PROC. INTERNAT. CONF. SCI. ENGIN.
Volume 2, 2019 | Pages: 245-248

ISSN 2597-5250
EISSN 2598-232X

The Implementation of Recursive Algorithm to Determine the Determinant of n x n Matrix Using Cofactor Expansion

Rahmat Hidayat[*], Seto Rahardyanto[**], Pahlevi Wahyu Hardjita[***]
Informatics Engineering Department, Faculty of Science and Technology, UIN Sunan Kalijaga Yogyakarta
Jl. Marsda Adisucipto No. 1 Yogyakarta 55281, Indonesia. Tel. +62-274-540971, Fax. +62-274-519739
Email: rahmat.hidayat@uin-suka.ac.id[*], [2]17106050014@student.uin-suka.ac.id[**], [3]17106050009@student.uin-suka.ac.id[***]

Abstract. This research examines the algorithm to determine the Determinant of n x n matrix using cofactor expansion and the implementation using recursive algorithm. Random function is added as an option to generate large matrix. To identified singular matrix and to avoid unnecessary further process, the algorithm searches for rows that are multiples of integer from another row, or columns that are multiples of integer from another column, then returns 0 as a Determinant value. In order to minimize the number of iterations and computations, the algorithm searches for rows or columns having the highest number of zero elements to be expanded. The program will not expand zero element and immediately returns a value of zero for those cofactor. Finally the program calculates the number of iterations performed each time the cofactor expands to calculate the Determinant of 2 x 2 matrix.In conclusion, the data shows that the use of recursice algorithm to iteratively expand cofactor considering the row and column having highest number of zero, will reduce the number of iteration and computation.

Keywords: Algorithm, Cofactor expansion, Determinant, Recursive

INTRODUCTION

Mathematics has a close relationship with informatics. In the academic text (Naskah Akademik) of the Indonesian National Qualification Framework For Informatic And Computer Cluster In Aptikom V Region (Kerangka Kualifikasi Nasional Indonesia untuk Rumpun Informatika dan Komputer di Aptikom Wilayah V), there are topics in Mathematics and Statistics with a field studies of discrete structure and computational science. Courses in the curriculum 2016 of the Informatics Engineering program in Sunan Kalijaga State Islamic University, Yogyakarta, related to those topics area are Discrete Mathematic, Calculus, Numerical Methods, Statistics & Probability and Linear Algebra. One of the material learned in Linear Algebra course is about determinants.

In addition to the topic area of Mathematics and Statistics, there are also topics in the area of Algorithms and Programming, with one of the related course namely the Algorithm & Programming in which a recursive function studied. This paper intends to collaborate the use of recursive algorithms to find determinants with orders of more than three using cofactor expansion, implemented using Python language programming. As the results, Informatics Engineering, Mathematics or Mathematics Education students can add their learning material.

Determinant, for more than hundreds of years, has many significant roles in a several of mathematical areas such as geometry, differential equations including theory of equations and matrix algebra. Seki Kowa (1642–1708) first discovered the idea of 2 x 2 determinant in 1683 while to denote it, Arthur Cayley (1821–1895) a British mathematician first introduced the notation of two vertical lines on either side of the array that remains the standard to date (Debnath, 2013). In Europe, the famous German mathematician, Gottfried Wilhelm Leibniz (1646 – 1716) separately invent the same idea of determinant. His results was used by Gabriel Cramer (1704–1752), a Swiss mathematician, for solving linear equations in terms of determinants which known as cramer's rule. Pierre Simon Laplace (1749–1827), a famous French mathematician, in 1772 proved the expansion of determinant of order n in terms of minors or cofactors along the ith row, known as the Laplace Expansion Theorem. Carl Friedrich Gauss (1777 – 1855) German Mathematician first used the properties of determinant in his Disquisitiones Arithmeticae. (Gauss, 1965). Afterward, French Mathematicians, Augustin Louis Cauchy contribute to the theory of determinants in the context of quadratic forms in n variables (Cauchy, 1821).

Recursive function theory can be traced to its origin since around 1931 about primitive recursive (Kleene, 1981). In 1932, Rozsa Peter presented Rekursive Funktionen, followed by Herbrand-Godel introduced term 'general recursiveness'. In 1936, Turing notion of 'computability' of recursive function. In its development, the recursion function and its variance are used in many cases, such as to compute the discrete cosine transform (Hou, 1987), for solving a class of non-linear matrix equations (Yan, B.Moore, & Helmke, 1994), to adaptive CFAR detection (Conte, De Maio, & Ricci, 2002), to construct Unitary and symplectic group representations (Baclawski, 1982), and many others cases in various fields of study.

Thus, this essay will discuss the implementation of recursive algorithm to determine the determinant of n x

n matrix using cofactor/laplace expansion in Python language programming.

MATERIALS AND METHODS

Matrix and Determinant

Simply, a matrix can be defined as a rectangular array of numbers. Nevertheless, matrix occurs in any contexts. For instance, the following rectangular array with four rows and five columns might describe the number of minutes that a lecturer spent times during an office hour in campus, it can be seen in the table 1.

After eliminate the headings, then we are given the following rectangular array of numbers with four rows and five columns, called a 'matrix'.

$$\begin{bmatrix} 120 & 0 & 60 & 100 & 30 \\ 30 & 150 & 30 & 45 & 0 \\ 10 & 30 & 0 & 15 & 120 \\ 180 & 200 & 150 & 60 & 0 \end{bmatrix}$$

Determinant is certain function that associates a real number with a square matrix. One of the properties of determinant is that the value of a determinant is zero if any two rows (or columns) are identical.

Table 1. The number of minutes one lecturer spent on campus during office hour.

	Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
Teaching	120	0	60	100	30
Research	30	150	30	45	0
Community Service	10	30	0	15	120
Administrative	180	200	150	60	0

Determinants by Cofactor Expansion

Definition: If ^{A} is a square matrix, then the minor of entry $\mathbf{a_{ij}}$ is denoted by $\mathbf{M_{ij}}$ and is defined to be the determinant of the submatrix that remains after the **i**th row and **j**th column are deleted from ^{A}. The number $(-1)^{(i+j)}$ Mij is denoted by $\mathbf{C_{ij}}$ and is called the cofactor of entry $\mathbf{a_{ij}}$. The definition of a n x n determinant in terms of minors and cofactors is

$$\det(A) = a_{1j}c_{1j}+a_{2j}c_{2j}+.....+a_{nj}c_{nj}$$

This method of evaluating det *(A)* is called cofactor expansion along the **jth** column of ^{A} And

$$\det(A) = a_{i1}c_{il}+a_{i2}c_{i2}+.....+a_{in}c_{in}$$

as cofactor expansion along the **ith** row.

Smart choice of using which row or column are used to simplifier the calculation as well as to ease the number of iteration. The row or column containing most zero is suggested as the basis of finding the determinant (Anton & Rorres, 2011).

Recursive Algorithm

Recursion is a method to solve a problem where the solution use the solution resulting from smaller instances of the same problem (compared to iteration) (Graham, Knuth, Patashnik, & Liu, 1989). Many computer programming languages, allows a function to call itself as a form of recursive code. Based on the number self-reference contained, recursive algorithm can be divided into single recursion or multiple recursion. Recursion also can be categorized into direct recursion and indirect recursion, depends on the way the function is called, whether it referred by its function, or by other function that called by the function. Several examples of recursive programs are the program to calculate the factorial of a natural number, to computes the greatest common divisor of two integers. Three important laws must be obeyed by Recursive algorithm :

1. A recursive algorithm must have a **base case**.
2. A recursive algorithm must change its state and move toward the base case.
3. A recursive algorithm must call itself, recursively.

In order to compare the number of iterations needs to find the determinant, we arrange three algorithms :

1. program1 using cofactor expansion without considering the row or column containing zero nor whether there is row or column multiple integer other row or column
2. program2 using cofactor expansion and analyze whether there is row or column multiple integer other row or column without considering the row or column containing zero
3. program3 using cofactor expansion, analyze whether there is row or column multiple integer other row or column and considering the row or column containing zero

Pseudocode *program1*

```
function hitungDeterminan:
input: matrix, jumlah
  iterasi = iterasi + 1
  1. if sizeMatrix=1 , return jumlah * mattrix[0][0]
  2. otherwise,
    for i in range(lebar):
      bantu1 = []
      for j in range(1, lebar):
        bantu2 = []
        for k in range(lebar):
          if k != i:
            bantu2.append(mat[j][k])
            angka += 1
        bantu1.append(bantu2)
      total += jumlah * hitungDeterminan(bantu1, kof * mat[0][i])
      kof *= -1
    return total
end hitungDeterminan
```

Pseudocode program 2

```
function hitungDeterminan:
input: matrix, jumlah
  iterasi = iterasi + 1
  1. if sizeMatrix=1 , total= jumlah * mattrix[0][0]
  2. otherwise,
     if there is a row or column, multiple integer other, total = 0
     else for i in range(lebar):
        bantu1 = []
        for j in range(1, lebar):
           bantu2 = []
           for k in range(lebar):
              if k != i:
                 bantu2.append(mat[j][k])
                 angka += 1
           bantu1.append(bantu2)
        total += jumlah * hitungDeterminan(bantu1, kof * mat[0][i])
        kof *= -1
     return total
end hitungDeterminan
```

Pseudocode *program 3*

```
function hitungDeterminan:
input: matrix, jumlah
  iterasi = iterasi + 1
  1. if sizeMatrix=1 , total= jumlah * mattrix[0][0]
  2. otherwise,
     if there is a row or column, multiple integer other, total = 0
     finding row or column containing most zero as basis
     for i in range(lebar):
        bantu1 = []
        for j in range(1, lebar):
           bantu2 = []
           for k in range(lebar):
              if k != i:
                 bantu2.append(mat[j][k])
                 angka += 1
           bantu1.append(bantu2)
        total += jumlah * hitungDeterminan(bantu1, kof * mat[0][i])
        kof *= -1
     return total
end hitungDeterminan
```

To compare program 1 and program2, samples matrix with size 5 to 10 containing row multiple integer other row are used.

Table 2. Sample matrix used to compare the number iterations program1 and program 2.

Ordo 5	[[4,3,2,5,0],[3,2,0,1,2],[8,6,4,10,0],[1,2,3,2,4],[2,4,7,4,1]]
Ordo 6	[[4,3,2,5,0,7],[3,2,0,1,2,2],[5,1,3,5,2,5],[1,2,3,4,2,4],[6,4,0,2,4,4],[1,6,3,1,4,2]]
Ordo 7	[[4,2,3,2,5,0,7],[3,7,2,0,1,2,2],[5,1,1,3,5,2,5],[1,7,2,3,4,2,4],[2,8,4,7,4,1,3], [2,14,4,6,8,4,8],[4,7,5,2,1,2,4]]
Ordo 8	[[4,2,3,2,5,0,2,7],[3,0,7,2,0,1,2,2],[5,1,0,1,3,5,2,5],[1,8,7,2,3,4,2,4], [2,2,8,4,7,4,1,3],[1,7,2,6,3,1,4,2][1,4,7,5,2,1,2, 4], [10,2,0,2,6,10,4,10]]
Ordo 9	[[2,4,2,3,2,5,0,2,7],[5,3,0,7,2,0,1,2,2],[7,5,1,0,1,3,5,2,5],[1,8,7,8,2,3,4,2,4],[2,8,2,8,4,7,4,1,3],[1,0,7,2,6,3,1,4,2],[1,4,7,6,5,2,1,2,4],[1,4,2,7,9,6,3,2,4], [14,10,2,0,2,6,10,4,10]]
Ordo 10	[[2,2,4,2,3,2,5,0,2,7],[5,6,3,0,7,2,0,1,2,2],[7,5,1,9,0,1,3,5,2,5],[1,1,8,7,8,2,3,4,2,4],[2,8,2,8,4,4,7,4,1,3],[1,0,7,2,6,3,1,2,4,2],[2,2,16,14,16,4,6,8,4,8],[1,4,2,7,6,9,6,3,2,4],[4,8,5,6,4,2,7,4,2,8], [2,4,2,3,2,5,0,2,7,9]]

On the other hand, to compare program 1 and program2, samples matrix with size 5 to 10 containing row or column with most zero are used.

Table 3. Sample matrix used to compare the number iterations program1, program2 and program 3.

Ordo 5	[[4,3,0,0,0],[3,2,0,1,2],[0,6,4,10,0],[1,2,3,2,4],[0,4,7,4,1]]
Ordo 6	[[4,3,2,5,0,7],[3,2,0,0,0,0],[5,1,3,5,2,5],[1,2,3,4,2,4],[6,4,0,0,4,4],[1,6,3,1,4,2]]
Ordo 7	[[4,2,3,2,5,0,7],[0,7,0,0,1,2,0],[5,1,1,3,5,0,5],[2,7,2,8,4,2,9],[2,8,4,7,4,1,3],[2,14,4,6,8,4,8],[4,7,5,2,1,2,4]]
Ordo 8	[[4,2,3,2,5,0,2,7],[0,0,7,2,0,0,0,2],[7,1,0,1,9,5,2,9],[1,8,7,2,3,4,2,4],[2,2,8,4,7,4,1,3],[1,7,2,6,3,1,4,2],[1,4,7,5,2,1,2,4],[10,2,0,2,6,10,4,10]]
Ordo 9	[[2,4,2,3,2,5,0,2,7],[5,3,0,7,2,0,1,2,0],[7,5,11,0,1,3,5,2,5],[1,8,7,8,2,3,4,2,4],[2,8,2,8,4,7,4,1,3],[1,0,7,2,6,3,1,4,2],[1,4,7,6,5,2,1,2,4],[0,0,0,0,9,6,0,2,0],[14,10,2,0,2,6,10,4,10]]
Ordo 10	[[2,2,4,2,3,2,5,0,2,7],[5,6,3,0,7,2,0,1,2,2],[0,0,1,9,0,1,0,0,2,0],[1,1,8,7,7,7,3,7,2,4],[2,8,2,8,4,4,7,4,1,3],[1,0,7,2,6,3,1,2,4,2],[2,2,16,14,16,4,6,8,4,8],[1,4,2,7,6,9,6,3,2,4],[4,8,5,6,4,2,7,4,2,8], [2,4,2,3,2,5,0,2,7,9]]

RESULTS AND DISCUSSION

The represented the sample result of the *program1*.

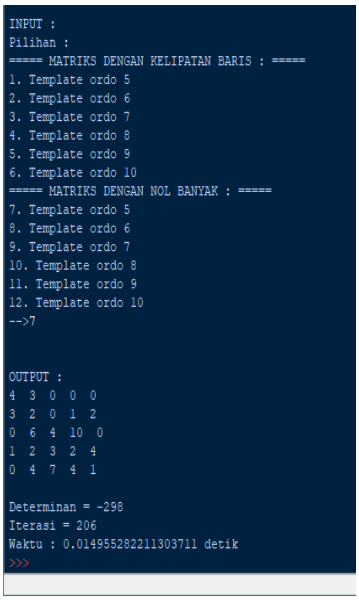

Figure 1. The result of the program1 using matrix ordo 5.

After running those three programs using given samples matrix, the results can be seen in Table 4 and Table 5.

It can be seen from the Table 4. that *program2* detect the characteristic of matrix, containing row multiple integer other row in n-matrix. Therefore program2 only did 1 iteration. The result would be different if the characteristic of matrix containing row or column multiple integer other row or column after some iterations in matrix with the size < n.

Table 4. The comprasion of the number of iterations program1 and program 2.

	Order 5		Order 6		Order 7		Order 8		Order 9		Order 10	
	Iterasi	Time	Iterasi	Time	Iterasi	Time	Iterasi	Time	Iterasi	Time	Iterasi	Time
Program 1	206	0.00446	1237	0.01015	8660	0.01895	69281	0.12955	623530	1.12113	6235301	11.53476
Program 2	1	0.00629	1	0.00908	1	0.00987	1	0.01024	1	000757	1	0.01331

Table 5. The comparison of the number of iterations program1, program2 and program 3.

	Order 5		Order 6		Order 7		Order 8		Order 9		Order 10	
	Iterasi	Time	Iterasi	Time	Iterasi	Time	Iterasi	Time	Iterasi	Time	Iterasi	Time
Program 1	206	0.01496	1237	0.01316	8660	0.02936	69281	0.12829	623530	1.22826	6235301	11.68181
Program 2	194	0.00421	1237	0.01854	8660	0.10956	62801	0.43502	578170	3.75425	5993381	41.68457
Program 3	45	0.011504	125	0.01396	1312	0.0331	8003	0.1282	74975	1.63498	749578	10.7766

The Table 5 illustrates the comparison among three program using given sample matrix in Table 3.

CONCLUSIONS

In conclusion, the data shows that the use of recursice algorithm to iteratively expand cofactor considering the row and column having highest number of zero, will reduce the number of iteration and computation.

From the Table 4. that *program2* detect the characteristic of matrix, containing row multiple integer other row in n-matrix. Therefore program2 only did 1 iteration. The result would be different if the characteristic of matrix containing row or column multiple integer other row or column after some iterations in matrix with the size < n. The Table 5 illustrates the comparison among three program using given sample matrix in Table 3.

REFERENCES

Anton, H., & Rorres, C. (2011). *Elementary Linear Algebra*.

Assaeed AM. 2007. Seed production and dispersal of *Rhazya stricta*. 50th annual symposium of the International Association for Vegetation Science, Swansea, UK, 23-27 July 2007.

Alikodra HS. 2000. Biodiversity for development of local autonomous government. In: Setyawan AD, Sutarno (eds) Toward Mount Lawu National Park; Proceeding of National Seminary and Workshop on Biodiversity Conservation to Protect and Save Germplasm in Java Island. Sebelas Maret University, Surakarta, 17-20 July 2000. [Indonesian]

Baclawski, K. (1982). Recursive Algorithms For Unitary And Symplectic Group Representations*, *3*(4), 592–605.

Balagadde FK, Song H, Ozaki J, Collins CH, Barnet M, Arnold FH, Quake SR, You L. 2008. A synthetic *Escherichia coli* predator-prey ecosystem. Mol Syst Biol 4: 187. www.molecularsystemsbiology.com

Cauchy, A. L. (1821). *Cauchy, Augustin Louis. Cours d'analyse de l'Ecole royale polytechnique; par m. Augustin-Louis Cauchy... 1. re partie. Analyse algébrique. de l'Imprimerie royale.*

Conte, E., De Maio, A., & Ricci, G. (2002). Recursive estimation of the covariance matrix of a compound-Gaussian process and its application to adaptive CFAR detection. *IEEE Transactions on Signal Processing*, *50*(8), 1908–1915. https://doi.org/10.1109/TSP.2002.800412

Debnath, L. (2013). A brief historical introduction to determinants with applications. *International Journal of Mathematical Education in Science and Technology*, *44*(3), 388–407. https://doi.org/10.1080/0020739X.2012.729682

Gauss, C. F. (1965). *Disquisitiones Arithmaticae*. Retrieved from http://www.elaprendiz.es/disquisitionesarithmeticae_Gauss.pdf

Graham, R. L., Knuth, D. E., Patashnik, O., & Liu, S. (1989). Concrete Mathematics: A Foundation for Computer Science. *Computers in Physics*, *3*(5), 106. https://doi.org/10.1063/1.4822863

Hou, H. S. (1987). A fast recursive algorithm for computing the discrete cosine transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *35*(10), 1455–1461. https://doi.org/10.1109/TASSP.1987.1165060

Kleene, C. (1981). Origins of Recursive Function Theory, *3*(1).

Muir, T. (1906). *The theory of determinants in the historical order of development* (Vol. 1). Macmillan and Company, limited.

Rai MK, Carpinella C. 2006. Naturally Occurring Bioactive Compounds. Elsevier, Amsterdam.

Saharjo BH, Nurhayati AD. 2006. Domination and composition structure change at hemic peat natural regeneration following burning; a case study in Pelalawan, Riau Province. Biodiversitas 7: 154-158.

Sugiyarto. 2004. Soil Macro-invertebrates Diversity and Inter-Cropping Plants Productivity in Agroforestry System based on Sengon. [Dissertation]. Brawijaya University, Malang. [Indonesian]

Webb CO, Cannon CH, Davies SJ. 2008. Ecological organization, biogeography, and the phylogenetic structure of rainforest tree communities. In: Carson W, Schnitzer S (eds) Tropical Forest Community Ecology. Wiley-Blackwell, New York.

Yan, W.-Y., B.Moore, J., & Helmke, U. (1994). Recursive algorithms for solving a class of nonlinear matrix equations with applications to certain sensitivity optimization problems*. *SIAM Journal on Control and Optimization*, *32*(6), 1559–1576.