# Gauss Jordan Elimination Method

**Pseudo-Code:**

```
1. Start
2. Read the order of the matrix 'n' and read the coefficients of the linear equations.
3. Do for i=0 to n-1
        Do for j=0 to n-1
            If (i equal to j) then,
                Set pivot = a[i][i]
                Do for k=0 to n-1
                    a[j][k] = a[i][k]/pivot;
                End for k
            Else
                Set pivot = a[j][i]/a[i][i]
                Do for k=0 to n-1
                    a[j][k] = a[j][k] – pivot*a[i][k];
                End for k
            Endif
        End for j
   End for i
4. Display Solution:
   Do for i=0 to n-1
        x[i] = a[i][n]
        Display x[i]
   End for i
5. Stop
```

**Basic Equations:**

$3x + 2y + 1z = 10$
$2x + 3y + 2z = 14$
$1x + 2y + 3z = 14$

*x= 1, y=2, z=3*

$4x + 2y + 3z = 4$
$2x + 2y + z = 6$
$x + y + z = 0$

*x= 6, y=1, z=-6*

$3x + 2y - 4z + 3u = 2$
$2x + 3y - 3z - u = 1$
$x + 2y + 3z - u = 10$
$2x - y + 2z + 3u = 7$

*x= 1, y=2, z=2, u=1*

## Program Code in C

```
/******************************************************************************
Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
Code, Compile, Run and Debug online from anywhere in world.
*******************************************************************************/
#include<stdio.h>
int main(){
   int i, j, k, n=3; float c, pivot_el, x[3], a[3][4] ={{3,2,1,10},{2,3,2,14},{1,2,3,14}};

   for(i=0; i<n; i++){
      for(j=0; j<n; j++){
         if(i==j){
            pivot_el = a[i][i];
            for(k=0; k<n+1; k++){
               printf("Before: a[%d][%d] = %f and pivot_el = %f",i,k,a[i][k],pivot_el);
               a[j][k]=a[i][k]/pivot_el;
               printf("\tAfter: a[%d][%d] = %f\n",i,k,a[i][k]);
            }
            printf("\n");
         }else{
            pivot_el=a[j][i]/a[i][i];
            for(k=0; k<n+1; k++){
               a[j][k]=a[j][k]-pivot_el*a[i][k];
            }
         }
      }
   }

   // Final Matrix
   printf("The Final Matrix is:\n");
   for(i=0; i<n; i++){
      for(j=0; j<n+1; j++){
         printf("%f\t",a[i][j]);
      }
      printf("\n");
   }

   printf("\n\nThe solution is:\n");
   for(i=0; i<n; i++){
      x[i]=a[i][n];
      printf("x%d=%f\n",i,x[i]);
   }
   return(0);
}
```

**Output**

```
Before: a[0][0] = 3.000000 and pivot_el = 3.000000 After: a[0][0] = 1.000000
Before: a[0][1] = 2.000000 and pivot_el = 3.000000 After: a[0][1] = 0.666667
Before: a[0][2] = 1.000000 and pivot_el = 3.000000 After: a[0][2] = 0.333333
Before: a[0][3] = 10.000000 and pivot_el = 3.000000 After: a[0][3] = 3.333333

Before: a[1][0] = 0.000000 and pivot_el = 1.666667 After: a[1][0] = 0.000000
Before: a[1][1] = 1.666667 and pivot_el = 1.666667 After: a[1][1] = 1.000000
Before: a[1][2] = 1.333333 and pivot_el = 1.666667 After: a[1][2] = 0.800000
Before: a[1][3] = 7.333333 and pivot_el = 1.666667 After: a[1][3] = 4.400000

Before: a[2][0] = 0.000000 and pivot_el = 1.600000 After: a[2][0] = 0.000000
Before: a[2][1] = 0.000000 and pivot_el = 1.600000 After: a[2][1] = 0.000000
Before: a[2][2] = 1.600000 and pivot_el = 1.600000 After: a[2][2] = 1.000000
Before: a[2][3] = 4.800001 and pivot_el = 1.600000 After: a[2][3] = 3.000000

The Final Matrix is:
1.000000	0.000000	0.000000	1.000000
0.000000	1.000000	0.000000	2.000000
0.000000	0.000000	1.000000	3.000000

The solution is:
x0=1.000000
x1=2.000000
x2=3.000000
```