & 16 \\ 0 & -\frac{3}{2} & -4 \\ 0 & -\frac{5}{4} & -3 \end{pmatrix}$$

$$\xrightarrow{\text{row}_3 - \frac{-5 \times -2}{4 \times 3}\text{row}_2} \begin{pmatrix} 4 & 9 & 16 \\ 0 & -\frac{3}{2} & -4 \\ 0 & 0 & \frac{1}{3} \end{pmatrix}.$$

There is a total of one row interchange, so the determinant is the negation of the product of the diagonal elements: $-(4 \times -\frac{3}{2} \times \frac{1}{3}) = 2$. The algorithm is coded as described and includes a counter for the interchanges, and once an upper triangular form is reached, if this number is odd, the diagonal product is negated.

Our condensation algorithm (see Appendix A.1) is coded according to the iterative process described in Section 1.1. While *Mathematica* has a built-in function for calculating the minors of a matrix, we only use those that are 2×2 and connected, and so we have written a function to calculate a desired minor.

For condensation with cyclical transposition of rows or columns, we have modified the condensation algorithm differently in the 3×3 (see Appendix A.2.1) and 4×4 (see Appendix A.2.2) cases. For 3×3 matrices, analogous to the idea of partial pivoting, we rotate the element of largest absolute value to the center, since this element will be the only divisor, by rotating columns from left to right and rows top to bottom. Condensation then follows, and the sign of the determinant is not effected by row and column rotations, since the dimension of the matrix is odd, as mentioned in Section 1.1. The determinant of the above matrix is calculated according to this algorithm as follows:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \\ 4 & 9 & 16 \end{pmatrix} \xrightarrow[\text{2 column rotations}]{\text{2 row rotations}} \begin{pmatrix} 3 & 4 & 2 \\ 9 & 16 & 4 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\xrightarrow{\text{condensation begins}} \begin{pmatrix} 12 & -16 \\ -7 & 12 \end{pmatrix}$$

$$\xrightarrow{\text{the determinant is}} 2.$$

Similarly, for 4×4 matrices, we position the 2×2 connected minor of largest absolute value in the center before proceeding with condensation. This is done by determining the largest (in absolute value) of the matrix's nine 2×2 connected minors and positioning it in the center by rotating rows and columns. The rotations are again top to bottom and left to right, respectively, but, in this case, the number of rotations must be monitored. For a matrix of even dimension, each row or column rotation involves an odd number of row or column interchanges. Therefore, if an odd number of rotations are used, the sign of the determinant is switched, and so the resulting value must be negated. Consider the following example illustrating this process: For the matrix

$$V = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{pmatrix},$$

in order from left to right and top to bottom the minors are 1, 6, 26, 1, 12, 144, 1, 20, and 400, so we use cyclical transposition of rows and columns to establish 400 as the center one and proceed with condensation:

$$V \xrightarrow[\text{3 column rotations}]{\text{3 row rotations}} \begin{pmatrix} 3 & 9 & 27 & 1 \\ 4 & 16 & 64 & 1 \\ 5 & 25 & 125 & 1 \\ 2 & 4 & 8 & 1 \end{pmatrix}$$

$$\begin{aligned} &\xrightarrow{\text{condensation begins}} \begin{pmatrix} 12 & 144 & -37 \\ 20 & 400 & -61 \\ -30 & -300 & 117 \end{pmatrix} \\ &\rightarrow \begin{pmatrix} 120 & 94 \\ 240 & 228 \end{pmatrix} \\ &\xrightarrow{\text{the determinant is}} 12. \end{aligned}$$

There are an even number of row and column rotations used, so the the value produced by condensation is correct and does not need to be negated, and 12 is the resulting value.

3.2 Comparisons

We compare the algorithms by constructing a matrix with a variable element(s) and calculating the percent errors produced with each new value of the variable. Each percent error is calculated as

$$\text{Percent Error} = \left| \frac{x_e - x_a}{x_e} \right| \times 100,$$

where x_e and x_a are the exact and approximate values of the determinant of the matrix. The approximate value is that produced by the algorithm calculated using machine precision floating-point arithmetic. The exact value, which is the same for comparing all the algorithms, is value of the determinant given by the built-in *Mathematica* function calculated using arbitrary precision arithmetic. For example, solving $x^2 = 2$ with two decimal digit precision arithmetic yields $x_a = 1.4$ and $x_e = \sqrt{2} = 1.41421\dots$. That is a percent error of about 1.

To determine the exact value, the value with the highest possible precision, of the determinant of an $n \times n$ matrix, M , with entries m_{ij} , we use the `SetPrecision` command

with the Infinity option specified:

$$M_{exact} = \text{SetPrecision}[\{\{m_{11}, m_{12}, \dots, m_{1n}\}, \{m_{21}, m_{22}, \dots, m_{2n}\}, \dots, \{m_{n1}, m_{n2}, \dots, m_{nn}\}\}, \text{Infinity}].$$

The exact value is

$$\text{Det}[M_{exact}],$$

where `Det` is *Mathematica*'s built-in function. Similarly to calculate the approximate values produced by condensation and row reduction using machine precision, we define the matrix as follows:

$$M_{machine} = \text{SetPrecision}[\{\{m_{11}, m_{12}, \dots, m_{1n}\}, \{m_{21}, m_{22}, \dots, m_{2n}\}, \dots, \{m_{n1}, m_{n2}, \dots, m_{nn}\}\}, \text{MachinePrecision}].$$

The approximate values are

$$\text{Condensation}[M_{machine}, n]$$

and

$$\text{RowReduction}[M_{machine}, n],$$

where `Condensation` and `RowReduction` are the functions we have written. Lastly the percent errors are calculated, and the results are organized in a table for easy comparison. The results included are those that exemplify the data trends and allow for error comparison, as seen in later chapters.

Chapter 4

Ill-Conditioned Matrices

The condition number of an $n \times n$ matrix, A , is used to estimate the amount of precision lost when solving a linear system of n equations and n unknown variables, [10], and it is given by

$$\kappa(A) = \|A\|_{\infty} * \|A^{-1}\|_{\infty},$$

where

$$\|A\|_{\infty} = \max_{1 \leq i \leq n} \left\{ \sum_{j=1}^n |a_{ij}| \right\},$$

and a_{ij} denotes the element in the i^{th} row and j^{th} column of A . We use the infinity norm, but any norm can be used.

In general, $\log_{10}(\kappa(A))$ estimates the number of base 10 digits lost when solving the system $Ax = b$. Hence, a large condition number indicates the solution to the system is likely to involve a significant relative error. A matrix is said to be ill-conditioned if $\log_{10}(\kappa(A))$ is greater than the precision of the matrix's elements - approximately 16 decimal digits, or 52 bits. Given the relationship between this solution and the determinant of A ,

$$Ax = b \iff x = A^{-1}b = \frac{1}{\det(A)} \text{adj}(A) b,$$

we investigate briefly the relationship between a matrix's condition number and the accuracy in its determinant when calculated with machine precision. We compare the

effects of the condition number on the results produced by row reduction and condensation.

There are some matrices that are well-known to be ill-conditioned, such as Hilbert matrices, whose condition number increases exponentially with their size. An $n \times n$ Hilbert matrix, called the Hilbert matrix of order n and denoted H_n , has elements of the form $1/(i+j-1)$, where i and j denote the column and row, respectively, in which the element is positioned. For example, the Hilbert matrix of order 3 is

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix},$$

and the Hilbert matrix of order 4 is

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{pmatrix}.$$

To calculate the determinant of a Hilbert matrix, H , exactly, we use the following formula [1],

$$\det H = \frac{1}{\prod_{i=1}^{n-1} (2i+1) \binom{2i}{i}^2}.$$

Mathematica does this calculation symbolically and displays the result using any specified precision. Table 4.1 compares this value to those produced by the row reduction and condensation algorithms. Notice both algorithms lose slightly less than $\log_{10}(\kappa(H_n))$ digits of precision, and neither algorithm loses more than this many digits. Take into account the percent error adds two to ten's power in the relative error.

n	$\log_{10}(\kappa(H_n))$	Exact Value	Row Reduct. % Error	Condens. % Error
2	1.43136	8.33333×10^{-2}	1.66533×10^{-14}	1.66533×10^{-14}
3	2.8739	4.62963×10^{-4}	3.51282×10^{-13}	2.10769×10^{-13}
4	4.45294	1.65344×10^{-7}	8.86894×10^{-12}	1.49203×10^{-11}
5	5.97481	3.7493×10^{-12}	3.66269×10^{-12}	1.26083×10^{-10}
6	7.46345	5.3673×10^{-18}	1.00953×10^{-8}	5.52884×10^{-9}
7	8.99352	4.8358×10^{-25}	3.25232×10^{-7}	3.03405×10^{-7}
8	10.5299	2.73705×10^{-33}	7.91173×10^{-7}	2.51456×10^{-6}
9	12.0413	9.72023×10^{-43}	3.19076×10^{-4}	1.64511×10^{-4}
10	13.5485	2.16418×10^{-53}	1.04445×10^{-2}	6.82095×10^{-3}
11	15.0912	3.0191×10^{-65}	2.71065×10^{-1}	1.52704×10^{-1}
12	16.6144	2.63778×10^{-78}	8.35136	1.20225
13	18.122	1.4429×10^{-92}	2.08272×10^2	1.31516×10^2
14	19.6568	4.94031×10^{-108}	8.93876×10^2	1.2411×10^4
15	21.1873	1.05854×10^{-124}	2.06927×10^6	4.11647×10^6

Table 4.1: Error comparisons for the determinants of Hilbert matrices of orders 2 through 15.

Other ill-conditioned matrices are those with elements that differ by large orders of magnitude, such as

$$A_1 = \begin{pmatrix} 2^{-x} & 1 & 2^x \\ 1 & 2 & 3 \\ 2^x & 1 & 2^{-x} \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 1 + 2^x & 2^{-x} & 2^x \\ 2^{-x} & 1 + 2^x & 2^{-x} \\ 2^x & 2^{-x} & 1 + 2^x \end{pmatrix},$$

and

$$A_3 = \begin{pmatrix} 2^{-x} & 2^x & 1 \\ 1 & 2^{-x} & 1 \\ 1 & 1 & 2^{-x} \end{pmatrix}.$$

The logarithms of the condition numbers and the determinants of these matrices for increasing values of x are shown in Tables 4.2, 4.3 and 4.4, along with the results produced by each algorithm. For A_1 , row reduction, condensation, and modified condensation produce the same results, and for A_2 and A_3 , row reduction and modified condensation produce the same results.

x	$\log_2(\kappa(A_1))$	Exact Value	All Algorithm % Errors
50	49.	-2.5353×10^{30}	0.
51	50.	-1.01412×10^{31}	0.
52	51.	-4.05648×10^{31}	0.
53	52.	-1.62259×10^{32}	0.
54	53.	-6.49037×10^{32}	0.
55	54.	-2.59615×10^{33}	0.

Table 4.2: Error comparisons for the determinant of A_1 for $50 \leq x \leq 55$.

x	$\log_2(\kappa(A_2))$	Exact Value	Condensation % Error	Other Algorithm % Errors
50.	51.	2.5353×10^{30}	2.22045×10^{-13}	4.44089×10^{-14}
51.	52.	1.01412×10^{31}	1.11022×10^{-13}	2.22045×10^{-14}
52.	53.	4.05648×10^{31}	6.66134×10^{-14}	0.*
53.	54.	1.62259×10^{32}	100.	100.
54.	55.	6.49037×10^{32}	100.	100.
55.	56.	2.59615×10^{33}	100.	100.

Table 4.3: Error comparisons for the determinant of A_2 for $50 \leq x \leq 55$.

* The modified condensation algorithm produces a $2.22045 \times 10^{-14}\%$ error.

x	$\log_2(\kappa(A_3))$	Exact Value	Condensation % Error*
51	52.	2.2518×10^{15}	0.
52	53.	4.5036×10^{15}	0.
53	54.	9.0072×10^{15}	0.
54	55.	1.80144×10^{16}	100.
55	56.	3.60288×10^{16}	100.
56	57.	7.20576×10^{16}	100.

Table 4.4: Error comparisons for the determinant of A_3 for $51 \leq x \leq 56$.

* The modified condensation and row reduction algorithms produce zero percent errors.

The matrices A_1 , A_2 , and A_3 are similarly constructed and ill-conditioned and yet their results are very different from each other and from the results for Hilbert matrices. The Hilbert data suggests there is a relationship between the condition number and the accuracy of the determinant. The data for A_1 , A_2 , and A_3 does not suggest a relationship. For A_2 , if there were a relationship, we would expect to see accuracy

degrade more gradually. Condensation produces errors for A_3 not because it is ill-conditioned but because it has a small interior element (see Chapter 5). Similarly constructed to A_1 , A_2 , and A_3 are the 4×4 matrices

$$A_4 = \begin{pmatrix} 2^{-x} & 1 & 1 & 2^x \\ 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \\ 2^x & 1 & 1 & 2^{-x} \end{pmatrix},$$

$$A_5 = \begin{pmatrix} 1 + 2^x & 2^{-x} & 2^x & 2^{-x} \\ 2^{-x} & 1 + 2^x & 2^{-x} & 2^x \\ 2^x & 2^{-x} & 1 + 2^x & 2^{-x} \\ 2^{-x} & 2^x & 2^{-x} & 1 + 2^x \end{pmatrix},$$

and

$$A_6 = \begin{pmatrix} 2^{-x} & 2^x & 1 & 1 \\ 1 & 2^{-x} & 1 & 1 \\ 1 & 1 & 2^{-x} & 1 \\ 1 & 1 & 1 & 2^{-x} \end{pmatrix},$$

whose corresponding results are shown in Tables 4.5, 4.6, and 4.7. For each matrix the condensation and modified condensation algorithms produce the same results.

x	$\log_2(\kappa(A_4))$	Exact Value	Row Reduction % Error	Other Algorithm % Errors
50	52	1.26765×10^{30}	0.	$1.11022 \times 10^{-14*}$
51	53	5.0706×10^{30}	2.22045×10^{-14}	0.
52	54	2.02824×10^{31}	1.11022×10^{-14}	0.**
53	55	8.11296×10^{31}	1.11022×10^{-14}	0.**
54	56	3.24519×10^{32}	3.33067×10^{-14}	1.11022×10^{-14}
55	57	1.29807×10^{33}	0.	0.

Table 4.5: Error comparisons for the determinant of A_4 for $50 \leq x \leq 55$.

* The condensation algorithm produces a 0% error.

** The condensation algorithm produces a $1.11022 \times 10^{-14}\%$ error.

x	$\log_2(\kappa(A_5))$	Exact Value	Row Reduction % Error	Other Algorithm % Errors
50	51	5.0706×10^{30}	8.88178×10^{-14}	$4.44089 \times 10^{-13*}$
51	52	2.02824×10^{31}	4.44089×10^{-14}	$2.22045 \times 10^{-13*}$
52	53	8.11296×10^{31}	2.22045×10^{-14}	$1.11022 \times 10^{-13*}$
53	54	3.24519×10^{32}	100.	100.
54	55	1.29807×10^{33}	100.	100.
55	56	5.1923×10^{33}	100.	100.

Table 4.6: Error comparisons for the determinant of A_5 for $50 \leq x \leq 55$.

* The modified condensation algorithm produces $4.5036 \times 10^{17}\%$, $9.0072 \times 10^{17}\%$, and $1.80144 \times 10^{18}\%$ errors.

x	$\log_2(\kappa(A_6))$	Exact Value	Condensation Algorithms' % Errors**
50	51.585	-1.1259×10^{15}	$2.66454 \times 10^{-13*}$
51	52.585	-2.2518×10^{15}	$1.33227 \times 10^{-13*}$
52	53.585	-4.5036×10^{15}	$6.66134 \times 10^{-14*}$
53	54.585	-9.0072×10^{15}	$4.44089 \times 10^{-14*}$
54	55.585	-1.80144×10^{16}	200.***
55	56.585	-3.60288×10^{16}	200.***

Table 4.7: Error comparisons for the determinant of A_6 for $50 \leq x \leq 55$.

* The modified condensation algorithm produces a zero percent error.

** The row reduction algorithm produces zero percent errors.

*** The condensation algorithm produces a 100% error.

The matrices A_4 , A_5 , and A_6 are similarly constructed and ill-conditioned, and like A_1 , A_2 , and A_3 , their results are very different from each other and from the results for Hilbert matrices. That is the A_4 data does not suggest a relationship between the condition number and the accuracy of the determinant, nor does the A_5 data and the A_6 data. As in the 3×3 case, we expect the accuracy in A_5 's determinant to degrade more gradually. The errors produced by condensation for A_6 are due to the small interior elements. The errors produced by the modified algorithm for A_6 are caused by the small elements involved in the largest minor. We explain in more detail how these small interior elements cause large errors in Chapter 6.

The discrepancy in these results could be attributed to the size of the matrices or to the magnitude of their determinants. Since we focus our analysis on 3×3 and 4×4 matrices, we disregard larger matrices. We investigate the possibility of a relationship between the condition number and the accuracy in a small determinant using 3×3 and 4×4 matrices, such as

$$A_7 = \begin{pmatrix} 7 \times 2^{-5x} & 4 \times 2^{-3x} & 2 \times 2^{-4x} \\ 6 \times 2^{-7x} & 5 \times 2^{-5x} & 2^{-3x} \\ 2^{-x} & 9 \times 2^{-4x} & 6 \times 2^{-7x} \end{pmatrix}$$

and

$$A_8 = \begin{pmatrix} 2 \times 2^{-9x} & 8 \times 2^{-6x} & 6 \times 2^{-4x} & 2^{-5x} \\ 3 \times 2^{-9x} & 5 \times 2^{-7x} & 3 \times 2^{-3x} & 4 \times 2^{-8x} \\ 2^{-3x} & 7 \times 2^{-x} & 9 \times 2^{-5x} & 9 \times 2^{-8x} \\ 4 \times 2^{-8x} & 4 \times 2^{-7x} & 6 \times 2^{-7x} & 3 \times 2^{-6x} \end{pmatrix}.$$

These matrices have elements of the form $a \times 2^{-bx}$, where a and b are random integers in the interval $[1, 9]$. The data for all such matrices is similar to that in Tables 4.8 and 4.9, with the exception that small interior elements can cause standard condensation or condensation modified for 4×4 matrices to produce significant errors. Given these results, further investigation is necessary to determine if there is a relationship between the condition number and the accuracy in the determinant for larger matrices. Within the scope of our analysis, we conclude that such a relationship is not apparent in 3×3 and 4×4 matrices.

x	$\log_2(\kappa(A_7))$	Exact Value	All Algorithm % Errors
24	48.	1.06911×10^{-50}	0.
25	50.	8.35239×10^{-53}	0.
26	52.	6.5253×10^{-55}	0.
27	54.	5.09789×10^{-57}	0.
28	56.	3.98273×10^{-59}	0.
29	58.	3.11151×10^{-61}	0.
30	60.	2.43087×10^{-63}	0.

Table 4.8: Error comparisons for the determinant of A_7 for $24 \leq x \leq 30$.

x	$\log_2(\kappa(A_8))$	Exact Value	All Algorithm % Errors*
0.	3.93963	7.32×10^2	0.
1.	9.87712	5.61314×10^{-4}	0.
2.	15.7415	5.41202×10^{-9}	0.
3.	22.2648	3.99367×10^{-14}	0.
4.	29.0282	2.95844×10^{-19}	0.
5.	35.9146	2.21685×10^{-24}	0.

Table 4.9: Error comparisons for the determinant of A_8 for $0 \leq x \leq 5$.

* Some of the algorithms produce nearly zero percent errors.

Chapter 5

3 by 3 Matrices

5.1 Contrived Matrices

The crux of Dodgson's condensation method is division by zero or amplification of floating-point arithmetic errors through iteration. In the case of 3×3 matrices, we can attempt to fix both issues simultaneously. Since there is only one interior element, we simply rotate the element of largest absolute value to the interior before commencing condensation, as described in Chapter 3. Consider the matrix

$$B_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2^{-x} & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

whose determinant is $2 - 2^{1-x}$. As x becomes large, the interior element, 2^{-x} , tends to zero. Increasing the value of x causes condensation to yield larger and larger percent errors, whereas row reduction is unaffected, as shown in Table 5.1. We can see condensation performs better if modified to rotate the largest element, in this case 3, to the interior. That is, condensation is performed on the following matrix:

$$B_1^* = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 2^{-x} & 1 & 1 \end{pmatrix}.$$

x	Exact Value	Row Reduction % Error	Condensation % Error	Modified Condens. % Error
50	2.	0.	4.44089×10^{-14}	0.
51	2.	0.	2.22045×10^{-14}	0.
52	2.	1.11022×10^{-14}	1.11022×10^{-14}	1.11022×10^{-14}
53	2.	1.11022×10^{-14}	100.	1.11022×10^{-14}
54	2.	0.	300.	2.22045×10^{-14}
55	2.	0.	100.	0.
56	2.	0.	100.	0.
57	2.	0.	100.	0.

Table 5.1: Error comparisons for the determinant of B_1 for $50 \leq x \leq 57$.

Note the exact value of the determinant is not 2 for all x but is actually slightly less than 2. The exact value is stored and used in comparative calculations, but it is displayed using fewer digits, which causes the printed value to be rounded up to 2. Data in subsequent tables is also displayed this way.

More generally, for the matrix

$$B_2 = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & 2^{-x} & r_5 \\ r_6 & r_7 & r_8 \end{pmatrix},$$

where r_i is a random integer in the interval $[1, 9]$ we obtain similar results. For $r_1 = 7$, $r_2 = 9$, $r_3 = 2$, $r_4 = 7$, $r_5 = 3$, $r_6 = 8$, $r_7 = 7$, and $r_8 = 3$ we have the results in Table 5.2.

x	Exact Value	Row Reduction % Error	Condensation % Error	Modified Condens. % Error
39	-22.	1.61487×10^{-14}	1.15412×10^{-1}	9.68922×10^{-14}
40	-22.	0.	6.74716×10^{-1}	1.13041×10^{-13}
41	-22.	0.	1.5625	1.45338×10^{-13}
42	-22.	0.	2.91193	1.13041×10^{-13}
43	-22.	0.	1.5625	1.2919×10^{-13}
44	-22.	0.	10.5114	1.13041×10^{-13}
45	-22.	0.	7.38636	9.68922×10^{-14}
46	-22.	0.	28.4091	8.07435×10^{-14}
47	-22.	1.61487×10^{-14}	43.1818	8.07435×10^{-14}
48	-22.	0.	100.	8.07435×10^{-14}
49	-22.	1.61487×10^{-14}	472.727	1.77636×10^{-13}
50	-22.	0.	1045.45	1.45338×10^{-13}
51	-22.	1.61487×10^{-14}	2390.91	6.45948×10^{-14}
52	-22.	0.	100.	8.07435×10^{-14}
53	-22.	0.	100.	8.07435×10^{-14}
54	-22.	0.	100.	8.07435×10^{-14}

Table 5.2: Error comparisons for the determinant of B_2 for $39 \leq x \leq 54$.

As described in Chapter 2, subtracting nearly equal quantities introduces significant errors. Any 3×3 matrix, A , of the form

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & \epsilon & a_5 \\ a_6 & a_7 & a_8 \end{pmatrix}$$

condenses according to Dodgson's method as follows:

$$A_2 = \begin{pmatrix} a_1\epsilon - a_2a_4 & a_2a_5 - a_3\epsilon \\ a_4a_7 - a_6\epsilon & a_8\epsilon - a_5a_7 \end{pmatrix},$$

$$A_3 = \frac{(a_1\epsilon - a_2a_4)(a_8\epsilon - a_5a_7) - (a_2a_5 - a_3\epsilon)(a_4a_7 - a_6\epsilon)}{\epsilon},$$

where the subscripts here denote the $(i + 1)^{th}$ stage of the iterative process.

As ϵ gets smaller, the terms in the difference in the numerator both tend to $a_2a_4a_5a_7$, usually at different rates. Machine rounding of these values accounts for the increasing percent errors produced by condensation. Eventually ϵ becomes small enough that, due to rounding, the machine evaluates the terms in the difference to $a_2a_4a_5a_7$ causing the 100% errors.

We are interested in cases where row reduction suffers as well, so we attempt to contrive such matrices using our understanding of how floating-point arithmetic can produce errors. To do this we begin with the arbitrary 3×3 matrix

$$M = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix},$$

and consider the calculations performed by row reduction and condensation to compute the determinant. They are

$$a \left(e - \frac{bd}{a} \right) \left(\left(i - \frac{cg}{a} \right) - \frac{\left(h - \frac{bg}{a} \right) \left(f - \frac{cd}{a} \right)}{e - \frac{bd}{a}} \right)$$

and

$$\frac{(ae - bd)(ei - fh) - (bf - ce)(dh - eg)}{e},$$

respectively, where parentheses are used to specify the order of the operations. Without loss of generality we assume, $a \geq c$, $a \geq d$, and $e - \frac{db}{a} \geq h - \frac{gb}{a}$, to avoid the need for row interchanges in partial pivoting for row reduction. For simplicity we choose $a = 1$, which gives us

$$(e - bd) \left(\left(i - cg \right) - \frac{(h - bg)(f - cd)}{e - bd} \right)$$

and

$$\frac{(e - bd)(ei - fh) - (bf - ce)(dh - eg)}{e}.$$

Errors can arise in row reduction in evaluating $e - bd$, $h - gb$, $i - gc$, $f - cd$, or in evaluating the final expression, and they can arise in condensation in evaluating $e - bd$, $ei - fh$, $bf - ce$, $dh - eg$, or the final expression. We seek values for these elements for which row reduction yields errors but condensation does not. Since $e - bd$ is computed by both algorithms, we do not want to choose values such that the machine cannot correctly evaluate this expression.

First we try $a = 1$, $b = 1$, $c = 1$, $d = 1/2$, $e = 2$, $f = 1$, $g = 1 - \epsilon$, $h = 1 - \epsilon/2$, and $i = 1$, where $\epsilon = 2^{-x}$, so we have

$$B_3 = \begin{pmatrix} 1 & 1 & 1 \\ \frac{1}{2} & 2 & 1 \\ 1 - \epsilon & 1 - \frac{\epsilon}{2} & 1 \end{pmatrix},$$

whose determinant is $\frac{5}{4}\epsilon$. Row reduction calculates

$$\begin{aligned} & \frac{3}{2} \left((1 - (1 - \epsilon)) - \frac{((1 - \frac{\epsilon}{2}) - (1 - \epsilon)) (\frac{1}{2})}{\frac{3}{2}} \right) \\ &= \frac{3}{2} \left((1 - (1 - \epsilon)) - \frac{1}{3} \left((1 - \frac{\epsilon}{2}) - (1 - \epsilon) \right) \right). \end{aligned}$$

For $\epsilon = 2^{-53}$, the machine evaluates to

$$\begin{aligned} & \frac{3}{2} \left(\epsilon - \frac{1}{3} (1 - (1 - \epsilon)) \right) \\ &= \frac{3}{2} \left(\epsilon - \frac{1}{3} \epsilon \right) \\ &= \epsilon, \end{aligned}$$

yielding a 20% error, and for $0 \leq \epsilon \leq 2^{-54}$, the machine evaluates

$$\begin{aligned} & \frac{3}{2} \left((1 - (1 - \epsilon)) - \frac{1}{3} \left((1 - \frac{\epsilon}{2}) - (1 - \epsilon) \right) \right) \\ &= \frac{3}{2} \left((1 - 1) - \frac{1}{3} (1 - 1) \right) \\ &= 0, \end{aligned}$$

yielding a 100% error. Condensation yields a 20% error for $\epsilon = 2^{-51}$, 2^{-52} , and 2^{-53} by calculating

$$\begin{aligned} & \frac{\frac{3}{2}(2 - (1 - \frac{\epsilon}{2})) - (-1)(\frac{1}{2}(1 - \frac{\epsilon}{2}) - 2(1 - \epsilon))}{2} \\ &= \frac{\frac{3}{2}(2 - (1 - \frac{\epsilon}{2})) + (\frac{1}{2}(1 - \frac{\epsilon}{2}) - 2(1 - \epsilon))}{2}. \end{aligned}$$

For $\epsilon = 2^{-51}$, the machine evaluates this to

$$\begin{aligned} & \frac{\frac{3}{2}(1 + \frac{\epsilon}{2}) + ((\frac{1}{2} - \frac{\epsilon}{4}) - (2 - 2\epsilon))}{2} \\ &= \frac{(\frac{3}{2} + \frac{3}{4}\epsilon) + (-\frac{3}{2} + 2\epsilon)}{2} \\ &= \frac{3}{2}\epsilon. \end{aligned}$$

For $\epsilon = 2^{-52}$, it evaluates

$$\begin{aligned} & \frac{\frac{3}{2}(2 - (1 - \frac{\epsilon}{2})) - (-1)(\frac{1}{2}(1 - \frac{\epsilon}{2}) - 2(1 - \epsilon))}{2} \\ &= \frac{\frac{3}{2}(1) + ((\frac{1}{2} - \frac{\epsilon}{4}) - (2 - 2\epsilon))}{2} \\ &= \frac{\frac{3}{2} + (-\frac{3}{2} + 2\epsilon)}{2} \\ &= \epsilon. \end{aligned}$$

For $\epsilon = 2^{-53}$, it evaluates

$$\begin{aligned} & \frac{\frac{3}{2}(2 - (1 - \frac{\epsilon}{2})) - (-1)(\frac{1}{2}(1 - \frac{\epsilon}{2}) - 2(1 - \epsilon))}{2} \\ &= \frac{\frac{3}{2}(2 - (1)) + (\frac{1}{2}(1) - (2 - 2\epsilon))}{2} \\ &= \frac{\frac{3}{2} + (-\frac{3}{2} + 2\epsilon)}{2} \\ &= \epsilon. \end{aligned}$$

Notice that floating-point arithmetic errors occur in different stages of the computation for different values of ϵ . For all $0 \leq \epsilon \leq 2^{-54}$, using condensation accuracy degrades as follows

$$\frac{\frac{3}{2}(2 - (1 - \frac{\epsilon}{2})) - (-1)(\frac{1}{2}(1 - \frac{\epsilon}{2}) - 2(1 - \epsilon))}{2}$$

$$\begin{aligned}
&= \frac{\frac{3}{2}(2 - (1)) + (\frac{1}{2}(1) - 2(1))}{2} \\
&= \frac{\frac{3}{2}(1) + (\frac{1}{2} - 2)}{2} \\
&= 0
\end{aligned}$$

producing 100% errors. These results are in Table 5.3. Note we do not need to include the modified condensation algorithm's results, since the largest element, 2, is positioned in the center, and so the results are identical to those produced by condensation.

x	Exact Value	Row Reduction % Error	Condensation % Error
50	1.11022×10^{-15}	0.	0.
51	5.55112×10^{-16}	0.	20.
52	2.77556×10^{-16}	0.	20.
53	1.38778×10^{-16}	20.	20.
54	6.93889×10^{-17}	100.	100.
55	3.46945×10^{-17}	100.	100.
56	1.73472×10^{-17}	100.	100.

Table 5.3: Error comparisons for the determinant of B_3 for $50 \leq x \leq 56$.

We can see we have succeeded in finding a matrix on which row reduction performs poorly, however condensation does as well. We try again constructing B_4 :

$$B_4 = \begin{pmatrix} 1 & 1 - \epsilon & 1 - \epsilon \\ 1 & 2 & 1 - \frac{\epsilon}{2} \\ 1 & 1 - \frac{\epsilon}{2} & 1 - \frac{\epsilon}{2} \end{pmatrix},$$

where $\epsilon = 2^{-x}$. The results in Table 5.4 are an improvement, as we can see there are cases when condensation outperforms row reduction. For example, when $\epsilon = 2^{-53}$, the percent error produced by condensation is 0 but that produced by row reduction is 100. These errors arise in a similar fashion as they did for B_3 , because of rounding and loss

of precision errors.

x	Exact Value	Row Reduction % Error	Condensation % Error
50	4.44089×10^{-16}	0.	4.44089×10^{-14}
51	2.22045×10^{-16}	0.	2.22045×10^{-14}
52	1.11022×10^{-16}	0.	50.
53	5.55112×10^{-17}	100.	0.
54	2.77556×10^{-17}	100.	100.
55	1.38778×10^{-17}	100.	100.
56	6.93889×10^{-18}	100.	100.

Table 5.4: Error comparisons for the determinant of B_4 for $50 \leq x \leq 56$.

5.2 Random Matrices

In Section 5.1, we show there exist matrices on which one algorithm performs better than another. In this section we make a more general comparison using matrices with random elements that are normally distributed with mean 0 and standard deviation 1. We use the following sequence of commands to generate random matrices:

```
random := RandomReal[NormalDistribution[0, 1], WorkingPrecision -> 40],
Mexact = Table[random, {i, 3}, {j, 3}],
```

and

```
Mmachine = SetPrecision[Mexact, MachinePrecision].
```

The `:=` assignment operator requires the right hand side to be reevaluated each time the left hand side is called. Since real numbers' precisions default to machine precision, we set the `WorkingPrecision` option of the `RandomReal` command to 40 decimal digits to construct `Mexact` with much higher precision than `Mmachine`. Then the row reduction, condensation, and modified condensation functions are called. Their

results are compared to the value of the determinant given by

$$a_1a_5a_9 - a_1a_6a_8 - a_2a_4a_9 + a_2a_6a_7 + a_3a_4a_8 - a_3a_5a_7,$$

where a_i is the i^{th} entry of `Mexact`. We choose this formula for comparison, since it involves the fewest operations, which combined with the high arbitrary precision of `Mexact`, will produce a much better approximation to the exact value of the determinant than the other methods. The number of percent errors beyond a specified tolerance, or number of failures, are totaled with counters and used to determine the percentage of failures. The results after one million trials for each tolerance are shown in Table 5.5.

Tolerance	Row Reduction Failure %	Condensation Failure %	Modified Condensation Failure %
$2^{-39}\%$	0.2423	0.6403	0.2749
$2^{-40}\%$	0.4999	1.1503	0.5555
$2^{-41}\%$	0.9952	2.1625	1.1102
$2^{-42}\%$	1.9628	3.955	2.1676
$2^{-43}\%$	4.0069	7.2574	4.3261
$2^{-44}\%$	8.4116	13.6199	9.1078
$2^{-45}\%$	20.2651	28.0657	22.3707
$2^{-46}\%$	52.8108	59.1794	55.3559
$2^{-47}\%$	66.7824	71.7554	68.9129
$2^{-48}\%$	66.7801	71.7576	68.9179
$2^{-49}\%$	66.7637	71.7434	68.8718

Table 5.5: Failure comparisons for the determinants of 3×3 matrices with random elements.

Chapter 6

4 by 4 Matrices

6.1 Contrived Matrices

Improving Dodgson’s algorithm in the case of 4×4 matrices is not as simple as it is for 3×3 matrices. In this case, throughout condensation there are five interior elements that are used in multiple calculations, including division. We prefer them all to be as large as possible, however, rotating rows and columns to achieve the appropriate positioning of one is likely to interfere with that of another. We focus on the last divisor, which is the central or interior element after the first stage in condensation, and it is the central or interior 2×2 connected minor of the original matrix. As described in Chapter 3 we use row and column rotations to position the largest of the matrix’s 2×2 connected minors in the center before proceeding with condensation. If the original central minor is extremely small, floating-point errors can arise when using standard condensation but can be avoided with row and column rotations. Such a matrix is

$$C_1 = \begin{pmatrix} 9 & 4 & 8 & 9 \\ 8 & 1 & 1 & 9 \\ 9 & 1 + \epsilon & 1 & 3 \\ 7 & 2 & 6 & 5 \end{pmatrix},$$

where $\epsilon = 2^{-x}$. As we can see from Table 6.1, the modified condensation algorithm performs much better, and when x becomes small enough, the central minor evaluates to zero causing the standard algorithm to yield “Indeterminate” results. Row reduction

produces no significant percent errors, so its results are excluded from the table.

x	Exact Value	Condensation % Error	Modified Condensation % Error
50	-392.	2.04082	0.
51	-392.	2.04082	0.
52	-392.	14.2857	0.
53	-392.	Indeterminate	0.
54	-392.	Indeterminate	0.
55	-392.	Indeterminate	0.

Table 6.1: Error comparisons for the determinant of C_1 for $50 \leq x \leq 55$.

Standard condensation does not always produce large errors for matrices like C_1 . The matrix, C_2 , below has the same general form and yet condensation does not produce any significant errors until x becomes small enough for the central minor to evaluate to zero. In this case the modified algorithm is necessary and also doesn't produce significant errors, as seen in Table 6.2. Again, row reduction produces no significant percent errors.

$$C_2 = \begin{pmatrix} 8 & 3 & 3 & 5 \\ 5 & 1 & 1 & 6 \\ 4 & 1 + \epsilon & 1 & 4 \\ 1 & 3 & 8 & 6 \end{pmatrix}$$

x	Exact Value	Condensation % Error	Modified Condensation % Error
50	-5.	1.95399×10^{-13}	5.32907×10^{-14}
51	-5.	4.61853×10^{-13}	3.55271×10^{-14}
52	-5.	8.88178×10^{-14}	7.10543×10^{-14}
53	-5.	Indeterminate	4.79616×10^{-13}
54	-5.	Indeterminate	2.30926×10^{-13}
55	-5.	Indeterminate	1.24345×10^{-13}
56	-5.	Indeterminate	5.32907×10^{-14}
57	-5.	Indeterminate	3.55271×10^{-14}
58	-5.	Indeterminate	1.77636×10^{-14}
59	-5.	Indeterminate	0.
60	-5.	Indeterminate	0.

Table 6.2: Error comparisons for the determinant of C_2 for $50 \leq x \leq 60$.

These matrices were constructed with random integers from the interval $[1, 9]$ on the exterior and 1, 1, 1, and $1 + \epsilon$ on the interior. While the modified algorithm is an improvement, it can also produce large percent errors if the largest 2×2 connected minor of a matrix involves values much smaller or larger than the others. For example, condensation performs well and modified condensation does not on the following matrix:

$$C_3 = \begin{pmatrix} 2 & 9 & 6 & 5 \\ 9 & 1 & 4 & 9 \\ 2 & 1 & 2^x & 2^{-x} \\ 9 & 2 & 2^{-x} & 2^x \end{pmatrix}.$$

The results for increasing values of x are shown in Table 6.3. Again row reduction produces no significant percent errors.

x	Exact Value	Condensation % Error	Modified Condensation % Error
47	-1.56476×10^{30}	0.	8.49054×10^{-12}
48	-6.25902×10^{30}	1.79884×10^{-14}	4.24527×10^{-12}
49	-2.50361×10^{31}	1.79884×10^{-14}	11.3924
50	-1.00144×10^{32}	0.	21.519
51	-4.00578×10^{32}	0.	21.519
52	-1.60231×10^{33}	0.	59.4937
53	-6.40924×10^{33}	0.	102.532
54	-2.5637×10^{34}	0.	102.532

Table 6.3: Error comparisons for the determinant of C_3 for $47 \leq x \leq 54$.

6.2 Random Matrices

As in the case of 3×3 matrices, in Section 6.1, we show there exist 4×4 matrices on which one algorithm performs better than another. In this section we make a more general comparison using matrices with normally distributed random elements with mean 0 and standard deviation 1. The procedure for running one million trials for each tolerance is the same as described in Chapter 5, and the results are as follows in Table 6.4.

Tolerance	Row Reduction Failure %	Condensation Failure %	Modified Condens. Failure %
$2^{-37}\%$	0.1026	0.7683	0.4448
$2^{-38}\%$	0.2032	1.413	0.8204
$2^{-39}\%$	0.3972	2.5052	1.5097
$2^{-40}\%$	0.8026	4.4475	2.7458
$2^{-41}\%$	1.593	7.6741	4.908
$2^{-42}\%$	3.1765	13.0874	8.7789
$2^{-43}\%$	6.4452	21.7476	15.4098
$2^{-44}\%$	13.9431	36.0122	28.0361
$2^{-45}\%$	32.0889	56.2207	49.06
$2^{-46}\%$	63.6639	78.3946	74.5288
$2^{-47}\%$	75.337	85.7237	83.037
$2^{-48}\%$	75.3071	85.7164	83.1048
$2^{-49}\%$	75.2743	85.6863	83.0159

Table 6.4: Failure comparisons for the determinants of 4×4 matrices with random elements.

Chapter 7

Conclusion and Discussion

7.1 Summary

While our results for ill-conditioned matrices are interesting, they are inconclusive. The results for Hilbert matrices suggest there is a direct relationship between the condition number of a matrix and accuracy of its numerically-calculated determinant, however, the results for matrices with elements differing by several orders of magnitude do not. It is possible this relationship exists for larger matrices.

For 3×3 matrices, condensation suffers the worst loss of accuracy when the central element is much smaller than others: If it is less than a certain critical value, the algorithm produces a 100% error. The modified algorithm, which includes row and column rotations is an improvement, but it can still produce significant errors in extreme cases.

For 4×4 matrices we have a greater possibility for error and more options for improving condensation. Since after the first stage of condensation the original matrix is reduced to dimension 3×3 , we modify the algorithm to ensure the central element in the second stage is large. Again the modified algorithm is an improvement, but there are also matrices for which it produces significant errors.

For each determinant-calculating algorithm, regardless of stability, there exist matrices such that large errors are produced. Comparisons of algorithm performance are

better made using random matrices. Our results show condensation performs worse than row reduction on 3×3 and 4×4 matrices. In the case of 3×3 matrices, modified condensation performs better than condensation and almost as well as row reduction. For 4×4 matrices, our modified algorithm only performs slightly better than condensation and worse than row reduction.

We have not found evidence of instability of Dodgson's method for 3×3 and 4×4 matrices, and we have not investigated the possibility for larger matrices. Condensation performs reasonably well in general cases, except when a zero or very small number arises as an interior element during the process. There are extreme types of matrices for which condensation yields significant errors, but the same can also be said for the numerically stable row reduction.

7.2 Future Work

With additional time and experience in numerical analysis, we would conduct a formal stability analysis of the condensation algorithm and the condensation algorithm modified for 3×3 matrices. Such work could indicate other ways to improve algorithm performance and stability, if it is unstable. There are certainly other possibilities for improving condensation in the case of 4×4 and larger matrices. One could investigate a possible general scheme for improving the stability of condensation for matrices of any size. Regardless of the stability of the condensation or modified condensation algorithms, it would be interesting to determine whether there exist families of matrices for which one or more of these algorithms consistently produces more accurate results than row reduction.

An explanation for the trends in the results for random matrices would also be interesting. The significant change in failure percentages for each new tolerance and the leveling off of these values after a certain tolerance are most likely related to the machine's capacity to represent the results.

Beyond our brief investigation, further exploration is needed to determine the relationship, if any, between the condition number of a matrix and its numerically-calculated determinant.

References

- [1] C.L. Dodgson. Condensation of Determinants, Being a New and Brief Method for Computing their Arithmetical Values. *Proceedings of the Royal Society of London*, 15:150–155, 1866.
- [2] G. Strang. *Introduction to Linear Algebra*. Wellesley Cambridge Pr, 2003.
- [3] Xiangsheng Xia. The Analysis of a Nonstandard Method for Calculating Determinants. Master’s Thesis, 1991.
- [4] David M. Bressoud. *Proofs and Confirmations: The Story of the Alternating Sign Matrix Conjecture*. Mathematical Association of America, 1999.
- [5] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial Mathematics, 1997.
- [6] A. Rice and E. Torrence. Lewis Carroll’s Condensation Method for Evaluating Determinants. *Math Horizons*, pages 12–15, 2006.
- [7] Ron Pandolfi. Doubting Dodgson’s Method of Determinants. Undergraduate Honors Project, 2008.
- [8] Wolfram mathematica documentation center. <http://reference.wolfram.com/mathematica/guide/Mathematica.html>, 2011.
- [9] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice-Hall, NJ, 1971.
- [10] K. E. Atkinson. *An Introduction to Numerical Analysis*. Wiley-India, 2009.
- [11] I. Koren. *Computer Arithmetic Algorithms*. AK Peters, Ltd., 2002.

Appendix A

Mathematica Code

A.1 Condensation

We define a function to calculate the 2×2 connected minor whose first element is in position $\{i, j\}$, that is in row i and column j .

```
Minor[M_, {i_Integer, j_Integer}]:=
  Block[{minor = Take[M, {i, i+1}, {j, j+1}]},
    minor[[1, 1]]*minor[[2, 2]]-minor[[1, 2]]*minor[[2, 1]]
```

Before beginning the iterative process (see Section 1.1), we locally declare the variable k and the arrays A_{next} , A_0 , and A_i . Iteration begins by calculating A_{next} and adding to the list of matrices, A_i , involved in condensation. Lastly matrices in this list are displayed in `MatrixForm` (versus nested lists). The last matrix in the list is a 1×1 matrix, or a scalar, the determinant of the original $n \times n$ matrix, M .

```
Condensation[M_, n_]:=
  Block[{k, Anext, A0 = Table[1, {i, n+1}, {j, n+1}], Ai = {}},
    {AppendTo[Ai, A0];
     AppendTo[Ai, M];
     For[k = 3, k ≤ n+1, k++,
       Anext = Table[1/((Ai[[k-2]])[[i+1, j+1]])*Minor[Ai[[k-1]], {i, j}],
```

```

    {i, n-(k-2)}, {j, n-(k-2)}, AppendTo[Ai, Anext]}}];
Table[MatrixForm[Ai[[i]]], {i, 2, n+1}]]]

```

A.2 Modified Condensation

A.2.1 3 by 3 Matrices

The modified condensation function modified for 3×3 matrices rotates rows and columns so as to place the largest entry (in absolute value) in the (2, 2) position. It then applies the standard condensation algorithm.

```

CondensationMod3x3[M_]:=
Block[{k, Anext, Amod, A0 = Table[1, {i, n+1}, {j, n+1}], Ai = {}},
{Amod = Catch[For[i = 1, i ≤ n, i++,
For[j = 1, j ≤ n, j++,
If[MemberQ[Position[Abs[M], Abs[Max[M]]], {i, j}],
Throw[RotateRight[M, {2-i, 2-j}]]]]]]];
AppendTo[Ai, A0];
AppendTo[Ai, Amod];
For[k = 3, k ≤ n+1, k++,
{Anext = Table[1/((Ai[[k-2]])[[i+1, j+1]])*Minor[Ai[[k-1]],
{i, j}], {i, n-(k-2)}, {j, n-(k-2)}, AppendTo[Ai, Anext]}}];
Table[MatrixForm[Ai[[i]]], {i, 2, n+1}]]]

```

A.2.2 4 by 4 Matrices

To modify condensation for 4×4 matrices, we use the same code structure as for 3×3 matrices and include the variable `sign`. This keeps track of the sign of the determinant as a result of row and column rotations of `M`, which are used to position the 2×2 connected minor of largest absolute value in the center. The resulting matrix is `Amod`. Finally the sign of the determinant, the last item in the list `Ai`, is adjusted, and the matrices at each stage of condensation are printed.

```

CondensationMod4x4[M_]:=
Block[{k, Anext, Amod, sign = 1, A0 = Table[1, {i, n+1}, {j, n+1}],
  Ai = {}, A1 = Abs[Table[Minor[M, {i, j}], {i, 3}, {j, 3}]]},
  {{sign, Amod} = Catch[For[i = 1, i ≤ 3, i++,
    For[j = 1, j ≤ 3, j++,
      If[MemberQ[Position[A1, Max[A1]], {i, j}],
        Throw[{-1}^(i+j), RotateRight[M, {2-i, 2-j}]]];
    AppendTo[Ai, A0];
    AppendTo[Ai, Amod];
    For[k = 3, k ≤ n+1, k++,
      {Anext = Table[1/((Ai[[k-2]])[[i+1, j+1]])*Minor[Ai[[k-1]],
        {i, j}], {i, n-(k-2)}, {j, n-(k-2)}], AppendTo[Ai, Anext]}}];
  Ai[[n+1, 1, 1]] = sign*Ai[[n+1, 1, 1]];
  Table[MatrixForm[Ai[[i]]], {i, 2, n+1}]]]

```

A.3 Row Reduction

We have two functions to define the columns and rows of an $n \times n$ matrix, M , as `col[i, M]` and `row[i, M]`, respectively, for $1 \leq i \leq n$.

```

DefineColumns[M_, n_] := For[i = 1, i ≤ n, i++,
  col[i, M] = Table[M[[k, i]], {k, n}]

```

```

DefineRows[M_, n_] := For[i=1, i ≤ n, i++, row[i, M] = M[[i]]

```

Before beginning the process of row reduction, we locally declare the matrix to be row reduced, the number of row **interchanges**, the **pivot** in a given column, and its **position** in the matrix. Row reduction is then implemented as described in Section 3.1, and the value of the determinant is returned.

```

RowReduction[M_, n_] :=
Block[{Mrr = M, interchanges = 0, pivot, position},
  {For[c = 1, c ≤ n, c++,

```

```

{DefineColumns[Mrr, n],
 DefineRows[Mrr, n],
 If[c ≠ n, {pivot = Position[Abs[Drop[col[c, Mrr], c-1]],
   Max[Abs[Dropcol[c, Mrr], c-1]]}][[1, 1]];
  position = c-1+pivot;
  If[pivot ≠ 1, {Mrr = ReplacePart[Mrr, {c → row[position, Mrr],
    position → row[c, Mrr]}], interchanges += 1}]}],
DefineColumns[Mrr, n],
DefineRows[Mrr, n],
For[r = 1, r ≤ n, r++,
  If[c < r && Mrr[[c, c]] ≠ 0,
    row[r, Mrr] =row[r, Mrr]-
      (1/(Mrr[[c, c]))*((row[r, Mrr])[c])*row[c, Mrr]],
  Mrr = Table[row[i, Mrr], {i, 1, n}];];
If[OddQ[interchanges],-1*Product[Mrr[[i, i]], {i, n}],
  Product[Mrr[[i, i]], {i, n}]]]

```