Research Inventy: International Journal Of Engineering And Science
Issn: 2278-4721, Vol. 2, Issue 3 (February 2013), Pp 49-54
Www.Researchinventy.Com

Nondeterministic Procedure of Solving Simultaneous Equations

N. Shobha Rani
Research scholar, Maharaja Research Foundation, MIT, Mysore.

Abstract:*This paper revises the study of computational complexity of solving the simultaneous equations. An equation is an expression of the form*

$$W_1.W_2.....W_k=id$$

where each W_i is a variable with operator and id is the identity element. A solution to such an equation is the assignment of variables which realizes the equality. A system of equations is a collection of such equations; a solution is the assignment of values to unknown variables which simultaneously realizes each equation. We show that the problem of determining the solution is NP. The system of simultaneous equations is probably one of the most important topics in modern engineering computations. The paper examines the Methods of solving the Simultaneous Equations. Section 1 presents introduction. Section 2 discusses the Simultaneous Equations solution as NP. Section 3 describes Method of solving Simultaneous equations in detail. Section 4 Concludes.

Keywords: *Simultaneous Equations, NP, Gaussian elimination*

I Introduction:

Many natural computational problems can be viewed as problems about the solvability of certain equations. When asked to characterize computation and to describe all of the problems that are solved through computation, a computer scientist will usually begin by describing a programming language in which to do the computation. Most of us would present an imperative language such as C++, but some might mention functional languages or logic programming languages. All however, would probably then discuss computation in terms of the resources required to perform the required task. There is another way, namely mathematical programming. This is where a problem is defined as the solution to a system of simultaneous equations. In fact, every one of the problems which can be computed using programs written in any favourite languages can also be described in terms of mathematical programming. The term Simultaneous equations or system of equations refer to conditions where two or more unknown variables are related to each other through an equal number of equations. A pair of equations with more than one value to be found is called simultaneous equations**.** The two equations cannot be solved on their own. Each one by itself has infinitely many solutions. For example x + y = 10 has infinitely many values for x and y for which this is true, like x=1, y=9 or x=10, y=0 or x=100, y=-90, etc. But two equations can be solved together to make one equation that has only one solution. The x and y values of this solution will solve both of the original equations at the same time. This is why they are called simultaneous equations, because of solving them both with the same values for x and y.Systems of simultaneous equations can be found in many engineering applications and problems. Systems that consist of small number of equations can be solved analytically using standard methods from algebra. Systems of large number of equations require the use of numerical methods and computers. The system of simultaneous equations is probably one of the most important topics in modern engineering computations. This is not an exaggeration if one considers that recent technological advances were made possible by the ability of solving larger and larger systems of equations.

Few examples of Simultaneous equations are as below:

example1

$X_1 + 3X_2 + 2X_3 = 15$
$2X_1 + 4X_2 + 3X_3 = 22$
$3X_1 + 4X_2 + 7X_3 = 39$

example2

$X_1 + 2X_2 - 4X_3 + \ldots\ldots\ldots = 20.1$
$-2X_1 + X_2 - 3X_3 + \ldots\ldots\ldots = 2$
$2X_1 - 7X_2 - 4X_3 + \ldots\ldots\ldots = -6.5$
.....
$8X_1 - 2X_2 + 8X_3 + \ldots\ldots\ldots = -11$

II Simultaneous Equation Solution As NP

Algorithm:

1. Read the Number of Unknown
2. Get the co-efficient Values
3. Determine the Identity
4. Select Randomly the values for the unknown
5. Substitute these values in Eq's check the result with the corresponding Identity, if not equal goto step 4 again
6. Solution found, display the values of Unknown
7. Soultion not found, display no solution

Implementation:

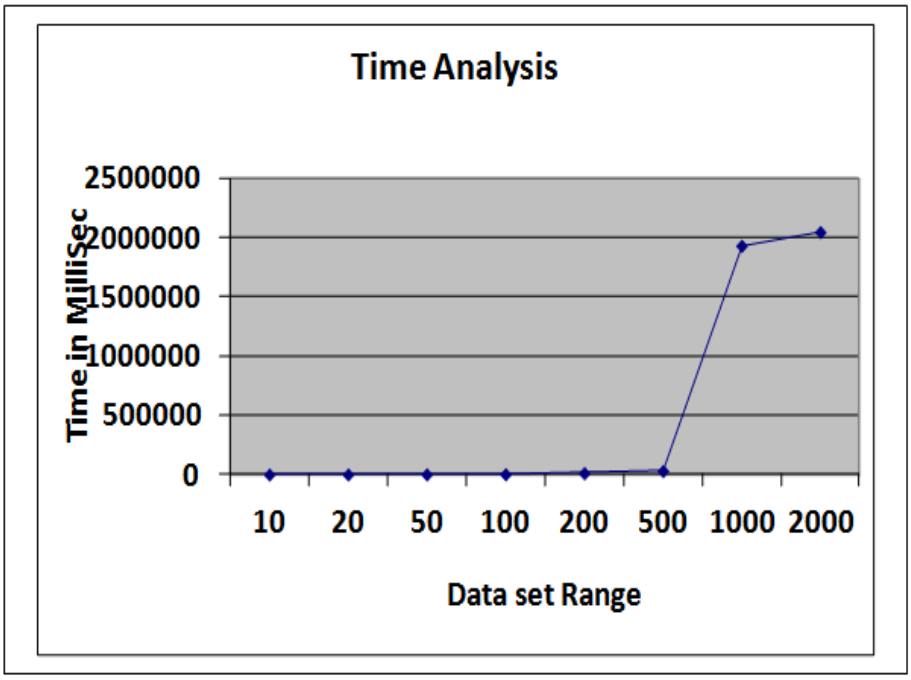
```
/*
* Program to find solution for Simulataneous *  Eq using exhaustive search method which
*   is a NP
*
*/
import java.io.*;
import java.util.Random;

class simeq
{
 public static void main(String args[])throws java.io.IOException
{
  int n;
  BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
  String str;
  System.out.println("Enter the No of Equ's:");
  str = br.readLine();
  n = Integer.parseInt(str);
  int a[][]= new int[n][n];
  int b[] = new int[1*n];
  int x[] = new int [1*n];
// read a[]
System.out.println("Enter the coefficients:");
for(int i=0;i<n;i++){
     for(int j=0;j<n;j++){
           str = br.readLine();
           a[i][j] = Integer.parseInt(str);
     }
}

// read b[]
System.out.println("Enter the Identity:");
for(int i=0;i<n;i++){
      str = br.readLine();
      b[i] = Integer.parseInt(str);
}

 // Execution time
long start, end;
start = System.currentTimeMillis();
// get starting time
l1: do{
// Randomly assign values for variables
      Random rand = new Random();
      for(int i=0;i<n;i++){
          x[i] = (int) (5000 * rand.nextDouble());
```

```
    }

    for(int i=0;i<n;i++)
    {
        int res=0;
        for(int j=0;j<n;j++)
        {
            res+=(a[i][j] * x[j]);
        }
        if(res!=b[i])
          continue l1;
    }
    break;
}while(true);
end = System.currentTimeMillis();
// get ending time
System.out.println("Elapsed time: " + (end-start)+" MilliSeconds");
for(int i=0;i<n;i++)
      System.out.println("x"+i+"="+x[i]);

 } // end of main
} // end of class
```

Analysis:

Consider the Eq's of 3 unknown

$X_1+3X_2+2X_3 = 15$



$2X_1+4X_2+3X_3 = 22$

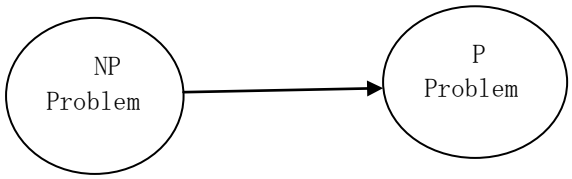
$3X_1+4X_2+7X_3 = 39$

Solving the equation using exhaustive search method, $X_1=1$ $X_2=2$ $X_3 = 4$

The exhaustive search method takes large time duration as the Data set range increases (Number to be chosen from the range) ex:- 10 Data set Range indicates that the randomly number to be chosen is from 0 to 10, which takes approximately 16 MilliSeconds as the dataset range increases the execution time increases for 2000 the time is 2047578 MilliSeconds. In reality the Data Set Range happens to be infinity hence the time taken is very very high, hence it is an NP.

III Solving Simultaneous Equations Deterministically:

Problem Transformation:



The System of Simultaneous equations can be solved by using any of the deterministic procedures like Substitution method, matrix method, graphical method or the Gauss elimination procedure which is feasible to deal with the System of equations of n unknowns.

number of unknowns.

Substitution Method : Consider a set of linear equations also known as a linear system of equations
$2x + y = 8$
$x + y = 6$
Solving this involves subtracting $x + y = 6$ from $2x + y = 8$ (using the elimination method) to remove the y-variable, then simplifying the resulting equation to find the value of x, then substituting the x-value into either equation to find y.
The solution of this system is: $x= 2$ and $y=4$, which can also be written as an ordered pair (2, 4), representing on a graph the coordinates of the point of intersection of the two lines represented by the equations.

Algorithm overview
The process of Gaussian elimination has two parts. The first part (Forward Elimination) reduces a given system to either triangular or echelon form, or results in a degenerate equation, indicating the system has no unique solution but may have multiple solutions (rank<order). This is accomplished through the use of elementary row operations. The second step uses back substitution to find the solution of the system above. Stated equivalently for matrices, the first part reduces a matrix to row echelon form using elementary row operations while the second reduces it to reduced row echelon form, or row canonical form.

ALGORITHM :
1. Locate the diagonal element in the pivot column. This element is called the **pivot**. The row containing the pivot is called the **pivot row**. Divide every element in the pivot row by the pivot (ie. use E.R.O. #1) to get a new pivot row with a 1 in the pivot position.
2. Get a 0 in each position below the pivot position by subtracting a suitable multiple of the pivot row from each of the rows below it (ie. by using E.R.O. #2).
3. Apply Back substitution method to obtain the values of unknowns.
4. Verify the Solution set.

Program :

```
/*Gaussian Elimination.C*/
#include <stdio.h>
#include <conio.h>
#define N 100
float coeff[N][N+1];
/* Co-efficient inputing variables */
void gauss(int n);   /* deterministically solves the equations */
void verify(int x[]);  /*Verifies whether the solution set is correct or not */
  void main()
{
 int i,j,k;    /*    Loop variables */
   int n;  /* Number of equations */
   float pivot;    /* pivoting variables */
   float sum;    /* Back substitution sum storing variable */
   float x[N];  /* values of the variables i.e. x's */
   char ch;        /* choice inputing variable */
   do
```

```
    {
       printf("\nEnter the number of variables\n");
       scanf("%d",&n);
       for(i=0;i<n;i++)
       {
          for(j=0;j<n+1;j++)
          {
             printf("\nEnter %d row %d col element\n",i+1,j+1);
             scanf("%f",&coeff[i][j]);
          }
       }
 gauss(n);
verify(x);
       printf("\nDo you wish to continue[y/n]\n");
       fflush(stdin);
       scanf("%c",&ch);
    }while(ch=='y' || ch=='Y');
    getch();
}
  void Gauss(int n )
{
     for(k=0;k<n;k++)
       {
          for(i=k+1;i<n;i++)
          {
             pivot=coeff[i][k]/coeff[k][k];
             for(j=k;j<n+1;j++)
                coeff[i][j]=coeff[i][j]-pivot*coeff[k][j];
          }
       }
/* Back Substitution */
       x[n-1]=coeff[n-1][n]/coeff[n-1][n-1];
       for(i=n-2;i>=0;i--)
       {
          sum=0;
          for(j=i;j<n+1;j++)
             sum=sum+coeff[i][j]*x[j];
          x[i]=(coeff[i][n]-sum)/coeff[i][i];
       }  }
 verify( int  x[])
{  int  i ,j;  /*verifying equations */
for(i=0;i<n;i++)
{ res=0;
for(j=0;j<n;j++)
{       res=res+coeff[i][j]*x[j]; }
if(res==x[i])
{       for(i=0;i<n;i++)
          printf("\n x[%d] = %g\n",i+1,x[i]);
  }
else {
 printf("The given equations are inconsistent and have no solution");
break;
}   }
return;
}
```

Program Explanation:

In our Program., the coefficients are read in a matrix and no. of the variables are read dynamically .the program is computing the solution in two steps forward elimination and back substitution which is incorporated in the method Gauss and the verification of the solution is done in the second method of the program i.e. , verify method.

Time Complexity:"Gaussian" elimination runs in O(n ^{3}) time. While it is obvious that the algorithm uses O(n^3) arithmetic operations. Gaussian elimination to solve a system of n equations for n unknowns requires $n(n+1) / 2$ divisions, $(2n^3 + 3n^2 − 5n)/6$ multiplications, and $(2n^3 + 3n^2 − 5n)/6$ subtractions,[4] for a total of approximately $2n^3 / 3$ operations. Thus it has arithmetic complexity of O(n^3). However, the intermediate entries can grow exponentially large, so it has exponential bit complexity

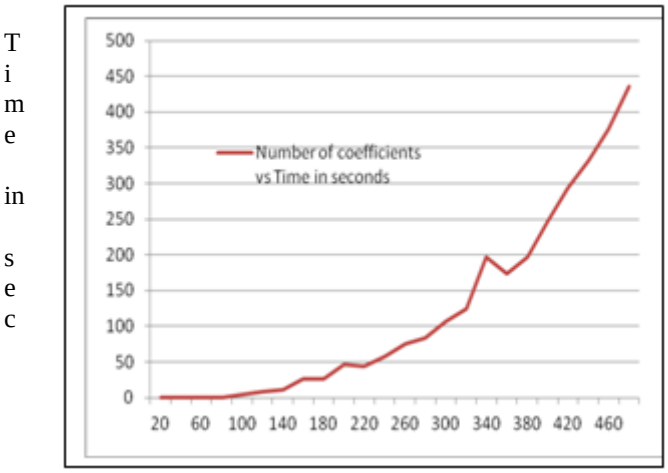


Figure-2: No of Coefficients (Unknown)

IV Conclusion.

The Gaussian elimination can be performed over any field. Gaussian elimination is numerically stable for diagonally dominant or positive-definite matrices. For general matrices, Gaussian elimination is usually considered to be stable.

References:

[1] Ron Larson, *"Elementary Linear Algebra"*, 5th Edition, The Pennsylvania State University, The Behrend College
[2] Mikael Goldmann, Alexander Russeli, *"The complexity of solving Equations over finite group"*, 2003
[3] Gary D. Knott, *"Gaussian Elimination and LU-Decomposition"*, www.civilized.com, Feb 28, 2012
[4] Joseph F. Grcar, *"Mathematicians of Gaussian Elimination"*, Notices of the AMS, June 2011