NUMERICAL SOLUTION OF DISTRIBUTED OPTIMAL CONTROL PROBLEM OF WAVE EQUATION USING PARALLEL ALGORITHM

SEYED MEHDI MOUSA VI
Young Researchers and Elite Club, Baft Branch, Islamic Azad University, Baft, Iran

ZAINAB MORADIAN
Young Researchers and Elite Club, Baft Branch, Islamic Azad University, Baft, Iran

MOHSEN ASADI ZANGENEH*
*CORRESPONDING AUTHOR
Young Researchers and Elite Club, Baft Branch, Islamic Azad University, Baft, Iran
Email: m.asadi0129@yahoo.com; Tel: +98-916-9880-129

REZA NOURMANDI-POUR
Department of computer, Sirjan Branch, Islamic Azad University, Sirjan, Iran

ABSTRACT
By arising of computers, human beings tried to achieve more processing power and nowadays processing systems have a serious role in different sciences such as: meteorology, medicine, nuclear, physics, and chemistry. In this paper, the solution of distributed optimal control and boundary control problems constrained with linear and semi-linear wave equation with simple shooting method with parallel algorithm to computing Jacobin matrix in Newton method has been presented. Moreover, an algorithm based on to find its numerical solution has been explained. In addition, exact controllability of boundary control of wave equation has been explained. Finally, the result showed that with decreasing $\Delta x$ the solution has low accuracy and for $\Delta x = 0.005$ the solution is unacceptable but for $s=0.5$ this problem is solved and the solution has high accuracy.

KEYWORDS: SHOOTING METHOD, PARTIAL DIFFERENTIAL EQUATION, LAGRANGE MULTIPLIER RULE

INTRODUCTION
When the industry wide switch too multicore and many core architecture parallel computing has become the only venue in sight for continued growth in Application performance. In order for the performance for the application to grow future generations of hardware, a significant portion of its computation must be done with scalable parallel algorithms. It's therefore important to develop and deploy as many scalable parallel algorithm as possible [2]. Therefore, why this is different from what the high-performance computing community has been working on in the past several decades? A simple answer is that the bulk of the prior work was along the lines of performing domain partitioning and use serial algorithms on each partition. While this approach has been very effective in allowing the science community to address increasingly larger problems using increasingly larger clusters, it has not made significant progress in parallelizing the algorithms used in each computing node [3]. A good example is that many Intel Math Kernel Libraries are still only in sequential form when asked to solve a single large problem. What we need to do this time around is to introduce parallel algorithms into each computing node. [4] The next question is what makes the parallel algorithms for multicore and many core processors challenging? As it turns out, the main challenges arise from the fact that the rate at which memory data can be accessed by the processor chips is much lower than the rate at which arithmetic operations can be performed on the chip.
A closer look into the standard organization of the memory system further reveals that data accesses need to be highly regular in order for the data to be delivered to the processor at a rate close to the advertised rate. Today, these deficiencies are compelling parallel algorithm designers to resort to highly sophisticated data locality and regularization techniques, thereby drastically complicating their algorithms [5]. Experts use different tools to achieve this goal, such as using VLSI technology, architecture techniques to increase processing speed such as vector processors and super scalar and taking advantage of several processors to execute a task. Cluster computing [6] which means using two or more computers together to achieve flexibility, scalability and more processing power is not a new field of calculation; however it is mostly being used when users frequently need parallel and distributed calculations [7]. The requisite to improve and upgrade the components of a parallel processing system in one hand and the speed of software components with high performance and availability one the other hand, makes these systems an effective tool which is greatly used in physics and Meteorology; in a manner that many algorithms in these fields are being executed in cluster environments to be solved with a higher speed. In following line we introduce some popular approach to Parallelism.

**I- CLUSTER COMPUTING SYSTEMS**

A cluster is a type of parallel or distributed processing system, which is consist of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource. Their taxonomy is based on how their processors, memory, and interconnect are laid out. The most common systems are [8]:

- MPP (Massively Parallel Processing)
- SMP (Symmetric Multi-processor)
- CC-NUMA (Cache-Coherent Non uniform Memory Access)
- Distributed systems
  - Cluster Systems

Based on distributed processing power in clusters, high availability and fault tolerance are important parameters to keep performance on a pleasant level. In these systems, failure of a node can lead to other processes and even the whole task’s failure. Here check pointing and process migration are the solutions of this problem.

**II- MESSAGE PASSING INTERFACE**

Programs need resources and due to architecture and physical limitations and the necessity of higher processing speed, parallel processing is massively needed. Some tools and models have been presented for parallel programming such as shared memory, message passing, oriented model, hybrid model, java [9]. In this criterion, message passing is the most efficient, widely used, programming paradigm on distributed memory systems. MPI is a useful library used in cluster systems; this library helps programmers to change their programs from serial to parallel form with a standard structure. In MPI the whole transmission is done by programmer and the compilers do nothing. Although there is other libraries such as Madeleine III and MPICH-G2 witch are compared in [10].

**III- WRF METEOROLOGY MODEL**

The development of the Weather Research and Forecasting (WRF) modeling system is a multi-agency effort intended to provide a next-generation mesoscale forecast model and data assimilation system that will advance both the understanding and prediction of mesoscale weather and accelerate the transfer of research advances into operations [11]. The model is being developed as a collaborative effort among the following organizations:

- NCAR Mesoscale and Micro scale Meteorology (MMM) Division [12]
- National Oceanic and Atmospheric Administration’s (NOAA) [13]
- National Centers for Environmental Prediction (NCEP)

This model is fully compressible, flexible and effective in parallel computing and can be used in scales of hundred meters to thousands of kilometers. It could be used in ideal experimental cases as well. Parameterization of atmospheric events, data assimilation, forecast research, real-time NWP, regional climate research, hurricane research, coupled-model applications and teaching are other aspects of WRF model. However, all method be useful, we want to solve distributed optimal control and boundary control problems constrained with linear and semi-linear wave equation with simple shooting method with parallel algorithm to computing Jacobin matrix in Newton method.

**1- METHODOLOGY**

The wave equation is an important second-order linear partial differential equation for the description of waves, as they occur in physics, such as sound waves, light waves and water waves. It arises in fields like acoustics, electromagnetic, and fluid dynamics. In its simplest form, the wave equation concerns a time variable $t$, one or more
spatial variables \( \{x_1, x_2, ..., x_n\} \) and a scalar function \( u = u(x_1, x_2, ..., x_n, t) \) whose values could model the height of a wave. The wave equation for \( u \) is:

\[
\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u
\]

(1-1)

where, \( \nabla^2 \) is the (spatial) Laplacian and where \( c \) is a fixed constant. In this article we will consider one-dimensional wave equation. We want to solve distributed optimal control and boundary control problems constrained with linear and semi-linear wave equation with simple shooting method. We know that the one-dimensional wave equation has the unique solution we need two initial and boundary Dirichlet and Neumann conditions have the following

\[
\begin{align*}
\text{Initial conditions:} & \quad u(x, 0) = w(x) \quad x \in (0, 1) \\
& \quad u_t(x, 0) = z(x) \quad x \in (0, 1)
\end{align*}
\]

\[
\begin{align*}
\text{Boundary conditions:} & \quad u(0, t) = 0 \quad t \in (0, T) \\
& \quad u(1, t) = 0 \quad t \in (0, T)
\end{align*}
\]

(1-2)

We define the cost functional as follows

\[
J(u, f) = \frac{\alpha}{2} \int_0^T \int_0^1 K(u)dxdt + \frac{\beta}{2} \int_0^1 \Phi_1(u(x, T))dx + \frac{\gamma}{2} \int_0^T \int_0^1 f^2 dxdt,
\]

(1-3)

Where, \( \alpha, \beta \) and \( \gamma \) are large positive numbers. \( u \) is state variable and \( f \) is control variable. If the control \( f \) shown in the state wave equation we call that distributed optimal control but if we see that as an unknown boundary of wave equation we say that is boundary control problem. Also, \( K, \Phi_1 \) and \( \Phi_2 \) are some suitable \( C^1 \) mappings. For example

\[
\begin{align*}
K(u) &= (u - U_{exact})^2 \\
\Phi_1(u) &= (u(x,T) - W)^2 \\
\Phi_2(u) &= (u(x,T) - Z)^2
\end{align*}
\]

We solve distributed optimal control problem of wave equation in the second section then explain an algorithm based on to find its numerical solution. In the third section we explain exact controllability of boundary control of wave equation. In the fourth section we have some numerical examples and results.

2. DISTRIBUTED OPTIMAL CONTROL OF WAVE EQUATION

Suppose that we have one-dimensional nonlinear wave equation as follows:

\[
\begin{align*}
\varphi_{tt} - \varphi_{xx} + \varphi_t + \frac{\alpha}{2} K'(u) &= 0 \\
\varphi(x,T) + \frac{\beta}{2} \Phi_2'(u(x,T)) &= 0 \\
-\varphi_t(x,T) + \frac{\gamma}{2} \Phi_1'(u(x,T)) &= 0 \\
\varphi(1, t) &= \varphi(0, t) = 0 \\
f - \varphi &= 0
\end{align*}
\]

(2-6)

Where \( \Psi \) is \( C^1 \) mapping (for example \( \Psi = u, \sin(u) \) or \( \Psi = e^u \)). If \( \Psi = 0 \) equation (2-1) is linear wave equation, else it is semi-linear wave equation. This operator is state equation and \( f \) is control of the wave equation such that defined as distributed control of wave equation [13]. Distributed optimal control of linear or semi-linear wave equation as follows:

Minimize (1-3) respect to control \( f \) subject to equation (2-1) is true. (2-2)

Definition 1-1: System (2-1) is said to be globally exactly controllable in a suitable Hilbert space \( H \) if, for every initial state \( (u, z) \in H \) and every terminal state \( (W, Z) \in H \), there exists a control \( f \) such that the solution of (2-1) satisfies:

\[
u(x, T; f) = w(x), \quad u_t(x, T; f) = z(x) \quad \text{in} \ (0, 1)
\]

(2-3)

Now we have a theorem of controllability of one-dimensional wave equation.

Theorem 1-2: Let \( \Psi \in W^{2, p}_{loc}(\mathbb{R}) \) and there exist constants \( k > 0 \) and \( p \geq 1 \) such that:

\[
|\Psi'(u)| \leq k |u|^{p-1} \quad \forall u \in \mathbb{R}
\]

(2-4)

If \( T > 0 \), then the system (2-1) is globally exactly controllable in \( H^1_0(0, 1) \times L^2(0, 1) \). That is, for every initial state \( (w, z) \) and every terminal state \( (W, Z) \) in \( H^1_0(0, 1) \times L^2(0, 1) \), there exists a control \( f \in C([0, T]; L^2(0, 1)) \) such that the solution \( u = u(x, t; f) \) of (1-4) satisfies:

\[
u(x, T; f) = W, \quad u_t(x, T; f) = Z \quad \text{in} \quad x \in (0, 1)
\]

(2-5)

PROOF. SEE [15].

Suppose Lipschitz function \( \Psi(u) \) and exact controllability problem (2-2) with local or global distributed control \( f \) through the use of Lagrange multiplier rule [14], the optimal control problem of (2-2) can be converted into the following system of equations for which an optimal solution may be determined:

\[
\begin{align*}
u_t - u_{xx} + \Psi(u) &= f \\
u(x, 0) &= w(x) \\
u_t(x, 0) &= z(x) \\
u(0, t) &= \nu(1, t) = 0
\end{align*}
\]

(2-7)

\[
\begin{align*}
\varphi_{tt} - \varphi_{xx} + \varphi_t + \frac{\alpha}{2} K'(u) &= 0 \\
\varphi(x,T) + \frac{\beta}{2} \Phi_2'(u(x,T)) &= 0 \\
-\varphi_t(x,T) + \frac{\gamma}{2} \Phi_1'(u(x,T)) &= 0 \\
\varphi(1, t) &= \varphi(0, t) = 0 \\
f - \varphi &= 0
\end{align*}
\]

(2-8)
The equation (2.6) is an initial-terminal value problem that should be solved.

\[ u_t - u_{xx} + \Psi(u) = \varphi \]

\[ u(x, 0) = w(x) \]

\[ u_t(x, 0) = z(x) \]

\[ u(0, t) = u(1, t) = 0 \]

\[ \varphi_t - \varphi_{xx} + \varphi \Psi'(u) + \frac{\alpha}{2} K'(u) = 0 \]

\[ \varphi(x, T) + \frac{\gamma}{2} \Phi'(u_t(x, T)) = 0 \]

\[ -\varphi_t(x, T) + \frac{\beta}{2} \Phi'_t(u(x, T)) = 0 \]

\[ \varphi(1, t) = \varphi(0, t) = 0 \]

(2.7)

The system (2.7) is an initial-terminal value problem that has exact solution. We can solve the state equation with forward in time and adjoint equation should be solved with backward in time. We use the shooting method for solving the initial value problem follows:

\[ u_t - u_{xx} + \Psi(u) = \varphi \]

\[ u(x, 0) = w(x) \]

\[ u_t(x, 0) = z(x) \]

\[ u(0, t) = u(1, t) = 0 \]

\[ \varphi_t - \varphi_{xx} + \varphi \Psi'(u) + \frac{\alpha}{2} K'(u) = 0 \]

\[ \varphi(1, t) = \varphi(0, t) = 0 \]

\[ \varphi(x, 0) = \omega(x) \]

\[ \varphi(x, 0) = \theta(x) \]

(2.8)

We should find the unknown functions \( \omega(x) \) and \( \theta(x) \) such that the terminal values of adjoint equation hold:

\[ F_1 = \varphi(x, T) + \frac{\gamma}{2} \Phi'_t(u_t(x, T)) = 0 \]

\[ F_2 = -\varphi_t(x, T) + \frac{\beta}{2} \Phi'_t(u_t(x, T)) = 0 \]

(2.9)

For solving this problem with shooting method we first choose initial functions \( \omega(x) \) and \( \theta(x) \) then we solve the system (2.8) and use Newton method for updating \( \omega(x) \) and \( \theta(x) \) such that (2.9) is established [15, 16]. We use the partitioning \( 0=x_0,...,\xi_i=1 \) and \( 0=t_0,...,t_i=T \) to get (2.10)

\[ (\omega_2, \theta_2, ..., \omega_{i-1}, \theta_{i-1}) \rightarrow J^{-1}F(\omega_2, \theta_2, ..., \omega_{i-1}, \theta_{i-1}) \]

(2.10)

To calculate the Jacobin matrix \( J \) we use the approximate of partial derivatives \( \partial F/\partial \vartheta_i \) and \( \partial F/\partial \omega_i \):

\[ \frac{\partial F}{\partial \omega_i} \approx \frac{1}{\Delta \omega_i} \left( F(\omega_1, \theta_2, ..., \theta_{i-1}, \omega_i + \Delta \omega_i, \theta_i) - F(\omega_1, \theta_2, ..., \theta_{i-1}, \theta_i) \right) \]

\[ \frac{\partial F}{\partial \theta_i} \approx \frac{1}{\Delta \theta_i} \left( F(\omega_1, \theta_2, ..., \omega_i, \theta_i + \Delta \theta_i, \theta_{i+1}, ..., \theta_j) - F(\omega_1, \theta_2, ..., \omega_i, \theta_{i+1}, ..., \theta_j) \right) \]

(2.11)

and Jacobin matrix is:

\[ J_{2i-1} = \frac{\partial F}{\partial \omega_{i-1}}, \quad J_{2i} = \frac{\partial F}{\partial \theta_{i-1}} \]

(2.12)

In each stage of calculating ith column of the Jacobin matrix \( J(\omega_i, \theta_j) \) we need to return to (2.8) and solve that system of PDEs. Hence we solve the system (2.8) 21-3 time in each stage of the Newton method. To get high accuracy one needs to increase \( I \). For example if for \( I=100 \) more time and computational cost the PDEs (2.8) is solved 197 times in each stage of Newton method. This requires time and cost. Therefore, to solve this problem we use parallel computing to find the Jacobi matrix. We discretize (2.8) and solve with finite difference method by central derivative approximation and notation \( u(x, t_0) = u^n \) and \( \varphi(x, t_0) = \varphi^n \) and then partial derivatives as follows:

\[ q_{ij} = \frac{\partial u^n_{i+1}}{\partial \omega_j}, \quad r_{ij} = \frac{\partial \varphi^n_{i+1}}{\partial \omega_j} \]

\[ p_{ij} = \frac{\partial u^n_{i+1}}{\partial \theta_j}, \quad b_{ij} = \frac{\partial \varphi^n_{i+1}}{\partial \theta_j} \]

(2.13)

We use from discretization of PDEs (2.8) and partial derivative of them rather then \( \omega_i \) and \( \theta_j \) for constructing a recursive relation to build (2.13) for \( i, j = 1, ..., I \) and \( n=1, ..., N \). We use the following equation for the Jacobin matrix:

\[ J_{2i-1,j-1} = p_{i+1,j} + \frac{\gamma}{2} \left( \frac{q_{i+1,j+1} - q_{i+1,j-1}}{\Delta t} \right) \frac{u^n_{i+1} - u^n_{i+1-1}}{\Delta t^N} \]

\[ J_{2i-1,j} = b_{i+1,j} - \frac{\gamma}{2} \left( \frac{q_{i+1,j+1} - q_{i+1,j-1}}{\Delta t} \right) \frac{u^n_{i+1} - u^n_{i+1-1}}{\Delta t^N} \]

(2.14)

In this case we don’t need to solve the partial differential equations a large number of times and we have only four 3-dimensional matrices \( q, r, b, p \). Numerical examples are given Section 4 to illustrate the effectiveness of this approach.

3- Optimal Boundary Control of Wave Equation

Now suppose that control variables shown as unknown boundary condition of linear or semi linear wave equation as follows:

\[ u_t - u_{xx} + f(u) = V \]

\[ u(x, 0) = u_0 \]

\[ u_t(x, 0) = u_1 \]

\[ u(X, T) = W \]

\[ u_t(X, T) = Z \]

\[ u(0, t) = g_L \]

\[ u(1, t) = g_R \]

(3.1)

Where \( g_L \) and \( g_R \) are control functions, \( W \) and \( Z \) are given target function, \( u_0 \) and \( u_1 \) are given initial conditions, \( V \) is a
function depended on \(x, t\) and \(f \in C^1(R)\) is a function of \(u\). We want to find the controls \(g_1\) and \(g_2\) such that PDE of (3-1) satisfies. Therefore, we convert (3-1) to optimal control problem, so the target functions \(W\) and \(Z\) are changed into the cost functional as follows:

\[
J(u, g) = \frac{\alpha}{2} \int_0^T \left[ u - U \right] \, dt + \frac{\sigma}{2} \int_0^1 \left[ u(x, T) - W(x) \right]^2 \, dx \\
+ \frac{\beta}{2} \int_0^1 (u(x, T) - Z(x))^2 \, dx + \frac{1}{2} \int_0^T \left[ g_L^2 + |g_R|^2 \right] \, dt
\]

Then the PDE (3-1) is shown as the state equation as follows:

\[
\begin{align*}
\frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left[ \phi(u, x, t) \right] &+ f(u) = V \\
\frac{\partial u}{\partial t}(x, 0) &+ g_L = 0 \\
\frac{\partial u}{\partial t}(1, t) &+ g_R = 0
\end{align*}
\]

(3-3)

Through the use of Lagrange multiplier rule, the optimal control problem of (3-3) may be converted into the following system of equations which an optimal solution may be determined:

\[
\begin{align*}
\phi_t - \phi_{xx} + f'(u)\phi &= -\alpha(u - U) \\
\phi(0, t) &= 0 \\
\phi(1, t) &= \theta(t) \\
\phi_t(x, T) - \beta(u(x, T) - Z(x)) &= 0 \\
\phi_t(x, 0) &= \sigma(u(x, T) - W(x))
\end{align*}
\]

(3-4)

Now as in the previous section, the control variables are omitted. The system (3-4) and (3-3) converting to IVP system:

\[
\begin{align*}
\frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left[ \phi(u, x, t) \right] &+ f(u) = V \\
\frac{\partial u}{\partial t}(x, 0) &= u_0 \\
\frac{\partial u}{\partial t}(1, t) &= \phi(1, t) = 0 \\
\phi_t(x, T) - \beta(u(x, T) - Z(x)) &= 0 \\
\phi_t(x, 0) &= \phi(1, t) = \theta(t)
\end{align*}
\]

Here we want to find the root of \(F\) to determine unknown initial conditions \(\omega\) and \(\theta\).

\[
\begin{align*}
F_1 &= \phi(x, T) + \beta(u(x, T) - Z(x)) = 0 \\
F_2 &= \phi_t(x, T) - \sigma(u(x, T) - W(x)) = 0
\end{align*}
\]

(3-5)

As before, section we use simple shooting method to solve (3-5) and (3-6) and use (1-16) to construct recursive relation to build Jacobin matrix \(J\).

4. Numerical results

In this section example 1 expresses distributed optimal control problem and example 2 describes boundary control problem of wave equation.

Example 1. Consider the linear wave equation \(\Psi(u) = 0\), \(\Phi_1 (u) = (u \cdot x) - W(x)^2\), and \(K(u) = (u-U)^2\) for given function \(W(x) = 0\), \(Z(x) = 2\pi \sin 2\pi x\) and \(U = \sin 2\pi x \sin 2\pi t\). The terminal time is \(T = 1\) and \(\alpha \beta \gamma = 10^3\).

We solve this example with tolerance \(10^8\) and discretization of \(Q = (0, 1) \times (0, T)\) with step size \(\Delta t\) and \(\Delta l = (\Delta t/ \Delta x)^2\) such that we have exact control \(f = 0\) we have reported the numerical results in Tables (1 to 3).

**Table 1: Solution for \(s=1\)**

| \(|F\)| | \(|f| L_2\)| | Iteration | |
|---|---|---|---|
| \(\Delta x\) | 0.1 | 0.05 | 0.01 | 0.005 |
| 5.5E-11 | 1.13E-10 | 3.09E-09 | 9.12E-09 |
| 1.87E-08 | 2.01E-12 | 1.29E-21 | 2.99E-22 |
| Iteration | 1 | 1 | 1 | 2 |

**Table 2 Solution for \(s=0.9\)**

| \(|F\)| | \(|f| L_2\)| | Iteration | |
|---|---|---|---|
| \(\Delta x\) | 0.1 | 0.05 | 0.01 | 0.005 |
| 2.44E-11 | 5.33E-11 | 1.04E-09 | 3.59E-09 |
| 1.05E-10 | 2.00E-13 | 7.48E-20 | 7.03E-23 |
| Iteration | 1 | 1 | 1 | 1 |

**Table 3 Solution for \(s=0.8\)**

| \(|F\)| | \(|f| L_2\)| | Iteration | |
|---|---|---|---|
| \(\Delta x\) | 0.1 | 0.05 | 0.01 | 0.005 |
| 2.73E-11 | 7.79E-11 | 9.24E-10 | 3.53E-09 |
| 3.77E-09 | 6.87E-12 | 4.25E-19 | 1.71E-22 |
| Iteration | 1 | 1 | 1 | 1 |
Example 2. Suppose that semi linear wave equation
\[ f(u) = \sin u, \ W(x) = 0, \ Z(x) = 0, \ V = 0, \ \alpha = 0, \ \alpha, \beta = 10^3 \] and
terminal time \( T = 3 \).
We solve this example with tolerance \( 10^{-8} \) and
discretization of \( Q = (0, 1) \times (0, T) \) with step size \( \Delta x \) and
\( \Delta t = (\Delta t/\Delta x)^2 \). We have reported the numerical
results in Tables (4 and 5) and Figures (1 and 2). Note that
tolerance for \( \Delta x = 0.005 \) is \( 10^{-3} \).

**Figure 1. Numerical results for Example 2. s=1. From left to right: Control \( g_L, g_R \), \( u(x,T) \) and \( u_t(x,T) \) and From Top to Bottom Figures are for \( \Delta x = 0.1, 0.05, 0.01 \) and \( 0.005 \)**
As Matsuo, et al., (2014) showed that compared with plain diagonal preconditioning, Block-Jacobi preconditioning improves the speed of convergence. It reduces the number of inner and outer iterations leading to substantial reductions of execution time – in spite of the fact that constructing these pre-conditioners needs more time [5]. With comparing of Tables (1, 2 and 3) we see the best solution of example 1 is in Table (2) with $s=0.9$ where with decreasing $\Delta x$ high accuracy is increased. From example 2 in the Table (4), we see that with decreasing $\Delta x$ the solution has low accuracy and for $\Delta x = 0.005$ the solution is unacceptable but for $s=0.5$ in the Table (5) this problem is solved and the solution has high accuracy.
Table 4 Numerical results for s=1

<table>
<thead>
<tr>
<th>Δx</th>
<th>0.1</th>
<th>0.05</th>
<th>0.01</th>
<th>0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(gL,gR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5 Numerical results for s=0.5

<table>
<thead>
<tr>
<th>Δx</th>
<th>0.1</th>
<th>0.05</th>
<th>0.01</th>
<th>0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(gL,gR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration</td>
<td>5</td>
<td>6</td>
<td>18</td>
<td>7</td>
</tr>
</tbody>
</table>

References


