Finding Determinants Through Programming

Wyatt Andresen

1 Review of Related Topics

1.1 Determinants

Definition. The determinant is a value that can be found from any square matrix, and is denoted as $det(A)$.

1.1.1 2×2 and 3×3 Matrices

Two important determinants are for 2×2 and 3×3 matrices.
Suppose that:

$$A_{2\times 2} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \text{and} \quad A_{3\times 3} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}.$$

Then,

$$det(A_{2\times 2}) = ab - cd \quad \text{and}$$
$$det(A_{3\times 3}) = (aei + bfg + cdh) - (gec + hfa + idb).$$

1.1.2 Larger Matrices and Laplace Expansion

Determinants larger than this are less simple to find. In these cases, Laplace expansion is a method that may be used to find the determinant.
Suppose that:

$$A_{n\times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}.$$

Laplace expansion may begin on any row or column in the matrix. Supposing we began with the column of a_{11}, the first expansion would produce:

$$det(A_{n\times n}) = \sum_{i=1}^{n} a_{i1} \times (-1)^{i+1} \times det(A'_{n\times n})$$

where a_{i1} is an entry in the column and $A'_{n\times n}$ is the matrix resulting from the removal of that row and column. The determinant of $A'_{n\times n}$ may then also be found, either through continued Laplace expansions or, if it is 2×2 or 3×3, the associated equations.

Example. We will find the determinant of a 3×3 matrix via Laplace expansion.
Suppose that:

$$A_{3\times 3} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Then, $det(A_{3\times 3})$

$$\begin{aligned}
&= \sum_{i=1}^{3} a_{i1} \times (-1)^{i+1} \times det(A'_{2\times 2}) \\
&= a_{11} \times (-1)^{1+1} \times det(A'_{2\times 2}) + a_{21} \times (-1)^{2+1} \times det(A'_{2\times 2}) + a_{31} \times (-1)^{3+1} \times det(A'_{2\times 2}) \\
&= 1 \times 1 \times det(\begin{bmatrix} 5 & 8 \\ 6 & 9 \end{bmatrix}) + 2 \times -1 \times det(\begin{bmatrix} 4 & 7 \\ 6 & 9 \end{bmatrix}) + 3 \times 1 \times det(\begin{bmatrix} 4 & 7 \\ 5 & 8 \end{bmatrix}) \\
&= 1 \times 1 \times ((5)(9) - (8)(6)) + 2 \times -1 \times ((4)(9) - (7)(6)) + 3 \times 1 \times ((4)(8) - (7)(5)) \\
&= (-3) - 2 \times (-6) + 3 \times (-3) \\
&= 0
\end{aligned}$$

1.1.3 Determinants of Triagangular Matrices

Definition. A triangular matrix is a matrix that is all 0s below the diagonal. The determinant of a triangular matrix is always the product of the elements of the diagonal.

Example. We will find the determinant of a triangular 4×4 matrix via Laplace expansion.
Suppose that:

$$A_{4\times 4} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

The zeros in column one will cause the Laplace expansion for column one to only include the first term in the expansion. So, $det(A_{4\times 4})$

$$= 1 \times (-1)^{1+1} \times det(\begin{bmatrix} 5 & 6 & 7 \\ 0 & 8 & 9 \\ 0 & 0 & 10 \end{bmatrix}) + 0\text{s}$$

The zeroes below column one in our new matrix will cause the same thing to occur. Thus:

$$det(\begin{bmatrix} 5 & 6 & 7 \\ 0 & 8 & 9 \\ 0 & 0 & 10 \end{bmatrix}) = 5 \times (-1)^{1+1} \times det(\begin{bmatrix} 8 & 9 \\ 0 & 10 \end{bmatrix}) + 0\text{s}$$

And the determinant of our final matrix is:

$$det(\begin{bmatrix} 8 & 9 \\ 0 & 10 \end{bmatrix}) = (8)(10) - (9)(0) = (8)(10)$$

Now, if we put it all together, $det(A_{4\times 4})$

$$= 1 \times 1 \times 5 \times 1 \times 8 \times 10$$

or,

$$= 1 \times 5 \times 8 \times 10$$

which is the same as our original diagonal.

1.1.4 Properties of Determinants

Multiplication The determinant of a product is the product of the determinants:

$$det(A.B) = det(A) \times det(B)$$

Inverses The product of the determinants of a matrix and its inverse is the determinant of the identity matrix. Because the identity matrix is a triangular matrix with all ones along its diagonal, we know that $det(I) = 1$. A result of this fact is that if A^{-1} exists, then $det(A) \neq 0$.

$$det(A) \times det(A^{-1}) = det(AA^{-1}) = det(I) = 1$$

Row Operations and Matrix Transformation

Row Replacement / Shear The shear matrix below performs a row replacement. Row replacements, and thus shears, maintain the determinant of the matrix (area/volume).

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \xrightarrow[\begin{bmatrix}1 & 0\\ -3 & 1\end{bmatrix}]{r_2'=-3r_1+r_2} \begin{bmatrix}1 & 2\\ 0 & -2\end{bmatrix}$$

$$det(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}) = -2 \quad \text{and} \quad det(\begin{bmatrix}1 & 2\\ 0 & -2\end{bmatrix}) = -2$$

Scaling / Stretching The stretch matrix below performs a scale on row one. Scaling and stretching do not maintain the determinant, but multiply it by the scaling factor.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \xrightarrow[\begin{bmatrix}3 & 0\\ 0 & 1\end{bmatrix}]{r_1'=3r_1} \begin{bmatrix}3 & 6\\ 3 & 4\end{bmatrix}$$

$$det(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}) = -2 \quad \text{and} \quad det(\begin{bmatrix}3 & 6\\ 3 & 4\end{bmatrix}) = -6$$

Interchanging / Reflecting The reflection below causes row one and two to interchange. The magnitude of the determinant is preserved, but the sign is flipped.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \xrightarrow[\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}]{r_1\leftrightarrow r_2} \begin{bmatrix}3 & 4\\ 1 & 2\end{bmatrix}$$

$$det(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}) = -2 \quad \text{and} \quad det(\begin{bmatrix}3 & 4\\ 1 & 2\end{bmatrix}) = 2$$

2 Extension

The following section showcases the creation of a Java program capable of computing the determinant of any square matrix.

2.1 Motivation

Electronic mathematics tools like Maple have slowly become more and more useful, particularly as I have progressed into and learned of more lengthy computations. Due to my dual interest in mathematics and computer science, it was natural for me to want to see how I could combine the two together myself. The recursive nature of finding determinants through Laplace expansion made for what I perceived would be an interesting program to create.

2.2 Implementation

The previously mentioned recursive nature of determinants made it natural to aim for a recursive implementation with the program. A looping implementation was a possibility, but the recursive implementation is more representative of determinant calculation and more intuitive.

The code is commented to provide clear indications of each step that is performed in the calculation. Overarchingly, the code is split into four methods; main which gets the ball rolling, findDeterminant which makes most of the magic happen, and two helper methods deleteRowAndColumn and printDeterminant. deleteRowAndColumn generates and returns the matrix after the passed column and row are deleted. printDeterminant simply prints the output in an easily readable format.

2.3 Java Code

```
//* Determinant.java - finds the determinant
 * of a matrix of pretty much any size.
 * Simply enter the (square) matrix into
 * the matrix array below.
 *
 * @author Wyatt Andresen
 * @version 6/26/17
 *
 */

import java.lang.*;

public class Determinant
{
    /* main - Declares the matrix whose determinant
     * will be found, obtains the determinant by
     * calling the findDeterminant method, and then
     * prints the determinant by calling the
     * printDeterminant method
     *
     * @param: args - command line args
     */
```

```
public static void main(String[] args)
{
    //Defines some example matrices
    //whose determinants may be found
    //Must be square
    int[][] matrix2x2 = {{1,2},
                         {3,4}};

    int[][] matrix3x3 = {{1,4,7},
                         {2,5,8},
                         {3,6,9}};

    int[][] matrix4x4 = {{1,2,3,4},
                         {0,5,6,7},
                         {0,0,8,9},
                         {0,0,0,10}};

    int[][] matrix5x5 = {{5,2,0,0,-2},
                         {0,1,4,3,2},
                         {0,0,2,6,3},
                         {0,0,3,4,1},
                         {0,0,0,0,2}};

    //Finds and prints the determinant of
    //the 2x2 matrix
    int determinant = findDeterminant(matrix2x2);
    printDeterminant(matrix2x2, determinant);

    //Finds and prints the determinant of
    //the 3x3 matrix
    determinant = findDeterminant(matrix3x3);
    printDeterminant(matrix3x3, determinant);

    //Finds and prints the determinant of
    //the 4x4 matrix
    determinant = findDeterminant(matrix4x4);
    printDeterminant(matrix4x4, determinant);

    //Finds and prints the determinant of
    //the 5x5 matrix
    determinant = findDeterminant(matrix5x5);
    printDeterminant(matrix5x5, determinant);
}

/* findDeterminant - finds and returns the
 * determinant of a matrix
 *
 * @param: matrix - the matrix whose determinant
 * will be found
```

```
 * @return: the determinant of the matrix
 */
public static int findDeterminant(int[][] matrix)
{
    //Initializes size which holds the
    //length of the matrix
    int size = matrix.length;

    //Initializes determinant to 0
    int determinant = 0;

    //Perform Laplace expansion on the
    //matrix to find the determinant
    for (int i = 0; i < size; i++)
    {
        //If size is less than two,
        //continue to expand
        if (size > 2)
        {
            //Obtains the matrix whose
            //determinent will need to
            //be found next
            int[][] nextMatrix = deleteRowAndColumn(i, 0, matrix);

            //Laplace expansion
            determinant += matrix[i][0] * Math.pow(-1, i)
                           * findDeterminant(nextMatrix);
        }
        //If size is equal to two,
        //find the determinant
        else if (size == 2)
        {
            determinant = matrix[0][0] * matrix[1][1]
                          - matrix[0][1] * matrix[1][0];
        }
    }
    return determinant;
}

/* deleteRowAndColumn - deletes the passed
 * row and column.
 *
 * @param: row - the row to be deleted
 * @param: column - the column to be deleted
 * @return: the new matrix with the passed in
 * row and column deleted
 */
public static int[][] deleteRowAndColumn(int row, int column,
                                         int[][] matrix)
```

```
{
    int oldSize = matrix.length;
    int newSize = matrix.length - 1;

    //Declares a temporary matrix to be used
    //to delete the correct row
    int[][] tempMatrix = new int[newSize][oldSize];
    //Adds the rows before the row to be deleted
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < oldSize; j++)
        {
            tempMatrix[i][j] = matrix[i][j];
        }
    }
    //Adds the rows after the row to be deleted
    for (int i = row; i < newSize; i++)
    {
        for (int j = 0; j < oldSize; j++)
        {
            tempMatrix[i][j] = matrix[i+1][j];
        }
    }

    //Declares the now correctly sized matrix
    //that will be returned
    //The column deletion will be performed
    //on this matrix
    int[][] newMatrix = new int[newSize][newSize];
    //Adds the columns before the column to be deleted
    for (int i  = 0; i < newSize; i++)
    {
        for (int j = 0; j < column; j++)
        {
            newMatrix[i][j] = tempMatrix[i][j];
        }
    }
    //Adds the columns after the column to be deleted
    for (int i  = 0; i < newSize; i++)
    {
        for (int j = column; j < newSize; j++)
        {
            newMatrix[i][j] = tempMatrix[i][j+1];
        }
    }

    //Testing code that prints every returned matrix
    /*for (int i = 0; i < newMatrix.length; i++)
    {
```

```
        for (int j = 0; j < newMatrix.length; j++)
        {
            System.out.print(newMatrix[i][j] + " ");
        }
        System.out.print("\n");
    }
    System.out.println();*/

    return newMatrix;
}

/* printDeterminant - prints the determinant
 *
 * @param: matrix - the matrix whose
 * determinant was found
 * @param: determinant - the determinant of
 * the matrix
 */
public static void printDeterminant(int[][] matrix, int determinant)
{
    //Initialization of size, which holds
    //the size of the matrix
    int size = matrix.length;

    //Start of the print determinant string
    String print = "The determinant of the " + size + "x" + size
                    + " matrix: ";

    //Declaration of numSpaces (which holds
    //the length of the previous line) and
    //spaces (which is filled with a blank
    //space equal to the size of the previous
    //line
    int numSpaces = print.length();
    String spaces = "";
    for (int i = 0; i < numSpaces; i++)
    {
        spaces += " ";
    }

    //Adds the first row of the matrix
    for (int i = 0; i < matrix.length; i++)
    {
        print += matrix[0][i] + " ";
    }

    //Adds the determinant statment
    print += "is: " + determinant + "\n" + spaces;
```

```
        //Adds the remaining rows of the matrix
        for (int i = 1; i < matrix.length; i++)
        {
            for (int j = 0; j < matrix.length; j++)
            {
                print += matrix[i][j] + " ";
            }
            print += "\n" + spaces;
        }

        //Prints the whole
        System.out.println(print);
    }
}
```

2.4 Output

The below output has been copied directly from the terminal. Additionally, some of these can be compared to the determinants we computed by hand in Section 1.

```
The determinant of the 2x2 matrix: 1 2 is: -2
                                   3 4

The determinant of the 3x3 matrix: 1 4 7 is: 0
                                   2 5 8
                                   3 6 9

The determinant of the 4x4 matrix: 1 2 3 4 is: 400
                                   0 5 6 7
                                   0 0 8 9
                                   0 0 0 10

The determinant of the 5x5 matrix: 5 2 0 0 -2 is: -100
                                   0 1 4 3 2
                                   0 0 2 6 3
                                   0 0 3 4 1
                                   0 0 0 0 2
```

2.5 Conclusion

Overall, this project has enforced and broadened my understanding of determinants and their calculation. The recursive nature of their calculation is of particular interest to me, and made for an interesting, challenging coding project that caused me to more deeply investigate and understand them.