Chapter 1

Review of Previous Literature

1.1 Parallel Algorithms for Boundary-Value Problems

A number of papers have considered parallelizing the numerical solution of two-point boundary value problems (BVPs) for ordinary differential equations (ODEs). Most of these consider parallelizing the matrix computations which arise when TPBVP is replaced by its equivalent finite difference scheme.

A vector implementation of an ODE code for multi-point boundary value problems has been developed in [7]. When applying the multiple-shooting code BNDSCO to boundary-value problems (BVPs) on vector computer, a vectorized version of a numerical integration method for ordinary differential equations (ODEs) has to be developed. Since the dominating part of the computing time is spent in solving a series of initial value problems (IVPs), the computing time can be considerably reduced by the use of a vectorized Runge-Kutta method. The method discussed in [7], is applicable for BVPs with nearly all kinds of discontinuities in the solution as well as in the right-hand side at a finite number of interior points and is therefore applicable for many types of optimal control problems (OCPs) with constraints.

For a large group of typical BVPs the linear algebra takes between 25% to 50% of the overall time of finding the solution. Most of the remaining parts of the BVP are either parallelizable or BVP leads to linear system of almost block diagonal (ABD) nature. The proposed approaches to BVP solution can be classified according to the granularity of parallelization they represent. Wright and Pereyra designed an
algorithm for solving ABDs on vector computers, but used an Alliant parallel computer which performed automatic fine-grained micro-parallelization. A medium granularity of parallelization is presented by the block Gaussian elimination algorithm based on the level 3 BLAS as discussed in [9]. A higher, but still medium, granularity of parallelization is represented by solving ABDs as block tridiagonal. Ascher and Chan [10] and others utilize an odd-even reduction to solve such a system. Paprzycki and Gladwell [11] developed an algorithm for solving ABDs on parallel computers.

In [12], Paprzycki and Gladwell present the possibility of solving two-point linear ODEs using a ‘whole problem’ level of granularity. The method proposed is similar to domain decomposition and is based on ‘chopping algorithm’. The parallel chopping algorithm is based on a set of specially designed meshes. Experiments show that this approach has the usual disadvantages of macroscale parallelization. It also suffers from the crudeness of the chopping procedure. The methods described in [9] and [10] may be used (independently or combined) to improve the performance of the BVP solver. They can also be used to divide work in a more flexible way between processors.

Despite the deficiencies in algorithmic design and the overload of parallel computation, quite impressive speedups are achieved when using a large enough number of processors. It seems clear that some of the speedup is due to deficiencies in COLNEW’s strategy [13,14] for mesh redistribution and to a poor initial mesh choice. But the parallel program could also be speeded up by an improved main meshpoint strategy, by performing single and double mesh solution in parallel and, possibly, by treating all active subintervals simultaneously. Katti and Goel [15] developed a parallel algorithm for the class of two-point boundary
value problems \( y'' = f(x, y) \), \( y(0) = A, \ y(1) = B \) with \( \frac{\partial f}{\partial y} \geq 0 \) and \( \frac{\partial f}{\partial y} \) continuous on \((0,1) \times (-\infty, \infty)\). Using an idea similar to [12], \([0, 1]\) is divided into \(p\) different divisions, each division consisting of \(N\) or \((N+1)\) (\(N\) small) unequal intervals. A high-order finite difference scheme developed for general non-uniform mesh was applied to the above class of TPBVP's on each of the \(p\) divisions and leads to an \(N \times N\) or \((N-1) \times (N-1)\) system of linear or nonlinear equations which is solved on \(p\) processor (\(p\) a power of 2) simultaneously.

In chapter 2, we consider the extension of the parallel algorithm given in [12] to the class of two point boundary value problems \( y^{(iv)} = f(x, y) \), \( y(0) = A_1, \ y''(0) = A_2, \ y(1) = B_1, \ y''(1) = B_2 \) with \( \frac{\partial f}{\partial y} \leq 0 \) and \( \frac{\partial f}{\partial y} \) continuous on \([0, 1] \times (-\infty, \infty)\). As in [12], the interval \([0, 1]\) is divided into \(p\) different divisions, each division consisting of \(N\) or \((N+1)\) unequal intervals. A new fourth order finite difference scheme is developed for general non-uniform mesh and is applied to the above class of TPBVP's on each of the \(p\) divisions. This leads to the solution of \(N \times N\) or \((N+1) \times (N+1)\) system of linear or non-linear equations which is solved on \(p\) processors (\(p\) a power of 2) simultaneously. The solution of the original problem is then obtained at \(n = Np\) equally spaced abscissas on \([0, 1]\). We also extend parallel mesh chopping algorithm for the solution of class of boundary value problems involving Elliptic Partial Differential Equations (PDEs).
1.2 Parallel Algorithms for Initial-Value Problems

Various methods have been developed for the parallel solution of initial value problems. One such method is front broadening and embedded parallelism, as discussed in [1]. It reviews and develops (i) broadening of the computation front (BCF) techniques, and (ii) block implicit (BI) methods to speed up the numerical solution of IVPs in ODEs by using MIMD computing systems. In particular, consideration is given to the performance, accuracy and stability characteristics of the techniques and possible extension of the BCF methods to improve the accuracy. It has been shown that it is not possible to get fourth and higher order parallel explicit Runge-Kutta (R-K) methods by using the BCF techniques. This paper also characterizes some of the drawbacks of BCF and considers two further techniques to exploit the parallelism embedded in BI methods. Parallel shooting is another method used for the parallel solution of initial value problems as discussed in [2]. It considers the implementation of parallel shooting for solving IVPs using a multiprocessor computing system. The paper [2] shows the significance of these methods and demonstrates their effectiveness in controlling the numerical stability of the initial value algorithms. Multiple shooting techniques for the numerical solution of boundary value problems (BVPs) in ODEs were developed by Keller [3] in order to control the stability of the sequential numerical solution since the solution of some BVPs is unstable when solved by simple shooting techniques. The former techniques have been developed further by many other researchers concerned to improve the stability of the sequential solutions. More recently researchers have sought to adopt such techniques for implementation on vector or parallel processors in order to reduce solution time. Khalaf and Hutchinson [4] proposed the modification of multiple shooting for solving IVPs on fully parallel computing systems. In [2], the combination of following ideas have been done: (i) Consider the IVPs as one-point BVPs. (ii) Decompose the integration interval. (iii)
Convert the forward integration processes of the multiple shooting techniques to backward integration processes.

Boundary value techniques for solving initial value problems have been proposed by several authors. In [6], some boundary value methods are considered and the notion of BV-stability has been introduced. These methods give some advantages with respect to usual initial-value methods for ODEs, such as the control of the global error rather than just the local one, the possibility to have A-stable methods of order greater than 2, the possibility to use the same method for stable and unstable problems and the possibility to be implemented efficiently on parallel computers, especially for stiff differential equations.

In [5], detailed analysis of Boundary value methods based on linear two-step methods and their stability properties has been done. The choice of two-step methods is essentially motivated by parallel implementation reasons, since BVM methods lead to block tridiagonal systems which can be solved efficiently on parallel computers.

In chapter 3, we have developed Runge-Kutta method for non-uniform mesh of 3rd order, to the class of initial value problems (3.1). The extension of the same analysis has been applied to the system of differential equations (3.21). Numerical illustration and theoretical estimates of the method has been done both for initial value problem and system of differential equations.
1.3 PARALLEL ALGORITHMS FOR SOLVING TRIDIAGONAL SYSTEMS

It is well known that a class of tridiagonal matrices $T$ can be factored as the product of a lower bidiagonal matrix and an upper bidiagonal matrix. Then the problem of solving $Tx = y$ is reduced to solving two bidiagonal linear systems. Since this leads effectively to a recursion formula for the unknowns this algorithm by itself is not well suited for parallel or vector computing. Algorithms that reduce the recursion problem to smaller recursion problems, or independent recursions, include recursive doubling [16,17] and cyclic reduction [18,19]. The effect of these techniques on the efficiency of the solution process on the vector computer CRAY-1 and CIBER 205 has been considered in [20], and for more general vector architecture in [21]. Recursive doubling as well as cyclic reduction increases the computational complexity, so that there is a price to be paid for the gain in parallelism. In [22], an alternative decomposition for a tridiagonal matrix has been considered. The tridiagonal matrix considered has the property that the decomposition as well as the subsequent solution process can be done in two parallel parts. This decomposition is equivalent to the two-sided Gaussian elimination algorithm that has been discussed by Babuska [23]. The computational complexity of this alternative decomposition is the same as for the standard decomposition and a remarkable aspect is that it often leads to slightly more accurate solutions than the standard process does. The algorithm can be combined with recursive doubling or cyclic reduction in order to increase the degree of parallelism and vectorizability.

Ruggiero and Galligani [24] consider a new form of the arithmetic mean method for solving large block tridiagonal linear systems. The iterative method converges for systems with coefficient matrices that are symmetric positive definite or positive real or irreducible L-matrices with a strong diagonal dominance. When the coefficient matrix is symmetric positive definite, an additive preconditioner for the
conjugate gradient method is derived. Both the iterative method and the
preconditioner are very suitable for parallel implementation on a multivector
computer.

The solution of tridiagonal linear systems is the kernel of a large variety of
problems. Main applications derive from the discretization of partial differential
equations in one dimension or scalar ordinary differential equations by using, for
example, boundary value methods. For this reason, a great deal of attention has
been focused on special methods to efficiently solve tridiagonal linear systems on
parallel computers.

The cyclic reduction is one of the most interesting methods both because it does
not need additional memory and because of its easy vector implementation. It was
introduced, together with the partition methods, to provide parallel solvers more
efficient than the LU factorization [25]. The stability analysis proves that the cyclic
reduction algorithm is stable for diagonally dominant matrices, and that the ratio
between the off-diagonal and the main diagonal elements tend to zero when the
reduction goes on [26]. Vector implementations of the cyclic reduction, also
provided for a large variety of related problems (block tridiagonal and banded
systems), are quite simple and efficient.

In [27], a parallel version of the cyclic reduction algorithm on a hypercube has
been considered. The basic idea is to divide the original system into subsystems
which are solved almost independently. Communications are only needed for
solving a tridiagonal system whose dimension is proportional to the number of
processors. By utilizing a hypercube as topology of interconnection among the
processors, the solution of this subsystem requires communication among
neighbouring processors. The number of synchronizations is proportional to the
logarithm of the number of processors. Optimized cyclic reduction for the solution
of Linear tridiagonal systems on parallel computers has been developed [28]. It
minimizes synchronizations and provides stability properties. The idea used is to delay communications among the processors until the solution of a reduced tridiagonal system of dimension which depends on the number of processors. The communication delay is obtained by considering opportune partitioning of the coefficient matrix.

In chapter 4, we develop an algorithm for the parallel solution of quasi tridiagonal systems. This is a generalization of an algorithm due to Kowalik et al. (1984) for tridiagonal systems. Such tridiagonal systems arise when approximating a class of two point boundary value problems having periodic boundary conditions.

1.4 PARALLEL ALGORITHMS FOR THE SOLUTION TO THE NAVIER-STOKES EQUATIONS

The National Science Foundation has identified a series of problems whose solutions would have a great economic or scientific impact worldwide. Supercomputational techniques appear to be the sole alternatives to achieving any significant advances in this context. The aforementioned problems, widely known as Grand Challenges, include the following:

- The behaviour of a moving object in a bulk fluid, either laminar or turbulent; new techniques for the development of aircraft.

- Meteorological forecasts, overall climatological changes, carbon dioxide dynamics, destruction of the ozone layer

- Turbulence: Structural analysis, transition to turbulence and turbulent flow modeling.

The starting point for all the above problems is the Naviers-Stokes equations (NSE), which describe the motion of "real" fluids. These are non-linear partial
differential equations (PDEs), so their exact solutions are only known for a fairly small number of simple problems. Consequently, their description relies on numerical techniques. The problem is rather complex; the numerical solution of NSE calls for the joint effort of at least numerical analysts and experts in computation sciences. The high complexity of these problems is reflected in the fact that, in practice, the numerical treatment of all interesting situations is not feasible by using sequential algorithms.

In [29], a parallel algorithm for the solution of NSE has been developed which is based on the splitting of the original problem in such a way that each subproblem is simultaneously executed by different processors. It is based on the splitting of the two main difficulties involved, the presence of non-linear terms and the zero divergence condition.

Parallel processing Algorithms for the Finite Difference Solution to the Navier-Stokes Equations has been developed in [30]. Numerical Solution involving finite difference representations of these equations usually consists of solving large sparse matrix equations. These matrix equations can be recast into M smaller coupled matrix equations amenable to solution by using M multiple computer processors operating in parallel. Parallel algorithms for solving these equations are developed and evaluated in terms of computational speed against conventional solutions on a serial machine. Timing studies are performed to assess the efficiency of these methods, and to determine the optimum number of parallel processors for these applications.

In chapter 4, we have developed highly efficient parallel algorithm for finite difference solution to Navier - stokes Equation on a hypercube.
REFERENCES


