Abstract

The objective of this research is to design the parallel algorithm of some numerical problems and also evaluate their performances on multiprocessor systems. Numerical problems arise in almost every branch of science. These problems are finding the solution of system of linear equations, numerical integration, and polynomial interpolation and their applications in method of moment model of Maxwell’s equations, fusion energy model, image processing, weather forecasting, and neural network. These problems are generally large in size and take too much time in sequential computation. Also there is demand for quick and accurate computation of these numerical problems in the nature. Therefore, we designed the parallel algorithms of these problems, evaluated their performances, and also compared with the existing sequential algorithms. We start with the interpretation of sequential algorithm of the problems; identify the data independent and independent operations. Using the facts we develop new parallel algorithms, analyze their complexity and also evaluate their performances on the shared memory multiprocessor systems.

The first problem of the work is to design the parallel algorithms for the solution of system of linear equations. Gauss method is a well known direct method for solving system of linear equations, the coefficient matrices of which are dense. We present the parallel algorithms of Gauss elimination and Gauss Jordon method for finding solution of system of linear equations. We theoretically analyze and present the complexity of parallel algorithms. We also evaluate the performance of parallel algorithms through their execution time. We compared the execution time of parallel algorithms with their sequential and found that the performance of parallel algorithms is better than sequential algorithms.

The second problem of the work is to design the parallel algorithms for the solution of definite integral. Numerical integration is the method of finding the numerical value of an integral $\int_a^b f(x)\,dx$. But if $f(x)$ is complex continuous function or $f(x)$ does not have a known analytical form then it is very difficult to find efficient solution of the problems using known sequential algorithms. Newton-cotes method is a well known
method for numerical integration. We present the parallel algorithms of \textit{Simpson’s 1/3} and \textit{Trapezoidal} method. In the design of parallel algorithms, we considered the \textit{integration formula level} of parallelization which is the most efficient method of parallelization. We theoretically analyze and present the complexity of parallel algorithms. We also evaluate the performance of parallel algorithms through their execution time. We compared the execution time of parallel algorithms with their corresponding sequential algorithms. We test the algorithms on three mathematical functions and found that the performance of parallel algorithms is better than the sequential

Third problem of the work is to design the parallel algorithm of \textit{polynomial interpolation} i.e. for given \( n \) points in the plane, \((x_k, y_k), k=1,\ldots, n\), with distinct \( x_k \)'s, there is a unique polynomial \( p(x) \) of degree less than ‘\( n \)’ whose graph passes through the given points. In this work we considered the method of \textit{Lagrange’s interpolation}, which is a polynomial \( p(x) \) of degree \( \leq (n-1) \), passes through the given points and defined as \( p(x) = \sum_{i=1}^{n} y_i l_i(x) \), where \( l_i \) is Lagrange’s polynomial and \( l_i(x) = \prod_{j=1 \atop j \neq i}^{n} \frac{x-x_j}{x_i-x_j} \).

We present the parallel algorithm of \textit{Lagrange’s interpolation} in which Lagrange’s polynomials are computed concurrently in parallel on different processors. We theoretically analyzed and presented the complexity of parallel algorithm. We also evaluate the performance of parallel algorithm through their execution time. We compared the execution time of parallel algorithm with their sequential algorithm and found that the performance of parallel algorithm is significantly better than the existing sequential algorithm.

Comparative analysis of proposed parallel algorithms with their existing sequential has been made. The results of studies show that performance of parallel algorithms are efficient than the sequential and also it is better than other existing studies. Thus the proposed parallel algorithm can be successfully applied to the various applications where quick and accurate computation is required.
We have used the tools *Intel Parallel Studio* and *Intel C++* compiler for the implementation of all the algorithms. We write program in C++ which has support of *OpenMP*. *OpenMP* (open multiprocessing) is an open standard for platform neutral parallel programming and supports *data* and *task parallelism*. 