Efficiency Comparison of Algorithms for Finding Determinants Via Permutations

Lugen M. Zake Sheet

Department of Mathematics, College of Basic Education, Al Mosul University

lujaenalsufar@uomosul.edu.iq

Received 01-12-2020, Accepted 13-01-2021, published 1-3-2021.

DOI: 10.52113/2/08.01.2021/56-65

Abstract: Determiners have been used extensively in a selection of applications throughout history. It also biased many areas of mathematics such as linear algebra. There are algorithms commonly used for computing a matrix determinant such as: Laplace expansion, LDU decomposition, Cofactor algorithm, and permutation algorithms. The determinants of a quadratic matrix can be found using a diversity of these methods, including the well-known methods of the Leibniz formula and the Laplace expansion and permutation algorithms that computes the determinant of any n×n matrix in O(n!).

In this paper, we first discuss three algorithms for finding determinants using permutations. Then we make out the algorithms in pseudo code and finally, we analyze the complexity and nature of the algorithms and compare them with each other. We present permutations algorithms and then analyze and compare them in terms of runtime, acceleration and competence, as the presented algorithms presented different results.

***Keywords*:** Determinants for Permutations, methods for Determinants, Generating Permutation.

1. Introduction

The There are several methods of finding the determinants of matrices, and among these methods (Cofactor method, LDU Decomposition, Laplace method), these methods have a different general equations between them.

The method of finding the determinant of matrices using permutations has one general equation which is O(n!) despite the multiplicity of algorithms for generating permutations. There are multiple studies for the purpose of informing and examining the efficiency of multiple methods for finding Determinants, including the one that he conducted [4] by comparing three methods in finding determinants of matrices. Also, a study was presented to make a comparison of parallel and sequential algorithms in finding the determinant of matrices by [9].

Here in this study I presented a study and comparison of a method for finding the determinant of the matrix through permutations and several algorithms presented to make a comparison to examine and test the higher efficiency of these methods with different algorithms for permutation, which gives us different ways to find the determinant of matrices.

In this study, the comparison algorithms for finding determinants of matrices by using various permutations methods were chosen only. This comparison was carried out with practical examples in the period of preparation and application of the research. We have found or discovered their competencies manually through practical examples. So Then I will only the steps of these algorithms are recorded. This is due to the difficulty in containing the one example in the research for each method for its length and cannot of absorption in the research papers.

2. Generating Algorithms Specified by Permutations

We will presentation of all three algorithms with detailed explanation of algorithms and algorithm diagrams. The algorithms for finding Determinants by permutations are based on the methods used to find permutations, and then we find the determinant. Since there are several methods of finding permutations, the ability and efficiency of finding the determinant depends on the efficiency of the two methods together. In this part, I will present the methods using its own program in order to make a comparison between them, including the following

2.1 Finding The Determinant by Using The Rrandom Method to Find Permutations

This method is one of the important methods in finding the determinant, and it has been used in many mathematical sciences in calculating the determinant, and the results of this method are characterized by good accuracy and speed. And this method was based on generating permutations using the usual method used, which is called the random method.

In these steps of the random method is to use the method of generating random permutations, then we use the method of finding the determinant by calling the equivalent matrices for each substitution and we find the product of multiplying their diameters and by adding all the diagonals we will find the determinant of the original matrix. The algorithm can be rendered in a graph spectrum.

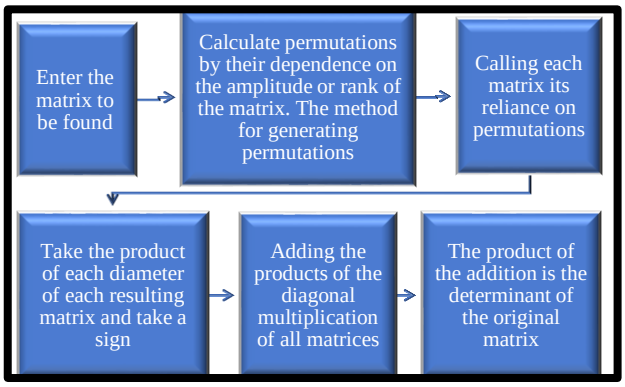


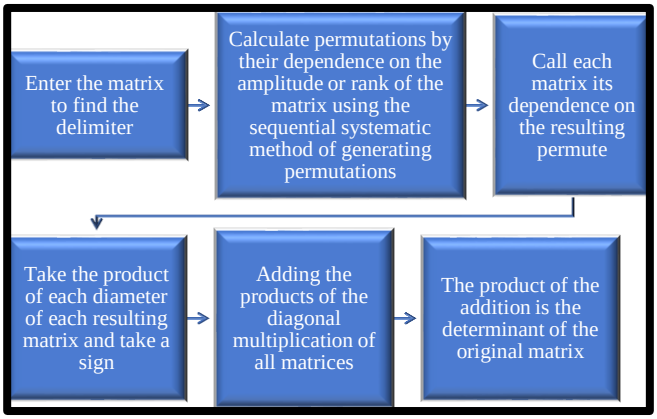
Fig. (1): Finding the determinant using random generation of Permutations.

Fig. (2): Finding the determinant by the installation of two permutations.

2.2 Finding The Determinant of Matrixes by Using Installing Two Objects to Find Permutations

This is the method that was suggested and published in my last research in 2020, and this method is characterized by ease and accuracy of results, and the method relied on generating permutations in the systematic, sequential way after installing two elements. This method of generating permutations reduces the time spent producing all permutations, which helps the method to find the determinant by reducing the time in finding the determinant of the matrix.

The method is summarized by finding the determinant of the required matrix that it is done using the generation of permutations by relying on the systematic generation algorithm chained by fixing two elements, and this algorithm has proven high efficiency in generation. The efficiency of finding the determinant and in the following scheme of this method.



2.3 Finding The Determinant of Matrixes by Using Fixing One Element Only to Find Permutations

This method was introduced in 2020 and it was a very good way to find the determinant using permutations compared to other previous methods after performing comparative tests with some methods, for example, Laplace method, LDU Decomposing method, and Cofactor method for determinants. This method relied on inferred permutations by fixing only one element, which was called the primitive groups method.

This method creates a determinant for high capacity square arrays and saves a lot of time and effort to extract the final output of the determinant. The method depends on the permutations resulting by using the fixation of only one element on a regular basis in finding the rest of the group of permutations. This method takes a different pattern than the previous one, as it depends on finding the determinant of the matrix using the inverse of the diameter in finding the inverse of the permutations to extract all the permutations and extracting their determinants as shown in the following diagram:-

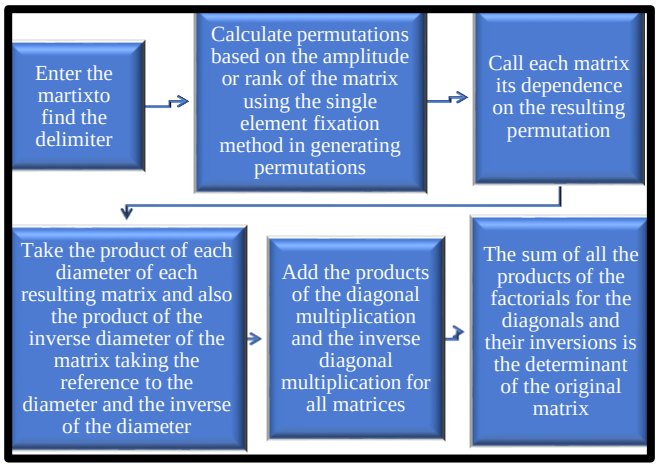


Fig. (3): Finding the determinant using regular generation of permutations by fixing only one element.

3. Comparison of The Presented Algorithms

We talk about all the comparisons and their results, which are shown in a diagram. We mentioned in the first part that finding the determinant of matrices uses several methods, and each method has a certain efficiency, and one of these methods was to find the determinant of matrices using generating permutations, and since generating permutations has several algorithms, the efficiency of finding the determinant depends on the efficiency of the permutation algorithm and we mentioned in the previous chapter these methods for finding the determinant With different algorithms to generate the determinant and permutations and by writing these programs for the three methods mentioned in the previous part in Matlab language and applying them to the same matrix on the computer and making the comparison for the three methods using square matrices with different capacities for all the tests and we started the test with a 4×4 matrix and thus we graduated the capacitance to 9×9 of the matrices used so that The matrix was the same for all the methods used with the same capacity in order to calculate the efficiency in terms of speed, and for all three methods in finding the determinant of the three recessed methods, the results were as follows:-

Table 1. The results of the comparison

Size	Determined using the Random permutation method	Determined using the regular method of by fixing one element	Determined using the concatenated regular method by fixing two elements
3	0.58444	0.01222	0.004876
4	12.04401	0.02875	0.010365
5	16.98645	0.06561	0.026442
6	25.77767	0.48960	0.256883
7	38.40388	1.32065	0.985106
8	92.91468	38.01198	5.426988
9	449.15959	212.31651	142.694230

The following is a graph of the results that appeared during the test as follows-:

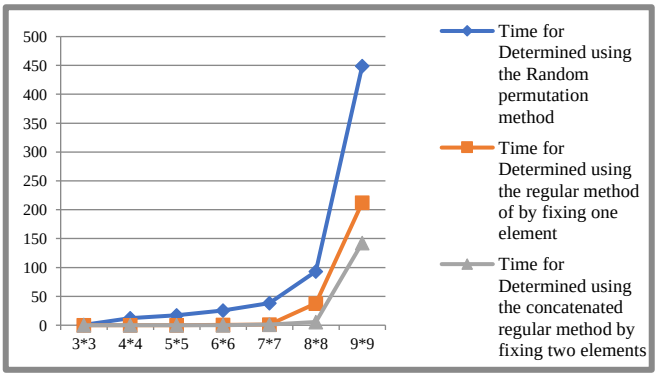


Fig. (4): A graph of the time consumed for the three methods

4. Comparison Results

Through the test, we notice that the method for finding the determinant of matrices using

generating permutations by fixing two elements is the least time consumed in finding the determinant. This indicates that this method is the most efficient with respect to the other methods used in this test.

Also, we notice through the results that the method that uses generating permutations by installing an element also has good efficiency in terms of time consumed and it comes second in terms of time consumption. Thus, multiple tests of different capacities of arrays have proven that the method that uses generating random permutations, which uses our habit in all fields, is the least efficient, as shown in the following diagram:-

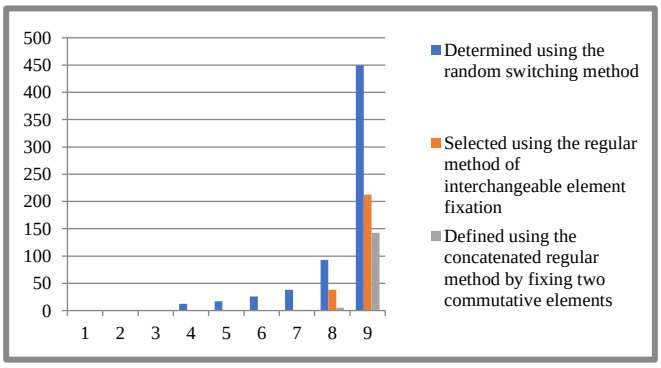


Fig. (5): Time spent methods of comparison.

5. Another Comparison To Evaluate

There are a lot of additional criteria I encountered in evaluating the roads when compared. Apart from the time spent in finding the determinant value of the proposed matrices for comparison there was the effort expended by The researcher. Where there was a difference in the effort used to find the determinant when using manual means in all ways. The effort to find the determinant with the algorithm that uses the fixation of two elements was equal to half the effort expended to find the determinant of the algorithm that uses the fixation of only one element, which in turn was outweighed by the effort expended by the algorithm that used alternating randomness.

This additional criterion was used when using the amplitude different matrices for all algorithms in the manual method using the paper.

The steps used in building this work were illustrated by the following points:-

1- In this work we have provided a summary with detailed diagrams of three methods, using permutations to find the determinant of the matrices (choosing methods).

2 - And then we wrote the algorithms for each method using the program in Matlab language (writing programs)

3- We conducted the test under the same conditions (the same matrices) for all methods of conducting experiments in order to compare the algorithms

4- Record the results

5- Comparison with experimental results

There is an illustrative and detailed plan used for these steps, on which the detailed construction planning work is shown as follows:-

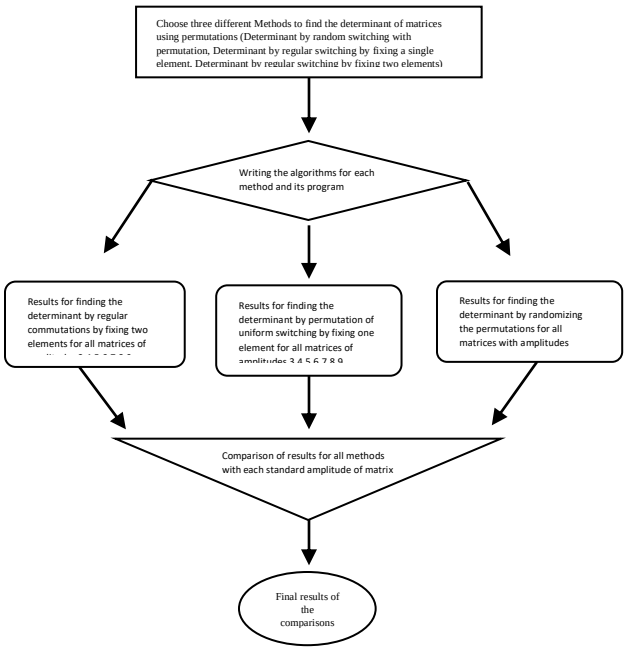


Fig.(6): Flowchart the build of work

6. Conclusion and Recommendation

This study is considered a complementary study to previous studies on research and finding new methods that are considered more efficient than the previous ones. As shown by the presented results of the presented algorithms, the proposed new algorithm which is regular by fixing two element time-advanced elements in all small arrays of size 3, whose result was (0.004876) compared to the results of other algorithms whose product; The regular method by fixing one element is (0.01222) and the Random permutation method (0.58444), and so on with the matrix size 4 also the results were gradually in favor of the regular method by fixing two elements (0.010365) While the other algorithms are (0.02875) for algorithm by regular by fixing one element, and (12.04401) for algorithm Random Permutation. In terms of high amplitudes, which are more than size 7, the results were also as follows (0.985106), (1.32065), (38.40388) which prove that the proposed algorithm which is regular by fixing two elements reads fewer results than other methods as regular by fixing one element and Random Permutation Method, as well as with the matrix size 9, the resulting algorithm regular by fixing two elements is the one that reads the lowest results such as (142.694230), (212.31651), (449.15959). From the rest of the previously proposed methods for using permutations in finding determinants.

Therefore, this study is a development of the previous method that I presented in previous years on fixing the substitution element to create a specific matrix. In fact, this study was a successful study because its results provided stronger than the previous study.

Through this work, I advise workers in this field to take the new approach in finding determinants of matrices when conducting their comparative experiments to test the efficiency of methods to find determinants of matrices.

References

[1] A. Salihu, “New Method to Calculate Determinants of n×n (n≥ 3) Matrix” by Reducing Determinants to 2nd Order,” *International Journal of Algebra*. Vol 6, no 19, 2012, pp 913- 917.

[2] E. Torrence, E. Rice, " Lewis Caroll's Condensation method for evaluating determinants*" Journal Math Horizons, 2006* , 12-15.

[3] F. Ahmad, and H. Khan, "An efficient and simple Algorithm for matrix inversion" *International Journal of Technology Diffusion* , 2010, pp 20-27.

[4] F. Limanta, " Efficiently Calculating the Determinant of a M atrix" *Conference in Indonesia Institut Teknologi*, Makalah 2017, IF2120 Matematika Diskrit.

[5] G. Rote, " Division-Free Algorithms for the Determinant and the Pfaffian" *Algebraic and Combinatorial Approaches. In Computational Discrete Mathematics*, Berlin, Germany: Springer Berlin Heidelberg, 2001, pp. 119-135.

[6] H. Khan, I. Shah, and F. Ahmad, "An efficient and generic algorithm for matrix inversion" *Internantional Jounal of Technology diffusion, 2010* , 36-41.

[7] J. Jones, " The Determinant of a Square Matrix" [Online]. Available: https://people.richland.edu/james/lecture/m116/matrices/determinant.html.

[8] L. M.Vavra, "Cramer's Rule" *M.S. dissertation*, Dept. Math., Univ. Warwick, Warwick, England 2010.

[9] S. Almalki, "New parallel algorithms for finding determinants of N×N matrices" *Conference Paper, June 2013 IEEE* 978-1-4799-0462-4/13.

Appendixes

1. The method for finding determinants of matrixes using random generation of permutations

```
function det_per =get det_per (x);
% permutation algorithm for determinant by lexicographic
% this script algorithm for determinant by lexicographic algorithm
% NFAC = number factor
% sine = output lexicographic permutations
% result = output for determinant
tic;       % (calculate the execution time for running)
n=input ('Enter n :');
M=input ('Enter array :');
NFAC=1.0;
mul =1;
count =0;
result =0;
%%%% (find factorial n) %%%%
        for i=1:n
          NFAC=NFAC*i;       % (build the factorial permutation)
        end
NFAC
%%%% (generate permutation by lexicographic) %%%%
p = perms (1:n)
  for i=1:NFAC
    r= M (:,  p(i,:) );
   arrays =r
  sine =p(i,:)
%% (generate matrices based on lexicographic permutations) %%%%
    for rows=1:n
      for col=1:n
      if rows==col          % (get the diagonal matrix)
 mul=mul*arrays(rows,col);    % (multiple elements diagonal matrix)
      end
```

```
    end
end
%%%% (get sign diagonal by inversion) %%%%
  for si1=1:n
      for si=si1:n
         if sine(1,si1)> sine(1,si)          % (count the version pair)
count=count+1;                          % (sum version pair)
        end
    end
end
%%%% (find the type sign for multiple the elements diagonal) %%%%
sum=mul;
    if mod(count,2) ~= 0                              % (if the count odd
except to divide by 2)
    sum=sum*-1;                                       % (if the count even
multiple with -1)
end
%%%% (find summation all multiple diagonal matrices) %%%%
sum
count =0;
  mul =1;
result=result+sum     % (getting determinant for original matrix)
  sum=0;
 end
result
toc
```

2. The method for finding determinants of matrixes using generation by installing two permutations

```
function Der_div_init = get Der_div_init (n));
%% program to make permutations and Determinant quickly with less
memory usage
%% Step2%%%%
     generate1.m
  clear
clc
% result = output for determinant
  tic;
% (calculate the execution time for running)
n=input ('Enter n :');
M=input ('Enter array :');
   mul =1;
    count =0;
     result =0;
       n=5;
      d=basicsellect(1:n,[1 n]);
        a=[1*ones(size(d,1),1) d n*ones(size(d,1),1)];
%% Step3%%%%%
    if size(a,1)==1
      d=basicsellect(a(1,:),[1 i]);
         b=[[1*ones(size(d,1),1) d i*ones(size(d,1),1)]];
      else
    b=[];
       for i=n-1:-1:2
         d=basicsellect(a(n-i,:),[1 i]);
           b=[b;[1*ones(size(d,1),1) d i*ones(size(d,1),1)]];
       end
  end
%% Step4 %%
   makecycle.m
       c=makecycle([a;b]);
          d=[c;fliplr(c)];
c
%% Step5 (generate matrices based on permutations fixing two ) %%%
       for i=1:c
            r = M (:,  c(i,:) );
              arrays =r
      sine =c(i,:)
%%%% Step6 (get diameter matrices based on permutations) %%%%
   for rows=1:n
        for col=1:n
     if rows==col         % (get the diagonal matrix)
  mul=mul*arrays(rows,col);      % (multiple elements diagonal matrix)
      end
   end
end
%%%% Step7 (get sign diagonal by inversion) %%%%
  for si1=1:n
      for si=si1:n
         if sine(1,si1)> sine(1,si)          % (count the version pair)
count=count+1;                % (sum version pair)
```

```
      end
   end
end
%%%% (find the type sign for multiple the elements diagonal) %%%%
sum=mul;
   if mod(count,2) ~= 0      % (if the count odd except to divide by 2)
   sum=sum*-1;          % (if the count even multiple with -1)
end
%%%% Step8  (find summation all multiple diagonal matrices) %%%%
sum
count =0;
 mul =1;
    result=result+sum        % (getting determinant for original matrix)
 sum=0;
 end
result
toc
```

3. The method for finding determinant of matrixes by using regular generation by installing one element only permutations

```
function det_temp =get_det_2(x);
% new algorithm for determinant by starter sets
% this script algorithm for determinant by starter sets algorithm
% init_mat= output starter sets permutations
% init_mat_b= output equivalent starter sets permutations
% det_temp= output for determinant
 tic;              % (calculate the execution time for running program)
x=input ('input x =   ')
n=size(x, 1);
det_temp=0;
init_mat_rows=1; % (variable (init_mat) to define the number of rows of the   permutation)
          for i=2:n-1
       init_mat_rows=init_mat_rows*i;  (build the factor permutation)
          end
init_mat_cols=n;                    % (number the columns equal n)
init_mat (1:init_mat_rows,1)=1;      % (first column from the rows =1)
r=init_mat_rows;
u=1;
          for j=2: n
              i=1;
                  r=r/ (n+1-j);
t=1;
                  u=init_mat_rows/r;
for v=1: u
                   flag=1;      % (control flag, control returns to the 1)
while t > n ||flag==1
                  if t >= n
t=1;
       end
     t=t+1;
        flag=0;          % (control flag, control returns to the 1)
         for p=1:size(init_mat,2);
         if init_mat(i,p)==t
        flag=1;
       end
   end
end
     for k=1:r
      init_mat (i, j) =t;
      i=i+1;
    end
  end
end
init_mat
%%%% (create equivalent starter sets permutations) %%%%
init_mat_b=ones (init_mat_rows, n);  %(create rows equivalent
permutation)
%%%% (create equivalent permutations) %%%%
     for i=1:init_mat_rows
       for j=2:n
init_mat_b (i, j) =init_mat (i, n+2-j);  (inverse permutation after fix 1
element)
    end
end
init_mat_b
%%%% (delete equivalent starter sets permutations) %%%%
i=1;
   while i<=init_mat_rows
j=i;
   while j<=init_mat_rows
```

```
    if init_mat (i,:)==init_mat_b (j,:)
 init_mat (j, :) = [] ;
  init_mat_b (j, :) = [] ;
 init_mat_rows=init_mat_rows-1;     % (delete one from equivalent permutation)
   end;
  j=j+1;
 end
 i=i+1;
end
init_mat_size=size (init_mat)
%% (create matrices based on starter sets permutation (init_mat)) %%
    for i=1:init_mat_size
     y=init_mat (i, :);
    for j=1: n
    generated_mat (:, j)=x(:,y(j));  % (generate matrices based on starter sets permutations)
 end
%% (generate all sub matrices by using cycle in columns) %%
   for j=1:n
       y_space (n) =y (1);
       x_space (:, n) =generated_mat (:,1);
    for k=1:n-1
       y_space (:, k) =y (:,k+1);  % (one space in the permutation)
       x_space (:, k) =generated_mat (:, k+1);  (one space in the colum )
  end
 y=y_space
generated_mat=x_space
d=y;
% (finds the sign permutations and sign inverse permutations) %%
d_size=length (d);
sign_val_1=1;                % (sign permutations)
sign_val_2=1;                 % (sign inverse permutations)
   for i=1:d_size
      for j=i+1:d_size
       sign_val_1=sign_val_1*(d (1, j)-d (1, i));   % (find sign permutations by inversion)
  end;
 end
    for i=1:d_size
       d_2 (1, i) =d (1, d_size+1-i);     % (finds inverse permutation)
   end
    for i=1:d_size
      for j=i+1:d_size
     sign_val_2=sign_val_2*(d_2 (1,j)-d_2(1,i));  % (sign inverse permutations by inversion)
      end
  end
    if  sign_val_1 < 0
     sign_val_1= -1;
     else sign_val_1=1;
  end
    if  sign_val_2 < 0
    sign_val_2= -1;
   else sign_val_2=1;
end
% (finds sub determinants by multiple elements diagonal and inverse) %
det_1=prod (diag (generated_mat));       % (multiple the elements diagonal matrix)
      for i=1:n
x_2 (i) =generated_mat (i,n+1-i);  end;  (create the inverse diagonal matrix)
 det_2=prod (x_2);     % (multiple the elements inverse diagonal matrix)
 sub_det=sign_val_1*det_1+sign_val_2*det_2     % (get the sub determinants of matrices)
    det_temp=det_temp+sub_det;
  end
end
det_temp
toc
```