Algorithm Applied-

- The program divides the rows of the equation matrix equally among the processes. Each process gets approximately n/p rows to process.
- On every iteration of the main loop, the current row is broadcasted to all the processes by the process that has been assigned that row according to the "map" variable. Each process then subtracts this row from its own assigned rows so that the foremost element becomes zero. It also subtracts the RHS elements according to the ratio.
- When the above iteration ends, the root process then applies back substitution to calculate the actual value of all the variables in the given input matrix and displays it.

Theoretical Speedup Possible-
Time complexity of Serial execution = n^2(For input) + $n*(n+1)*(2*n +1)/6$ (For execution of Gaussian algorithm) + $n*(n+1)/2$ (For back substitution)

Time complexity of Parallel execution = $n^2 + n*(n+1)*(2*n +1)/(6*p) + n*(n+1)/2$

Here n refers to the number of equations in the input and p refers to the number of processes

Theoretical Speedup = $(2*n^3 + 12*n^2 + 4*n)/(9*n^2 + 3*n + (n*(n+1)*(2*n + 1))/p)$

Actual Speedup for input of 100 equations-

Time taken for sequential execution = 0.019971

Time taken for p(=2) = 0.041152, Speedup = 0.4853

Time taken for p(=5) = 0.116553, Speedup = 0.1713

Time taken for p(=7) = 0.193809, Speedup = 0.1030

Time taken for p(=10) = 32.29180, Speedup = 0.000618

Mismatch in experimental and actual speedup may be because of increasing communication costs while increasing the number of processors. (Since, we have ignored the communication costs in analysis)