Abstract

Many engineering and applied science problems required the solutions of time dependent equations with nonlinear features. Usually a time-marching scheme, such as Euler's method, Runge-Kutta methods, multi-step methods, etc., with time step length restrictions is employed in any temporal integration procedure. Parallelisation of the time stepping becomes difficult and it is almost impossible to achieve a distributed/parallel algorithm that is able to yield a de-coupling of the original problem. On the other hand there are also many problems which require solution details not at each time step of the time-marching scheme, but only at a few crucial steps and the steady state. Therefore effort in finding fine details of the solutions using a temporal integration procedure with many intermediate time steps is considered being wasted. This chapter presents the idea of inducing parallel properties into an otherwise sequential transient problem. A number of transformation methods and their relations to the possibility of providing concurrency in the solutions of partial differential equations are examined. Several examples related to the present approach are discussed, including convection diffusion problems and option pricing problems. In some cases numerical experiments are also included to support the concept. In other cases implementation issues are included to support the concept. Finally a two-level time domain parallel algorithm is presented with numerical tests from a nonlinear parabolic problem to demonstrate the viability of the method.

Keywords: Temporal integration, distributed algorithms, parallel algorithms, transformation methods.

1 Introduction

Many engineering and applied science problems required the solutions of nonlinear diffusion equations where the nonlinear feature usually comes with the material properties or the conductivity. In the case of unsteady problems a time-marching scheme, usually with time step length restrictions, is employed in any temporal integration procedure. These restrictions are usually due to stability criteria of explicit schemes or truncation errors of implicit schemes in approximating the temporal derivatives. Computing time of such numerical methods inevitably becomes significant. On the other hand fine grain parallelisation of time stepping becomes difficult and it is almost impossible to achieve a distributed/parallel algorithm that is able to yield a de-coupling of the original problem. There are also many problems which require solution details not at each time step of the time-marching scheme, but only at a few crucial steps and the steady state. Therefore effort in finding fine details of the solutions using many intermediate time steps is considered being wasted. Such effort becomes significant in the case of nonlinear problems where a linearisation process, which amounts to an inner iterative loop within the time-marching scheme, is required. It would be a significant save in computing time when the linearisation process and the time-marching scheme can both be done in parallel. The main objective of the present work is to remove the time stepping and to combine the resulting algorithm with parallel/distributed computers based on certain mathematical tools. The other objective is to examine the use of two different time stepping size with the combination of a transformation method and a temporal iterative method leading to a 2-level time domain parallel algorithm.

A discussion is given of a number of transformation methods and their relations to the possibility of providing concurrency in the solutions of partial differential equations. Section 2 discusses the general algorithm for temporal integration methods, including a brief overview of several other attempts in the parallelisation of the temporal axis. Section 3 investigates three major types of transformation methods include the method of dimensional analysis, general stretch transformations, and Laplace transformation. The concept is introduced by using an easier description based on a one spatial dimension problem. Section 4 discusses several examples related to these transformations, including convection diffusion problems and option pricing. Implementation issues are also included to highlight the distributed properties. More details of numerical experiments related to the Laplace transform method is demonstrated in Section 5 by using a typical example of option pricing with nonlinear volatility. A two-level temporal mesh is examined by the alternative use of a coarse and a fine temporal meshes for nonlinear problems. Finally discussions and conclusions are presented in Section 6.

2 Temporal Integration Methods

A time marching scheme is a temporal integration method usually applied to transient problems. These problems may or may not be nonlinear. Consider the time-dependent problem denoted by

$$\frac{\partial u}{\partial t} = \mathcal{F}(u, \underline{x}, t) \tag{1}$$

where $\mathcal{F}$ is a nonlinear functional dependant on u, the derivatives of u, the spatial coordinates $\underline{x}$ and the time. Suppose initial conditions are provided at $t = 0$ and a temporal step size of δt is chosen to be used in an integration method. Let $t_k = k\delta t$, $k \geq 0$, and $u^k = u(k\delta t, \underline{x})$. An explicit scheme is defined by

$$\frac{u^{(k+1)} - u^{(k)}}{\delta t} = \mathcal{F}(u^{(k)}, \underline{x}, t_k) \tag{2}$$

may be re-arranged as

$$u^{(k+1)} = u^{(k)} + \delta t \mathcal{F}(u^{(k)}, \underline{x}, t_k) \tag{3}$$

Equation (2) provides a simple update formula such that $u(\underline{x}, t_{k+1})$ depends solely on the knowledge of $\mathcal{F}$ at $t = t_k$ and can be evaluated readily. Note that there is a stability criterion relating δt and the choice of the mesh size in the spatial discretisation imposing sometimes very small step sizes along the temporal axis. Similarly an implicit scheme can be written as

$$\frac{u^{(k+1)} - u^{(k)}}{\delta t} = \mathcal{F}(u^{(k+1)}, \underline{x}, t_{k+1}) \tag{4}$$

which can then be re-arranged as

$$u^{(k+1)} - \delta t \mathcal{F}(u^{(k+1)}, \underline{x}, t_{k+1}) = u^{(k)} \tag{5}$$

Equation (5) provides an implicit formula for $u(\underline{x}, t_{k+1})$. Either a linear system or a nonlinear system, depending on the functional $\mathcal{F}$, is required to be solved. Although there is no stability criterion governing the step size δt and the spatial discretisation, it is important to note that it is impossible to use an unreasonably large δt in Equation (5). In essence it is impossible to compute the final step simply based on the initial data.

From the above discussion for both methods of temporal integration, it is therefore easily see that concurrent computation of u at many different times is not possible using a temporal marching scheme. Attempts have been made by several researchers in this direction. For examples Miranker and Liniger proposed a family of parallel Runge-Kutta methods [1]; Lions, Maday and Turinici proposed the 'parareal' algorithm in time discretisation [2]; and more recently Gander and Vandewalle provided an anlysis of the parareal algorithm for parallel temporal integration [3]. These concepts largely based on partitioning the temporal axis into a number of non-overlapped sections known as time subdomains and the same time-dependent problem is considered in each of these time subdomains. The partitioning of the temporal axis resembles very similar to a multiple shooting method [4]. Methods developed within this context have been further enhanced by the use of a coarser temporal partitioning in conjunction with a finer temporal partitioning based on the concept of a multigrid correction technique.

Another popular way of introducing temporal parallelism into time dependent problems is knwon as waveform relaxation methods. A good review may be found in [4]. The idea is based on a dynamic iterative concept or Picard-Lindelöf iteration which was originally applied to systems of algebraic equations. The iterative method was then extended to handle the system of ODE's resulted from a spatial discretisation of the time dependent problem. It should be pointed out that there is possible time lagged and overlapped effort occur if one attempts to do simultaneous updates at different times based on a parallel temporal integration scheme. In other words the temporal dependence, i.e. the intrinsic sequential computation, component remains there.

3 Transformation Methods

The idea of a transformation method was used to transform a given time dependent problem based on the use of certain parameters resulting to a simplified equation dependent on the parameters only. Such tool was very useful in obtaining analytic solutions to many engineering problems. It was also very useful in studying the long term behaviour of many time dependent problems Like most parallel algorithmic research work in the early days of parallel computing, researchers used existing numerical algorithms, such as the Jacobi iterative method for a system of linear equations [5], in order to introduce data parallelism into a piece of sequential algorithm. While Gauss-Seidel iterative method was not able to be parallelised, alternative or equivalent ways such as the red and black iterative method [5] was used to substitute the Gauss-Seidel iterative method in a data parallel environment. These methods are known as parallelisation at the discretisation level. In this Section existing mathematical tools are re-cycled and re-examined in order to induce parallel/distributed properties into the governing time-dependent equation describing engineering applications so that computation along the temporal axis, which is otherwise sequential, results to a set of computationally independent parametric differential equations. These independent parametric differential equations are therefore decoupled and may then be computed in a concurrent or distributed manner. Note that such parallelism occurs at the level of the continuous problems. There are two ways of obtaining such parametric equations, i.e. similarity transforms and Laplace transforms. For simplicity and ease in presentation of the algorithms, only one spatial dimension is used in the mathematical formulation.

3.1 Similarity Transforms

Consider the nonlinear time-dependent problem as given below.

$$\frac{\partial u}{\partial t} = \mathcal{F}(u, x, \partial_x, \partial^2_{xx}, \sigma)u(x,t) \tag{6}$$

where $\mathcal{F}$ is a nonlinear operator involving partial differential operators ∂_x and ∂^2_{xx}, $u = u(x,t)$, and σ encapsulates the physical quantities involved in each particular problem.

The main idea in this class of methods is to identify certain groupings of the independent variables so that the solution of a given time-dependent PDE may be written in terms of a function of the grouped independent variables. Instead of writing the solution in terms of the independent variables it is more convenient to use a suitable grouping of independent variables leading to a new transformed independent variable.

Suppose there exists a transformation of coordinates based on the grouping of independent variables and the physical quantities, say $\eta = \eta(x,t,\sigma)$, and a similarity solution based on the transformation, say $u(x,t) = f(\eta)$. Substituting the similarity solution $f(\eta)$ into Eqn (6) leads to the nonlinear ODE [6] :

$$\frac{\partial \eta}{\partial t}\frac{df}{d\eta} = \mathcal{G}(f, \frac{\partial \eta}{\partial x}\frac{d}{d\eta}, \frac{\partial^2 \eta}{\partial x^2}\frac{d^2}{d\eta^2})f \tag{7}$$

Here $\mathcal{G}$ is a nonlinear operator. There are two ways of obtaining such combination of independent variables and the physical quantities, namely the dimensional analysis and the stretch transformation.

First, the dimensional analysis involves writing a similarity solution in the form

$$u(x,t) = f(x, t, u_1, \sigma) \tag{8}$$

where u_1 denotes some boundary conditions imposed on Equation (6), and application of the dimensional analysis leads to

$$[u] = [x]^a[t]^b[u_1]^c[\sigma]^d \tag{9}$$

where a, b, c and d are parameters to be determined. In general it is possible to reduce these parameters to one parameter which gives a relation between the independent variables and the physical quantities, i.e. $\eta = \eta(x, t, \sigma, a, b, c, d$. Hence Equation (9) can be written in the form $u(x,t) = f(\eta)$ which satisfies the ODE in Equation (7). Solving Equation (6) amounts to obtain the solution of Equation (7).

Second, stretch transformations involve writing the independent variables and the dependent variable as below:

$$x = e^a X; \;\; t = e^b T; \;\; u = e^c U \tag{10}$$

Substituting the stretch transformation into Equation (6), under certain conditions of the nonlinear operator $\mathcal{F}$, one arrives

$$e^{r_1(a,b,c)}\frac{\partial u}{\partial t} = e^{r_2(a,b,c)}\mathcal{F}(u, x, \partial_x, \partial^2_{xx}, \sigma)u(x, t) \tag{11}$$

where $r_1(a, b, c)$ and $r_2(a, b, c)$ are functions of the three parameters a, b and c. This method requires Equation (6) to be invariant under the transformation. In other words one requires $r_1(a, b, c) = 0$ and $r_2(a, b, c) = 0$ which leads to two relations between the three parameters such that Equation (6) is invariant. These two relations can be easily reduced to be a suitable grouping of the independent variables and a suitable similarity form of the solution. Substituting the similarity solution into Equation (6) reduces the PDE into an ODE involving the similarity function.

In both cases, suitable boundary conditions are needed to be imposed on Equation (7). This can be achieved by converting the relevant boundary conditions of the original equation to the transformed equation. Examples and discussion on nonlinear problems are presented in Section 4.

3.2 Laplace Transforms

This class of transformation method is based on the concept of the Laplace transform and its numerical inverse. Usually a linear problem becomes easier to handle computationally. The linear time-dependent problem with one spatial dimension is considered here.

$$\frac{\partial u}{\partial t} = F(x, \partial_x, \partial^2_{xx}, \sigma)u(x, t) \tag{12}$$

subject to initial condition of $u(x, 0)$ and suitable boundary conditions. Here F is a linear operator involving partial differential operators ∂_x and ∂^2_{xx}, $u = u(x, t)$, and σ encapsulates all physical quantities, which do not depend on u, involved in each particular problem. Apply the Laplace transform to u leads to

$$L(u) \equiv \int_0^\infty e^{-\lambda t}u(x, t)dt = U(\lambda; x) \tag{13}$$

such that the dependence on time is now converted to the dependence on the parameter $\lambda \in \{\lambda_p = p\frac{ln2}{T} : p = 1, 2, ..., m\}$, which is a finite set of transformation parameters. Similarly apply the Laplace transform to Equation (12) leads to

$$\lambda_p U(\lambda_p; x) - u(x, 0) = F(x, \partial_x, \partial^2_{xx}, \sigma)U(\lambda_p; x)) \tag{14}$$

subject to suitable boundary conditions derived from using the Laplace transform of the given boundary and initial conditions as well. The set of parametric solutions $\{U(\lambda_p; x) : p = 1, 2, ..., m\}$ may be obtained by solving Equation (14) and each element of the set may be solved independently from the others.

4 Decoupled Parametric Systems

4.1 Similarity Transformation Methods

Consider the linear time-dependent diffusion problem given by

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2} \tag{15}$$

subject to $u(0,t) = u_0$ and some other boundary conditions at the other end. It is easy to show, in the case of dimensional analysis, the parameters in Equation (9) are given by $c = 1$ and $b = d = -a/c$. Therefore it is possible to choose $\eta = \frac{x}{\sqrt{t\sigma}}$ and $u(x,t) = u_0 f(\eta)$. With such choice Equation (7) for the problem given by Equation (15) becomes

$$\frac{d^2 f}{d\eta^2} + \frac{1}{2}\eta \frac{df}{d\eta} = 0 \tag{16}$$

It is also easy to show, in the case of a stretch transformation, the invariant of Equation (6) requires the parameters in Equation (11) to be $2a = b$. The requirement of invariant of Equation (6) leads to $\eta = \frac{x}{\sqrt{t}}$ and the form of solution is $u(x,t) = t^{c/b} f(\eta)$. Here c vanishes when the boundary condition is considered, the problem defined in Equation (15) becomes

$$\sigma \frac{d^2 f}{d\eta^2} + \frac{1}{2}\eta \frac{df}{d\eta} = 0 \tag{17}$$

Suppose one requires the numerical solution of u at various different times on a set of fixed grid points along the x-axis defined by the discretised mesh δx, the corresponding mesh size to be used in the transformed system is $\delta\eta = \delta\eta(\delta x, t, \sigma)$. It is possible to simultaneously compute as many time dependent solutions as possible at various different times, i.e.

$$\delta\eta_k = \delta\eta(\delta x, t_k, \sigma) \tag{18}$$

It should be noted that computations at different times can be performed simultaneously and that each independent calculation depends on the value of t_k which affects the corresponding mesh size $\delta\eta_k$ used only at that particular time. In other words there are as many problems, each defined by Equation (7) and is decoupled from the others, as the number of times at which solutions are required. Each discretised problem of Equation (7) has a peculiar mesh size defined by $\delta\eta_k$ compare with other discretised problems. Therefore there is a load balancing issue to be considered in actual implementation if one assumes each discretised problem is being handled by one computer. However there is no reason why one should not use more than one computer for one discretised problem in order share more work load leading to a synchronus algorithm. The algorithm below shows the decoupled computation of $u(x, t_k)$ for linear problems.

```
*Decoupled computation of* $u(x,t_k)$ *for* $k = 1, 2, ...,$ (linear problems):-
  Distribute $k = 1, 2, ...,$
      Compute mesh size: $\delta\eta_k = \delta\eta(\delta x, t_k, \sigma)$;
      Find $f : \sigma\frac{d^2f}{d\eta^2} + \frac{1}{2}\eta\frac{df}{d\eta} = 0$;
      $u(x,t_k) := {t_k}^{c/b}f$;
  End Distribute
```

While transformation methods are effective for linear problems in achieving a grouping of independent variables, it is not always the case for some nonlinear problems. The above similarity transformation methods may be extended to handle nonlinear problems in a robust computational algorithm but not resulting to an analytical method. The latter case requires other suitable transformation methods leading to a linear problem before applying the similarity transformation. Here a linearisation technique is adopted in the computation. Consider the nonlinear time-dependent diffusion problem given by

$$\frac{\partial u}{\partial t} = \sigma(u)\frac{\partial^2 u}{\partial x^2} \tag{19}$$

subject to the initial condition $u(x, 0) = u_0$ and some other boundary conditions at the other end. Here σ is a function of u. Let $\bar{u}$ be an approximation of $u(x,t)$ at time t. $\bar{u}$ is updated in every step of an inner iterative update process. Each step of the iterative process involves obtaining the solution to the quasi-linear problem

$$\frac{\partial u}{\partial t} = \sigma(\bar{u})\frac{\partial^2 u}{\partial x^2} \tag{20}$$

The decoupled computation described above may be used with an inner iterative loop to handle the nonlinearity. Let $u^{(n)}$ and $f^{(n)}$ be the nth iterative approximations to $u(x,t)$ and $f(\eta)$ respectively. Algorithm C below shows the procedure for incorporating the iterative update process within the parallel/distributed computation. Note that the counter n may be different for different values of k.

```
Algorithm C: Iterative coefficient - Transformation methods.
Initial values: $n = 0$; $u^{(0)}(x,t_k)$, $k = 1, 2, ...,$, are given.
  Distribute $k = 1, 2, ...,$
      Do {
          Compute $\sigma(\bar{u})$ such that $\bar{u} := u^{(n)}(x,t_k)$;
          Compute mesh size: $\delta\eta_k = \delta\eta(\delta x, t_k, \sigma)$;
          Find $f^{(n+1)} : \sigma(\bar{u})\frac{d^2f^{(n+1)}}{d\eta^2} + \frac{1}{2}\eta\frac{df^{(n+1)}}{d\eta} = 0$;
          $u^{(n+1)}(x,t_k) := {t_k}^{c/b}f^{(n+1)}$;
          $n := n + 1$;
      } Until $\|\sigma(u^{(n)}(x,t_k) - \sigma(\bar{u})\| < \epsilon$
  End Distribute
```

4.2 Laplace Transformation Methods

While similarity transformation methods are able to handle many time-dependent diffusion type of equations they have limitations in dealing with diffusion-radiation type of problems. Laplace transforms is an alternative technique. In the case of Laplace transforms the transformed variable is $U(\lambda; x)$ for one dimensional problems as the one described in Equation (12). Suppose one requires the solution of u at a given time, say $t = T$, one requires to compute the set of parametric solutions $\{U(\lambda_p; x) : p = 1, 2, ..., m\}$ each satisfies Equation (14). This can be done simultaneously for every value of p provided there are m computers available. Early numerical tests on diffusion problems in porous media can be found in [7]. A comprehensive study on diffusion-convection-radiation problems can be found in [8] with applications to option pricing. Finally one requires to compute an approximate inverse Laplace transform in order to retrieve the solution $u(x, t)$. One such inverse Laplace transform method was developed in [9] using the weighted formula

$$u(x, T) \approx \frac{\ln 2}{T} \sum_{p=1}^{m} w_j U(\lambda_p; x) \tag{21}$$

where $w_p = (-1)^{m/2+p} \sum_{i=(1+p)/2}^{\min(p,m/2)} \frac{k^{m/2}(2i)!}{(m/2-i)!i!(i-1)!(p-i)!(2i-p)!}$ is known as the weighting factor. In other words the solution $u(x, T)$ may be retrieved as a weighted sum of the set of parametric solutions. The algorithm below shows the decoupled computation of the solution in the Laplace space.

```
*Decoupled computation of* $U(\lambda_p; x)$ *for* $p = 1, 2, ..., m$ (linear problems):-
    Distribute $p := 1$ to $m$
            Find $U(\lambda_p; x)$ from (14), i.e.
            $\lambda_p U(\lambda_p; x) - u(x, 0) = F(x, \partial_x, \partial^2_{xx}, \sigma)U(\lambda_p; x))$;
    End Distribute
    Compute $u^{(k)}(x, T)$ using inverse Laplace in Equation (21);
```

Note that the last part of the above algorithm concerns the inverse Laplace transform of the solutions obtained in the Laplace space. The inverse computation is an intrinsic sequential calculation and cannot be distributed or parallelised. Therefore this becomes a kind of overheads to the distributed computation. Unlikely similarity transformation methods where the retrieval of the solution of the original problem can be done in parallel.

Similar techniques of using an iterative loop, in the present case an outer loop instead of an inner loop as used in similarity transformation methods, may be applied for nonlinear problems in the case of Laplace transforms. Consider the nonlinear diffusion convection and radiation problem which may be used to describe option pricing:

$$\frac{\partial u}{\partial t} = \alpha(u,x)\frac{\partial^2 u}{\partial x^2} + \beta(x)\frac{\partial u}{\partial x} - \gamma u \tag{22}$$

where α is a function of u and x, β is a function of x, and γ is a constant. Equation (22) is a typical PDE used in pricing options. The Laplace transform method may be applied to a linearised problem. Let $u^{(n)}$ denotes iterative approximation to $u(x,T)$ where T is the time when fine details of the solution is needed. The nonlinear problem in Equation (22) may be linearised and written as

$$\frac{\partial u^{(n+1)}}{\partial t} = \alpha(u^{(n)},x)\frac{\partial^2 u^{(n+1)}}{\partial x^2} + \beta(x)\frac{\partial u^{(n+1)}}{\partial x} - \gamma u^{(n+1)}$$

to which Laplace transformation can be applied easily. Similarly the stopping criterion for this nonlinear iteration can be chosen as $\|\alpha(u^{(n+1)},x) - \alpha(u^{(n)},x)\| < \epsilon$. As the Laplace transform method requires to retrieve the solution by means of an inverse Laplace, some further numerical experiments of nonlinear parabolic equations are given in Section 5. Note that the linearisation process involved in the above description is very crude and time step length should be brought back in order to handle highly fluctuated solution of u. Examples are shown in Section 5.

5 Some Numerical Tests

A time dependent non-linear parabolic problem and a nonlinear European option pricing problem are used to illustrate and compare the Laplace tranformation method and a temporal integration method. Two methods of handling the nonlinearity are demonstrated in this section. A novel two-level time-domain is then introduced by combining the use of the inverse Laplace method and a temporal integration method. Numerical experiments are provided to examine the efficiency and accuracy of the new algorithm. As the method can be easily generalized to higher dimensional problems, only problems with one spatial dimension is used in this section for illustration purposes.

5.1 A Nonlinear Parabolic Problem

The model problem treated in this section is the nonlinear time dependent diffusion equation,

$$\frac{\partial u}{\partial t} + g(x,t) = K(u)\frac{\partial^2 u}{\partial x^2} \quad \in (a,b)\times(0,T] \tag{23}$$

subject to suitable boundary conditions, defined at $x = a$ and $x = b$, and initial conditions. Here the thermal conductivity, $K(u)$, is a given function of u, and g is a given function of x and t.

The conductivity of the model problem, defined in the time interval (t_i, t_{i+1}), is computed by using an approximation $\bar{u}$, which is updated in every step of an iterative update process within the given time interval. Each step of the iterative update process involves obtaining a numerical solution to the quasi-linear problem

$$\frac{\partial u}{\partial t} + g(x,t) = K(\bar{u})\frac{\partial^2 u}{\partial x^2} \quad \in (a,b) \times (t_i, t_{i+1}] \tag{24}$$

Here $t_i = i\delta t$, $i = 0, 1, \ldots$, and δt is the time step length of a temporal integration method. Let $u^{(n)}(x, t_{i+1})$ and $u^{(n)}(x, t_i)$ be the numerical solutions of Equation (23) at $t = t_{i+1}$ and $t = t_i$ respectively. The iterative update process to obtain the numerical solution $u^{(n)}(x, t_{i+1})$, using $u^n(x, t_i)$ as the initial approximation to $\bar{u}$, is given in the algorithm below.

```
Algorithm C1: Iterative coefficient - temporal integration.
Initial approximation:- $u^{(0)}(x, t_{i+1}) := u^{(n)}(x, t_i)$;
$k := 0$;
Iterate
        $k := k + 1$;
        Compute $K(\bar{u})$ such that $\bar{u} := u^{(k-1)}(x, t_{i+1})$;
        Find $u^{(k)}(x, t_{i+1})$ such that
        $\frac{u^{(k)}(x,t_{i+1}) - u^{(n)}(x,t_i)}{\delta t} + g(x, t_{i+1}) = K(\bar{u})\frac{\partial^2 u^{(k)}(x,t_{i+1})}{\partial x^2} \quad \in (a,b)$
Until $||u^{(k)}(x, t_{i+1}) - \bar{u}|| < \epsilon$
$n := k$
```

Numerical solutions of Equation (23) at each time step requires to execute Algorithm C1 within an outer iteration loop, with $t_i = i\delta t$, $i = 0, 1, \cdots$,, where δt is the step length of the temporal integration. Restrictions on δt are inevitable and are due to stability criteria of using an explicit scheme or the truncation errors of using an implicit scheme in approximating the temporal derivative in Equation (24). It should be noted that there are many different ways of obtaining a linearization. The quasi-linear problem defined in Equation (24) is only one way to linearize the problem in Equation (23).

Now consider the quasi-linear form as shown in Equation (24) now being defined in the time interval $(T_j, T_{j+1}]$, i.e.

$$\frac{\partial u}{\partial t} + g(x,t) = K(\bar{u})\frac{\partial^2 u}{\partial x^2} \in (a,b) \times (T_j, T_{j+1}] \tag{25}$$

Here $T_j = j\delta T$, $j = 0, 1, \cdots$,. A Laplace transform may be applied to Equation (25) [8], in such a way that many intermediate time steps can be removed which leads to the choice $\delta T > \delta t$. Therefore the time step counter used here is denoted as j which is different from i, i.e. $\delta T \neq \delta t$.

$$\mathcal{L}\left(\frac{\partial u}{\partial t}\right) + \mathcal{L}(g(x,t)) = \mathcal{L}\left(K(\bar{u})\frac{\partial^2 u}{\partial x^2}\right) \in (a,b) \tag{26}$$

Let

$$\mathcal{L}\left(u\right) \equiv \int_0^\infty e^{-\lambda\tau}u(x,\tau)d\tau = U(\lambda;x) \tag{27}$$

be the Laplace transform of the function $u(x,t)$. Equation (26) becomes

$$\lambda U - u(x,T_j) + G = K(\bar{u})\frac{d^2U}{dx^2} \in (a,b)\ , \tag{28}$$

subject to suitably Laplace transformed boundary conditions at $x = a$ and $x = b$. Here $G = \mathcal{L}(g)$ and

$$\lambda \in \left\{\lambda_p, p = 1, 2, \cdots, m : \lambda_p = \frac{p\ln 2}{T_{j+1} - T_j}\right\}\ , \tag{29}$$

where m is required to be chosen as an even number [10][11]. Therefore the quasi-linear problem in Equation (25) is converted to a set of independent parametric boundary value problems each as described by Equation (28), and these problems may be distributed and solved independently in a distributed environment which consists of a number of processors linked by a network. From experience, the value of m is usually a small even number not larger than 10 [10][11]. Numerical experiments shown later also confirm such experience. The inverse Laplace method described before is used to retrieve the solution.

```
Algorithm C2: Iterative coefficient - Laplace transformation method.
        Initial approximation:- $u^{(0)}(x,T_{j+1}) = u^{(n)}(x,T_j)$; $k := 0$;
        Iterate
                $k := k + 1$;
                Compute $K(\bar{u})$ such that $\bar{u} := u^{(k-1)}(x,T_{j+1})$;
                Distribute $p := 1$ to $m(j+1)$
                    Find $U(\lambda_p;x)$ from (28), i.e.
                    $\lambda_p U(\lambda_p;x) - u(x,T_j) + G = K(\bar{u})\frac{d^2U(\lambda_p;x)}{dx^2} \in (a,b)$;
                End Distribute
                Compute $u^{(k)}(x,T_{j+1})$ using inverse Laplace in Equation (21);
        Until $||u^{(k)}(x,T_{j+1}) - \bar{u}|| < \epsilon$ ;
        $n := k$;
```

Here $m(j)$ is the number of parametric equations. It is required to solve $m(j)$ parametric systems each described by Equation (28) resulting to $U(\lambda_p;x)$. In order to solve for $u(x,T)$ as described by Equation (24), Algorithm C2 needs to be iterated through $T_j = T_1, T_2, \cdots, T$, with suitable choices of $m(j)$, in the form of an outer iteration over Algorithm C2. One can, in other words, say that this method is a time-marching scheme with a coarser temporal step length ΔT. In actual implementation often different values of $m(j)$ are not necessary, and the results shown in this paper use the same number of parametric equations, denoted as $\bar{m}$, for all values of j.

Tests were carried out for $g(x,t) = 2x^4e^{-t}$, $K(u) = u^2$, boundary conditions $u(0,t) = 0$ and $u(1,t) = e^{-t}$, initial condition $u(x,0) = x^2$, and $T = 1$. The analytic solution of the model problem defined in Equation (23) is given by $u(x,t) = x^2e^{-t}$.

Algorithm C1 is used to provide a reference solution for comparison. A backward finite difference for the temporal derivative and a second order finite difference method are used in Equation (24). The spatial mesh size, $h = 1/2^7$, and the step size, $\delta t = 0.001$, are chosen in the numerical tests. The resulting linear system at each time step using the above discretisation is a tri-diagonal system, and the work required to solve such system is defined as one work unit. The reference solution at $t = T$, denoted as u_{C1}, requires 2171 work units and $\|u(x,T) - u_{C1}\|_2 = 0.000006$.

Algorithm C2 is used to solve the Laplace transformed set of equations and to retrieve an approximate solution, denoted as u_{C2}, to $u(x,T)$. A second order finite difference method, similar to the one used in Algorithm C1, is applied to discretize Equation (28), and this resulted to a set of linear tri-diagonal equations in which elements of the set can be solved in a distributed environment. The spatial mesh size is also chosen as $h = 1/2^7$. The total sequential work unit is recorded and is divided by $\bar{m}$ in order to obtain the distributed work unit, excluding the overhead of computing the inverse of the Laplace transformed solution, in the distributed computing environment.

Table 1 shows the error $\|u(x,T) - u_{C2}\|_2$ against various $\bar{m}$. Each column represents the errors using a particular value of and the last column shows the errors when $\Delta T = \delta t$. There is an optimal accuracy for each column and it seems to occur at $\bar{m} = 4$ and $\bar{m} = 6$. By choosing $\Delta T = 5\delta t$ the accuracy is not degraded very much as compared to the reference solution obtained by using Algorithm C1. On the other hand the accuracy is only one decimal place less than that obtained by using Algorithm C1 when $\Delta T = 25\delta t$. For larger values of ΔT, say $\Delta T = 125\delta t$, the accuracy increases with larger values of $\bar{m}$, which is a sensible observation for nonlinear problems, because a larger step requires larger number of parametric equations to compensate the lost of accuracy due to truncation errors. These results show that the inverse Laplace method based on Stehfast is accurate enough by taking the choice of $\Delta T \leq 25\delta t$. The results also suggest that the value of $\bar{m}$ should not be chosen unnecessarily large.

Table 2 shows the distributed work unit, excluding the work required for the inverse Laplace computation, against various $\bar{m}$. Again each column represents a given value of ΔT. Note that the distributed work units correspond to the optimal accuracy as shown in Table 1 appear in italic fonts. These work units are significantly less than that required in obtaining the reference solution. Note also that the distributed work units in the last column, i.e. when $\Delta T = \delta t$, are similar to the sequential work unit as required by using Algorithm C1. This serves to validate the correctness of Algorithm C2. At $t = T$ the errors with respect to the analytic solution, $\|u(x,T) - u_{C1}\|_2$ and $\|u(x,T) - u_{C2}\|_2$, and the discrepancy between the reference solution and the inverse Laplace solution, $\|u_{C1} - u_{C2}\|_2$, are presented in Figure 5.1. The inverse Laplace solution is correct up to 4 decimal places.

Table 1: $\|u(x,T) - u_{C2}\|_2$ for various $\bar{m}$ and ΔT.

	ΔT			
$\bar{m}$	$125\delta t$	$25\delta t$	$5\delta t$	δt
2	0.016913	0.017783	0.018089	0.018159
4	0.001039	*0.000048*	0.000348	0.000417
6	0.000504	0.000064	*0.000009*	0.000021
8	0.000497	0.000094	0.000020	*0.000005*
10	0.000496	0.000094	0.000019	0.000004
12	0.000495	0.000094	0.000019	0.000004

Table 2: Distributed work unit for various $\bar{m}$ and ΔT.

	ΔT			
$\bar{m}$	$125\delta t$	$25\delta t$	$5\delta t$	δt
2	73	200	680	3000
4	47	*152*	600	2082
6	46	151	*600*	2090
8	46	150	600	*2091*
10	46	150	600	2091
12	46	150	600	2091

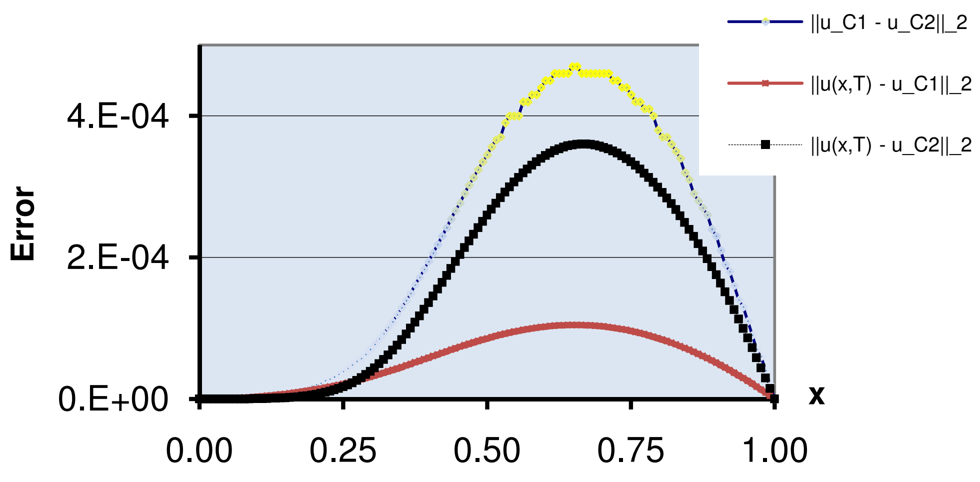

Figure 1: Discrepancies of solutions - the nonlinear parabolic problem.

5.2 An European Option Example

Recall Equation (22)

$$\frac{\partial u}{\partial t} = \alpha(u,x)\frac{\partial^2 u}{\partial x^2} + \beta(x)\frac{\partial u}{\partial x} - \gamma u$$

where $u(x,t)$ denote the value of an option, x is the current value of the underlying asset and t is the time. The value of the option relates to the current value of the underlying asset via two stochastic parameters, namely the volatility σ and the interest rate γ, embedded in the leadin coefficients $\alpha(u,x) = \frac{1}{2}\sigma(u)^2x^2$ and $\beta(x) = \gamma x$. The above equation subject to the initial and boundary conditions,

$$u(x,0) = \max\{K - x, 0\} \tag{30}$$

$$u(0,t) = Ke^{-rt}, \;\; u(L,t) = 0 \tag{31}$$

is the famous forward Black-Scholes model for an European option defined in $\Omega^+ \times (0,T]$. Here $\Omega^+ = \{x : 0 \le x \le L\}$; K is the strike price which means that the holder of the option may execute at a prescribed asset, usually known as the underlying asset, for the prescribe amount K at expiry T; L is usually chosen as a large value covering as much values of x as possible. Details of derivation of the Black-Scholes equation is not discussed here. Readers who are interested in the derivation may find numerous references, such as [12], [13] and [14], just to name a few, in the area of financial mathematics.

Very often, over a short period of time the interest rate, γ, is fixed while the volatility, $\sigma(u)$, is varying and may be a function of the transaction costs [15], the second derivative of the option value [16], or, in some cases, the solution of a nonlinear initial value problem [15]. In order to illustrate the nonlinear solver in combination with the Laplace transformation method, the nonlinear volatility [17]

$$\sigma(u) = \sigma_0\sqrt{1 + a(u)} \tag{32}$$

is used in the following numerical tests, where a is the proportional transaction cost scaled by σ_0 and the transaction time. Here the author adopted a heuristic approach in which the transaction cost is related to the option value and follows a Gaussian distribution. In order to demonstrate the inverse Laplace transform technique for nonlinear problems, a sine function is used to produce the effect of a pulse-like distribution instead of a Gaussian distribution. Therefore the proportional transaction cost, a, may be replaced by a function of the option value such as

$$a(u) = \sin\left(\frac{u\pi}{K}\right) \tag{33}$$

where K is the strike price. The above $a(u)$ is used in the subsequent numerical tests.

The coefficient α of the option pricing problem, defined in the time interval (t_i, t_{i+1}), is computed by using an approximation $\bar{u}$, which is updated in every step of an iterative update process within the given time interval. Each step of the iterative update process involves obtaining a numerical solution to the quasi-linear problem

$$\frac{\partial u}{\partial t} = \alpha(\bar{u}, x)\frac{\partial^2 u}{\partial x^2} + \beta(x)\frac{\partial u}{\partial x} - \gamma u \in \Omega^+ \times (t_i, t_{i+1}] \tag{34}$$

Here $t_i = i\delta t$, $i = 0, 1, \ldots$, and δt is the time step length of a temporal integration method. Let $u^{(n)}(x, t_{i+1})$ and $u^{(n)}(x, t_i)$ be the numerical solutions of Equation (23) at $t = t_{i+1}$ and $t = t_i$ respectively. The difference between the iterative update processes for the nonlinear parabolic problem and the option pricing problem in a temporal integration method is listed below:

1. Compute $\alpha(\bar{u})$ such that $\bar{u} := u^{(k-1)}(x, t_{i+1})$;
2. Find $u^{(k)}(x, t_{i+1})$ such that $\frac{u^{(k)}(x,t_{i+1}) - u^{(n)}(x,t_i)}{\delta t} = \alpha(\bar{u})\frac{\partial^2 u^{(k)}(x,t_{i+1})}{\partial x^2} + \beta\frac{\partial u^{(k)}(x,t_{i+1})}{\partial S} - \gamma u^{(k)}(x, t_{i+1}) \in \Omega^+$

The iterative coefficient technique with a temporal marching scheme, is used to provide a reference solution which is used for comparison.

Linearization methods are needed for nonlinear problems even in the case of using Laplace transformation methods. Apart from using iterative coefficient for nonlinear problems, Netwon's linearisation is also examined in this section. The former one is the same as the algorithms described in Section 5.1.

Iterative coefficient. Take Laplace transform of Equation (34), now being defined in the larger time interval $\tau \in (T_j, T_{j+1}]$, and the transformed equation in its differential form is given below.

$$\alpha(\bar{u})\frac{d^2 U}{dx^2} + \beta(x)\frac{dU}{dx} - (\gamma + \lambda)U = -u(x, T_j) \tag{35}$$

where $U = \mathcal{L}(u)$. An update process is used to renew the value of $\bar{u}$ and an inverse Laplace transform is used to obtain the numerical solution $u^{(n)}(x, T_{j+1})$, using $u^{(n)}(x, T_j)$ as the initial approximation to $\bar{u}$. The update process is described in the algorithm below.

```
Algorithm C2: Iterative coefficient - Laplace transformation method.
Initial approximation:- $u^{(0)}(x, T_{j+1}) := u^{(n)}(x, T_j)$;
$k := 0$;
Iterate
    $k := k + 1$;
    Compute $\alpha(\bar{u})$ such that $\bar{u} := u^{(k-1)}(x, T_{j+1})$;
    Distribute $p := 1$ to $m(j + 1)$
```

```
            Find $U(\lambda_p; x)$ such that
                $\alpha(\bar{u})\frac{d^2U(\lambda_p;x)}{dx^2} + \beta(x)\frac{dU(\lambda_p;x)}{dx} - (\gamma + \lambda_p)U(\lambda_p; x)$
                $= -u(x, T_j)$
        End Distribute
        Compute $u^{(k)}(x, T_{j+1})$ using the inverse Laplace transform in Equation (21);
    Until $||u^{(k)}(x, T_{j+1}) - u^{(k-1)}(x, T_{j+1})|| < \epsilon$
    $n := k$
```

Here $m(j + 1)$ is the number of transformation parameters and $T_j = j\Delta T$. In order to solve Equation (35) for $U(\lambda_p; x)$, one can employ the same finite difference technique described above. In order to solve Equation (22) for $u(x, T)$, Algorithm C2 needs to be iterated through $T_j := T_1, T_2, ..., T$ by using suitable values of $m(j)$ in the form of an outer iteration. Similar to the case in the nonlinear parabolic problem, the actual implementation does not require different values of $m(j)$ for many problems, and the numerical tests shown here used the same number of transformation parameters, denoted as $\bar{m}$, for different values of during the outer iteration loop. Note that in this case ΔT can be chosen to be much greater than δt because the fine details of at each time step of a temporal integration is not required in the inverse Laplace transformation calculation. A full account on different choices of ΔT is not included in this work. In the numerical tests shown in this section $\Delta T = \frac{T}{10}, \frac{T}{20}, \frac{T}{40}, \frac{T}{80}$, all normalised with respect to one year, and $t_0 = 0$. In the numerical tests shown in subsection $\Delta T = \frac{T}{10}, \frac{T}{20}, \frac{T}{40}, \frac{T}{80}$, all normalised with respect to one year, and $t_0 = 0$.

Newton's linearisation. This is a common linearisation method. Consider a small perturbation δu applied to Equation (22), defined in the time interval $t \in (T_j, T_{j+1}]$, leads to

$$\left\{\frac{\partial}{\partial t} - \left(\alpha'(u)\frac{\partial^2 u}{\partial x^2} + \alpha(u)\frac{\partial^2}{\partial x^2} + \beta(x)\frac{\partial}{\partial x} - \gamma\right)\right\}\delta u$$
$$= -\left\{\frac{\partial u}{\partial t} - \left(\alpha(u)\frac{\partial^2 u}{\partial x^2} + \beta(x)\frac{\partial u}{\partial x} - \gamma u\right)\right\} \in \Omega^+ \times (T_j, T_{j+1}] \qquad (36)$$

where δu is a small incremental change of u. The initial and boundary conditions can be obtained by taking a small perturbation to the initial condition defined at $t = T_j$ and the boundary condition defined in Equation (31) leads to the initial condition:

$$\delta u(x, T_{j+1}) = 0 \qquad (37)$$

and the boundary conditions:

$$\delta u(0, t) = 0, \;\; \delta u(L, t) = 0 \qquad (38)$$

A Laplace transform can be applied to Equation (36), now being defined in the time interval $t \in (T_j, T_{j+1}]$. The transformed equation in its differential form becomes

$$\mathcal{L}(\delta u) - \delta u(S, T_j) - \left(\alpha'(u)\frac{d^2 u}{dx^2} + \alpha(u)\frac{d^2}{dx^2} + \beta(x)\frac{d}{dx} - \gamma\right)\mathcal{L}(\delta u)$$

$$= -\mathcal{L}(u) - u(x,T_j) - \left(\alpha(u)\frac{d^2u}{dx^2} + \beta(x)\frac{du}{dx} - \gamma u\right) \tag{39}$$

Using $u^{(n)}(x,T_j)$ as the initial approximation to $u^{(0)}(x,T_{j+1})$ to start the inner iterative loop in order to obtain the numerical solution $\mathcal{L}(\delta u^{(n)}(x,T_{j+1}))$. The inner iterative loop is described in the algorithm below.

```
Algorithm NM2: Newton's Method - Laplace transformation method.
Initial approximation:- $u^{(0)}(x,T_{j+1}) := u^{(n)}(x,T_j)$;
$k := 0$;
Iterate
        $k := k+1$;
        Compute { $\alpha(\bar{u})$, $\alpha'(\bar{u})$, $\alpha'(\bar{u})\frac{d^2\bar{u}}{dx^2}$,
            $-\mathcal{L}(\bar{u}) - u(x,T_j) - \left(\alpha(\bar{u})\frac{d^2\bar{u}}{dx^2} + \beta(x)\frac{d\bar{u}}{dx} - \gamma\bar{u}\right)$
        } such that $\bar{u} := u^{(k-1)}(x,T_{j+1})$;
        Distribute $p := 1$ to $m(j+1)$
            Find $\mathcal{L}(\delta u^{(k)}(x,T_{j+1}))$ such that
            $\mathcal{L}(\delta u^{(k)}(x,T_{j+1})) - \delta u(x,T_j)$
            $- \left(\alpha'(\bar{u})\frac{d^2u}{dx^2} + \alpha(\bar{u})\frac{d^2}{dx^2} + \beta(x)\frac{d}{dx} - \gamma\right)\mathcal{L}(\delta u^{(k)}(x,T_{j+1}))$
            $= -\mathcal{L}(\bar{u}) - u(x,T_j) - \left(\alpha(\bar{u})\frac{d^2\bar{u}}{dx^2} + \beta(x)\frac{d\bar{u}}{dx} - \gamma\bar{u}\right)$
        End Distribute
        Compute $\delta u^{(k)}(x,T_{j+1})$ using the inverse Laplace transform in Equation (21);
        $u^{(k)}(x,T_{j+1}) := \bar{u} + \delta u^{(k)}(x,T_{j+1})$;
Until $||\delta u^{(k)}(x,T_{j+1})|| < \epsilon$
$n := k$
```

Here $m(j)$ is the number of transformation parameters. Similar to the spatial discretisation method described above the same finite difference technique is employed here. The Algorithm NM2 can now be iterated through $T_i := T_1, T_2, ..., T$ by choosing suitable values of $m(i)$ in the form of an outer iteration. The other discussion on the choice of $m(j)$ and ΔT also applies.

The same mathematical model of European put option described above is used to demonstrate the efficiency of the two linearization methods. In particular the computational complexity is to be examined when the algorithms are implemented on distributed computers. Here the volatility is chosen as a nonlinear function as described above and the parameters σ_0 and γ are chosen to be 0.4 and 0.5 respectively. A second order finite difference method is applied to each parametric equation as given by Equation (39). The spatial mesh size is chosen to be $\frac{320}{2^9}$.

Algorithm C1 is used to obtain a reference solution for comparison. It is being run sequentially on a COMPAQ laptop using a F90 program implementing a backward finite difference method along the temporal axis with $\delta t = 1/365$, i.e. 1 day, leading

Table 3: Nonlinear volatility - Computational work unit (Note: The sequential work unit using Algorithm C1 is 246).

$\bar{m}$	Algorithm C2				Algorithm NM2			
	ΔT				ΔT			
	1/40	1/80	1/160	1/320	1/40	1/80	1/160	1/320
4	58	103	177	326	43	83	134	249
6	58	103	177	326	43	70	126	245
8	58	103	177	326	43	71	126	220
10	58	103	177	326	43	71	126	214
12	180	123	186	327	43	71	126	213

to an implicit scheme in which a second order finite difference methods along the spatial axis x can be implemented. The stopping criterion used in the linearization step is chosen as $\epsilon = 10^{-5}$. The numerical solution $u(x,T)$ obtained by this temporal integration is denoted as u_{C1}.

Each of the two linearisation methods, Algorithms C2 and NM2, leads to a set of parametric equations in the Laplace space. The set of equations is solved, sequentially in the same computational environment, by taking different values of m. An approximation to $u(x,T)$ corresponds to each value of m is found by using the inverse Laplace transform as given by Equation (21). The approximations obtained by means of the iterative coefficient and Newton's methods are denoted as u_{C2} and u_{NM2} respectively. By using the choices of $\Delta T = \frac{T}{10}, \frac{T}{20}, \frac{T}{40}, \frac{T}{80}$, the number of outer iterations are 10, 20, 40, and 80 respectively for both Algorithms C2 and NM2.

As the problems set up are the same in the linear and nonlinear cases of volatilities, the computational complexity of solving a set of tri-diagonal system of equations with a fixed spatial mesh size is the same as that in one step of a temporal scheme, say η. For simplicity it is possible to call the computational complexity η as one work unit. For the purpose of comparison the work unit required to obtain u_{C1} is 246. Table 3 shows the distributed work unit when the solutions of the set of parametric equations are done independently. The work unit involved in Algorithm NM2 is less than that of Algorithm C2 in general. On the other hand there is no sudden increase of work when $m = 12$. The total sequential work unit may be obtained by multiplying the distributed work unit to m. The total parallel complexity is simply the complexity for solving the set of equations plus inverse Laplace transform and communication overheads, i.e.

$$W_m \eta + \mu + c$$

Here W_m denotes the work unit when there are m computers and m parametric equations, μ is the overhead of computing the inverse Laplace transform and c denotes communication. In general the dominant contribution to the total work is the solutions process of the set of parametric equations and contribution from c is almost negligible.

For comparison purposes the reference solution u_{C1} is used in comparing results using the Laplace transformation method. Discrepancies in solutions, i.e. $\|u_{C1} -$

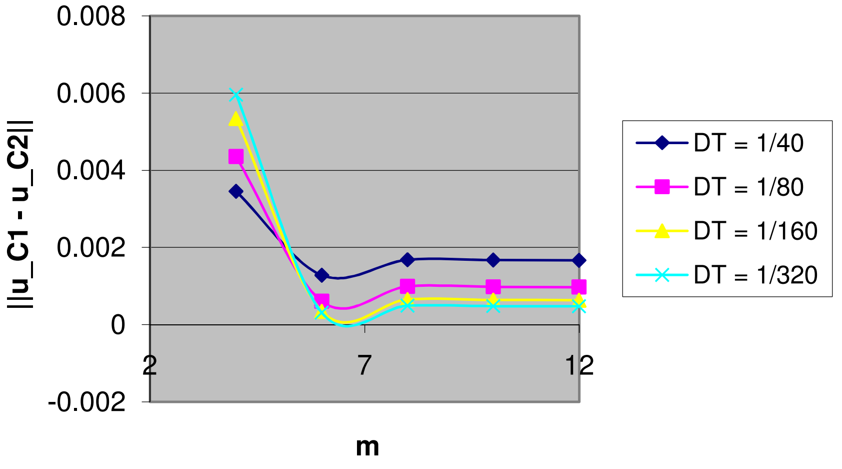

Figure 2: Discrepancies of solutions: $\|u_{C1} - u_{C2}\|_2$ - the nonlinear European option problem.

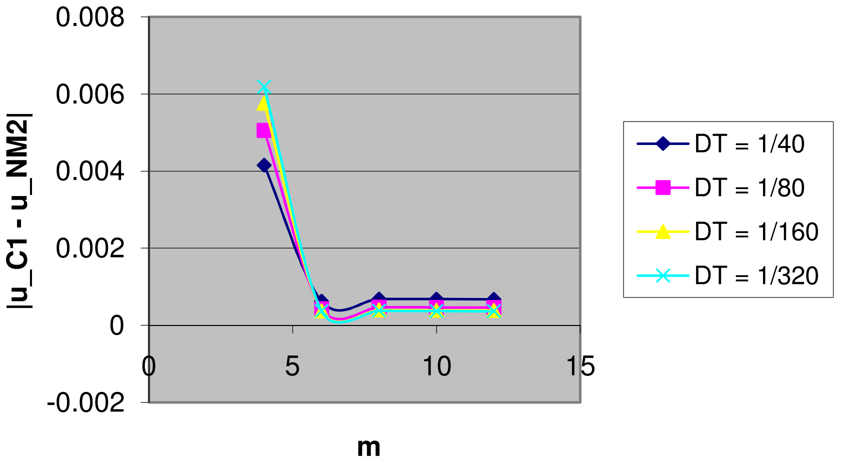

Figure 3: Discrepancies of solutions: $\|u_{C1} - u_{NM2}\|_2$ - the nonlinear European option problem.

$u_{C2}\|_2$ and $\|u_{C1} - u_{NM2}\|_2$ using various ΔT, are recorded in Figures 2 and 3. In general the discrepancy levels off when $m = 8$, which suggests that the use of more terms in the inverse Laplace transform at a fixed value of ΔT has no effect on the accuracy. On the other hand smaller ΔT produces small discrepancy at the expense of requiring more work unit as recorded in Table 3 for comparisons.

The other feature of the two linearization methods is that the use of the iterative coefficient method has no advantage over Newton's method. The pointwise discrepancies of the solutions, $|u_{C1}(x,T) - u_{C2}(x,T)|$ and $|u_{C1}(x,T) - u_{NM2}(x,T)|$, as compared with a time-stepping method applied to the nonlinear problem with $\bar{m} = 8$ and $\Delta\tau = 100$ for the case $h = 320/2^9$ is recorded in Figure 4. Such comparison is not the best way of comparing results, but it gives an idea of the deviation from the numerical solution obtained by an implicit temporal integration based on back-

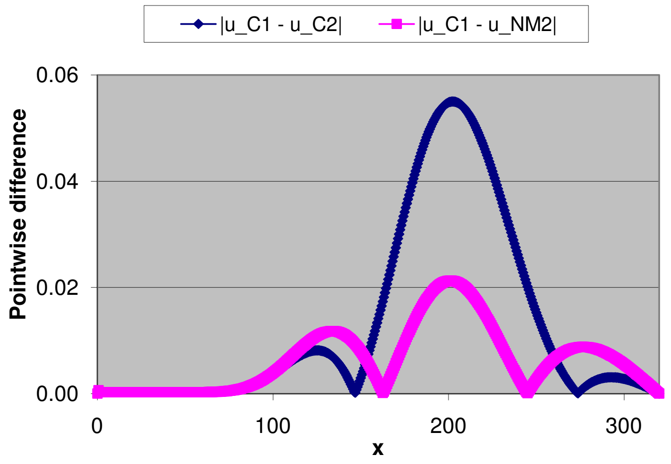

Figure 4: Pointwise comparison of numerical solutions for the nonlinear European option problem.

ward difference. It shows that the largest discrepancies occur nearby the strike price. Therefore it is the resolution problem of the spatial finite difference scheme near the strike price. On the other hand the order of discrepancies is of the order of 10^{-3} which is in consistent with the results for linear problems. Finally the numerical approximations to $u(x,T)$ using various methods and the initial condition $u(x,0)$ are plotted in Figure 5.

Note that the linearization techniques described above may be extended to handle American Options and Asian Options with little difficulties. A complimentary problem can always be formed for American options, in which the free boundary is embedded into the formulation. A Laplace transform can then be applied to the complimentary problem [13] followed with an inner iterative loop to drive the inequality to convergence.

5.3 Testing a Two-Level Time-Domain Algorithm

A careful examination of Algorithms C2 and NM2, tested in Sections 5.1 and 5.2, revealed that a time-stepping is re-introduced to handle the nonlinearity. The time step length re-introduced is of a larger size compared with those used in a temporal integration scheme. These algorithms is in essence a temporal integration technique with a larger temporal step ΔT in which parallelism is introduced at each individual temporal step via transformation methods described earlier. The main reason for using such scheme is to ensure the nonlinearity in the differential equation can be dealt with adequately. The result of this scheme leads to numerical solutions on a coarse temporal mesh. The accuracy of the numerical results obviously depends on the accuracy of the inverse Laplace transform method used and is out of the scope of this Chapter. The objective of this section is to examine the possibility of using two different temporal meshes in order to induce parallel properties into time-domain computation. The ex-

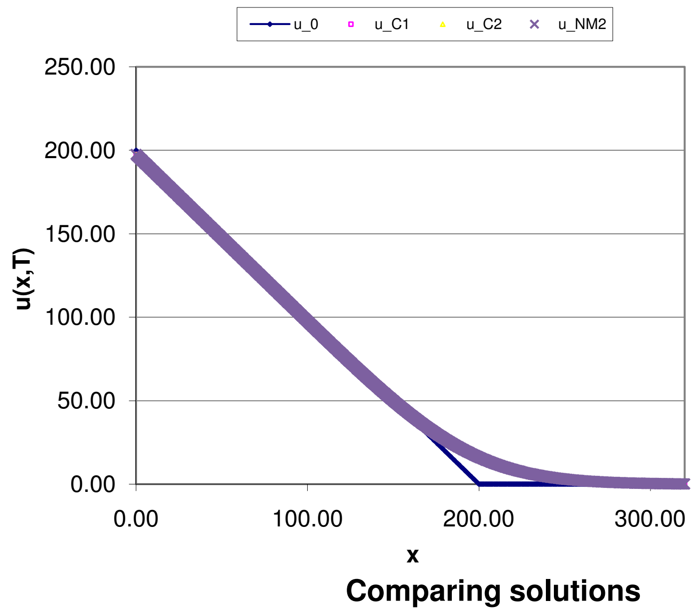

Figure 5: Numerical solution at $t = T$ of the nonlinear European option problem.

ample nonlinear parabolic problem used in Section 5.1 is used to provide numerical tests.

Let the time domain $(0, T]$ be divided into n temporal steps, i.e. $\delta t = T/n$. The result of this leads to a fine time level where temporal integration may take place. This is essentially applying Algorithm C1 to Equation (24), in the case of the nonlinear parabolic problem, and an outer iterative loop to run through each time interval $(i\delta t, (i+1)\delta t]$, $i = 0, 1, \cdots, n-1$. Note that it is no reason why the temporal step length needs to be restricted to a constant value and that the algorithm may be easily generalised to variable temporal step length.

Suppose now the time domain $(0, T]$ is equally divided into N parts, where $N < n$, each represents a larger time interval, i.e. $\left(j\frac{n}{N}\delta t, (j+1)\frac{n}{N}\delta t\right]$, $j = 0, 1, \cdots, N-1$. This corresponds to a coarser time mesh with temporal step length $\Delta T = T/N$. Solutions obtained at each $j\Delta T$, where $j = 0, 1, \cdots, N$, can be considered as initial conditions for each of the time intervals $\left(j\frac{n}{N}\delta t, (j+1)\frac{n}{N}\delta t\right]$, $j = 0, 1, \cdots, N-1$, where a temporal integration method can be applied in each of these intervals independent from the others. Thus the fine details can be obtained using a combination of two different temporal meshes.

Clearly it is not sensible to employ the same temporal integration method to obtain solutions at the coarse time mesh for two reasons. First the accuracy of these solutions is not of the same order of those obtained by using a finer time step. Second parallelism cannot be introduced into the temporal integration method. However using the Laplace transformation method described in Section 5.1 the solution of the nonlinear problem at each time interval of the coarse temporal mesh may be obtained by means of a parallel/distributed computer. Sequential applications of such parallel/distributed algorithm allows stepping through all of the time steps in the coarse level problem. Finally it should be noted also that the solution accuracy at time steps $j\Delta T, T = 1, \cdots, N$ are not degraded and can be used as the initial values for the fine level problem. The two-level time domain parallel algorithm is given as below.

```
Algorithm: Two-level time domain parallel algorithm
    do $j := 0$ to $N - 1$
        Coarse level ($\Delta T$)- Apply Algorithm C2 or NM2:-
        Apply Laplace Transformation Method for $t \in (T_j$ to $T_{j+1}]$
    enddo
    Distribute $j := 0$ to $N - 1$
        Fine level ($\delta t = \frac{N}{n}\Delta T$) - Apply Algorithm C1:-
        Apply temporal integration for $t \in (T_j$ to $T_{j+1}]$.
    end Distribute
```

In the present experiment, the coarse temporal mesh is chosen as $\Delta T = 0.0005$ and the fine temporal mesh is chosen as $\delta t = 0.0001$. The same nonlinear parabolic problem in Section 5.1 is tested here. The total work unit recorded by using the implicit time marching algorithm is 2171, as in Section 5.1. The parallel work unit recorded by using the coarse temporal mesh is 600 assuming there are 5 computers available. The parallel work unit recorded by using the fine temporal mesh is $\Delta T/deltat = 5$. Hence the total parallel work unit is 605. Let $u_{C1}(x,t)$ denote the fine level solution obtained by means of a temporal marching scheme and $u_{C1-2level}(x,t)$ denote the fine level solution obtained by means of the above 2-level temporal method. At the time $i\delta t$, $i = 1, 2, \cdots, n$ these norms are computed: $\|u(x,i\delta t) - u_{C1}(x,i\delta t)\|$, $\|u(x,i\delta t) - u_{C1-2level}(x,i\delta t)\|$, and $\|u_{C1}(x,i\delta t) - u_{C1-2level}(x,i\delta t)\|$. Figure 6 displays these norms in logarithmic scale. Note that the first two norms behave similarly showing that the present 2-level time domain method possess similar accuracy as the implicit temporal scheme. However the advantage of the total parallel work unit can easily be seen.

6 Conclusion

This Chapter presents the idea of inducing parallel/distributed computational properties into an otherwise intrinsic sequential algorithm via the use of suitable mathematical tools. In the present case the tool is chosen to be those available and adapted frequently in the construction of analytic solutions and the understanding of solution behaviour - transformation methods. Two different class of transformation methods, i.e. similarity transform and Laplace transform, are investigated. Linear problems are used to provide a description of the methodologies which are then extended to nonlinear problems. Two types of problems are examined in this Chapter, they are a nonlinear parabolic problem and an European option pricing problem. Computational work

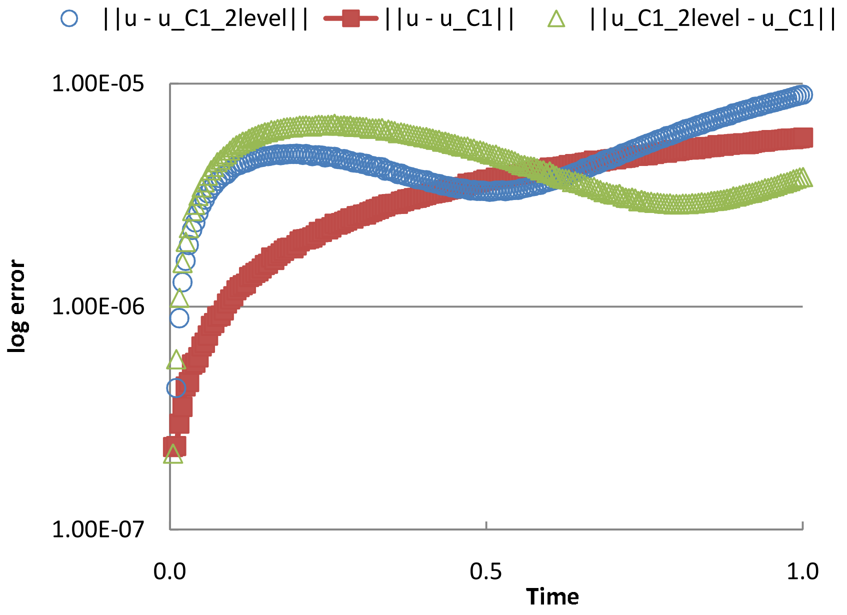

Figure 6: Error of the nonlinear parabolic problem.

and complexity are investigated showing the advantage of inducing parallel properties into the given continuous differential problems. A two-level time domain parallel algorithm is used to provide fine details of the solution at very fine temporal steps. The use of two-level time domain has the advantage of combining temporal integration methods and transformation methods in deriving an accurate parallel/distributed algorithm for nonlinear problems.

References

[1] Miranker, W.L., Liniger, W.: Parallel methods for the numerical integration of ordinary differential equations. *Math. Comp.*, **91**, (1967) 303–320

[2] Lions, J.-L., Maday, Y., Turinici, G.: A parareal in time discretisation of PDE's. *C.R. Acad. Sci. Paris, Serie I*, **332**, (2001) 661–668

[3] Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, **29**, (2007) 558–578

[4] Gander, M.J., Hairer, E.: Nonlinear convergence analysis for the parareal algorithm. **Lecture Notes in Computational Science**, **60**, (2008) 45–56

[5] Hockney, R.W., Jesshope, C.R.: **Parallel Computers 2: architecture, programming and algorithms**. CRC Press, 1988

[6] Chan Man Fong, C.F., De Kee, D., Kaloni, P.N.: **Advanced Mathematics for Applied and Pure Sciences**. Gordon and Breach Science Publishers, 1997

[7] Moridis, G.J., Reddell, D.L.: The Laplace transform finite difference method for simulation of flow through porous media. *Water Resources Research*, **27**, (1991) 1873–1884

[8] Lai, C.-H., Parrott, A. K., Rout, S., Honnor, M.E.: A distributed algorithm for European options with nonlinear volatility. *Computers and Mathematics with Applications*, **49**, (2005) 885–894

[9] Stehfast, H.: Numerical inversion of Laplace transforms. *Comm ACM*, **13**, (1970) 47–49

[10] Crann, D.: The Laplace transform - Numerical inversion for computational methods. Technical Report No. 21, University of Hertfordshire, July (1996).

[11] Crann, D., Davies, A.J., Lai, C.-H. and Leong, S.W.: Time domain decomposition for European options in financial modelling. In **Proceedings of the 10th International Conference on Domain Decomposition Methods**, August 1997, Colorado. J. Mandel, C. Farhat, and X.-C. Cai, editors, **Domain Decomposition Methods 10**, American Mathematical Society, (1998).

[12] Wilmott, S., Howison, S., Dewynne, J.: **The Mathematics of Financial Derivatives: A Student Introduction**. Cambridge University Press, 1995

[13] Wilmott, P.: Derivatives - The Theory and Practice of Financial Engineering. Wiley, 1998

[14] Stampfli, J., Goodman, V.: The Mathematics of Finance: Modelling and Hedging. Brooks/Cole Thomson Learning, 2001

[15] Barles, G., Soner, H.M.: Option pricing with transaction costs and a nonlinear black-Scholes equation, *Finance Stochast*, **2**, (1998) 369–397

[16] Parás, A., Avellaneda, M.: Dynamic hedging portfolios for derivative securities in the presence of large transaction costs, *Appl. Math. Finance*, **1**, (1994) 165–193

[17] Boyle, P., Vorst, T.: Option replication in discrete time with transaction costs, *Journal of Finance*, **47**, (1973) 271–293