The Role of Initial Value Solvers in a Moving Mesh Method

by

Lin (Linda) Ju

B.Sc, Jilin University, China, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN THE DEPARTMENT
OF
MATHEMATICS

APPROVAL

Name:	Lin (Linda) Ju
Degree:	Master of Science
Title of thesis:	The Role of Initial Value Solvers in a Moving Mesh Method
Examining Committee:	Dr. David Muraki Chair
	Dr. Robert Russell Senior Supervisor
	Dr. John Stockie Supervisor
	Dr. Manfred Trummer Supervisor
	Dr. J.F Willams External Examiner
Date Approved:	April 2, 2007

Abstract

This thesis investigates the role of different initial value solvers in moving mesh software for solving second-order parabolic partial differential equations. We confine our attention primarily to the effect of the implicit ODE solver DDASSL used in the software MOVCOL of Huang and Russell, which is based on a moving collocation method. The advantages and limitations of Backward Differentiation Formula (BDF) methods and Implicit Runge-Kutta (IRK) methods are discussed in this particular context and the alternative ODE solver PSIDE, which is based on a four-stage RADAU IIA algorithm, is examined. The new interface of PSIDE with MOVCOL is discussed in some detail. Its performance is compared with DDASSL for several numerical problems.

Acknowledgments

Firstly, I would like to express my deepest gratitude and sincerest appreciation to my supervisor Dr. Robert D. Russell for all his patience, guidance, encouragement and constant support during my study at Simon Fraser University.

I would also like to thank my committee members, Dr. John Stockie, Dr. Manfred Trummer and Dr. J.F. Willams for their feedback and helpful comments.

Moreover, special thanks to my colleague Xiangmin Xu, for all her help and encouragement during this thesis work.

Finally, my thanks also go out to everyone in the applied mathematics department for so many memorable times spent together.

Dedication

To my parents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many time dependent partial differential equations (PDEs) arise in physics and engineering. They have been used to predict and control the dynamic properties of physical constructions, weather, heating and melting of metals, flow of air past cars and airplanes, etc. They describe changes of weather, density, velocity, pressure, temperature, etc. For these physical processes, people wish to know not only how the solution changes in space, but also how the solution changes in time.

In this thesis, we focus on using numerical methods for solving time dependent PDEs, such as the following general second-order PDE

$$\mathbf{F}(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_t, \mathbf{u}_{xt}) = \frac{\partial}{\partial x}\mathbf{G}(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_t, \mathbf{u}_{xt}) \tag{1.1}$$

for $x^L(t) < x < x^R(t)$ and $t_a < t \leq t_b$ supplemented with the boundary conditions

$$\mathbf{B}^L(t, x^L, x_t^L, \mathbf{u}(x^L, t), \mathbf{u}_x(x^L, t), \mathbf{u}_{xx}(x^L, t), \mathbf{u}_t(x^L, t), \mathbf{u}_{xt}(x^L, t)) = 0 \tag{1.2}$$

$$\mathbf{B}^R(t, x^R, x_t^R, \mathbf{u}(x^R, t), \mathbf{u}_x(x^R, t), \mathbf{u}_{xx}(x^R, t), \mathbf{u}_t(x^R, t), \mathbf{u}_{xt}(x^R, t)) = 0 \tag{1.3}$$

for $t_a < t \leq t_b$ and the initial condition

$$\mathbf{u}(x, t_a) = \mathbf{U}(x), \quad x^L(t_a) \leq x \leq x^R(t_a) \tag{1.4}$$

where $\mathbf{F}$, $\mathbf{G}$, $\mathbf{B}^L$, $\mathbf{B}^R$, $\mathbf{u}$ and $\mathbf{U}$ are vector-valued functions for this PDE system.

Various numerical discrete methods have been developed to approximate the exact solutions of PDEs. Most of them involve iterative procedures which depend upon how the mesh is chosen. Normally, there are two main choices. One choice is the fixed mesh, where the number and location of mesh points are fixed. The other is the adaptive mesh approach, in which the mesh points are changed appropriately as time and the solution change. It has been shown that for some time dependent PDEs, which have large solution variations, such as boundary layers, shock layers or contact surfaces, an adaptive mesh is better than a fixed mesh. As pointed out in [11], by using adaptive mesh methods, one can achieve significant improvements in both accuracy and efficiency.

The adaptive mesh methods can be divided into two categories: static and dynamic. For the static method, the equation and the discrete solution are initially defined on a given mesh. During the calculation, a new mesh is constructed, based on the properties of a certain function that measures the goodness of approximation. This new mesh might have a different number of mesh points from the old mesh. The solution is then interpolated from the old mesh to the new mesh, and a new approximation to the solution is done on the new mesh. The redistribution of the mesh points, the possible addition of new nodes and the interpolation of dependent variables from the old mesh to the new mesh are all done at a fixed time [10].

The dynamic method is known as the moving mesh method. A mesh equation involving mesh speeds is used to move a fixed number of mesh points. The original PDE and the mesh equation are generally solved simultaneously for the physical solution and the mesh. The key in developing a moving mesh method lies in formulating a satisfactory mesh equation. In [10], several moving mesh partial differential equations(MMPDEs) based on the equidistribution principle are derived and studied both theoretically and numerically. Among them, the MMPDE4 and MMPDE6 which we consider later are easy to implement. Moreover, it has been found that under general conditions, for the above MMPDEs, not only are mesh crossings guaranteed not to occur, but the meshes retain equidistribution for the monitor function [11]. Among moving mesh methods, there are moving finite element methods (MFE) [19], [20] and moving finite difference methods [5].

In this thesis, we investigate the software MOVCOL from Huang and Russell [12], which is composed of five parts: Driver, MOVCOL, DDASSL, DLINPK and DAUX. The basic algorithm for this software is a moving collocation method, which uses a cell averaging cubic Hermite collocation discretization for the physical PDEs and a central finite difference discretization for the MMPDE. During the discretization, an ODE system

$$F(t, y, y') = 0 \tag{1.5}$$

where F, y and y' are vector valued functions, is constructed.

The original MOVCOL software uses the Fortran code DDASSL to solve the resulting initial value problems. However, DDASSL has some computational limitations due to the properties of its basic algorithm, BDF(Backward differentiation formula) methods. In this thesis, we examine the possibilities of using other software to replace DDASSL in MOVCOL.

Among the variety of software which could be considered, we pay attention to software which could solve an ODE system of form (1.5) since this formula has advantages in the numerical computation [4]. As one type of the one-step methods, IRK (Implicit Runge-Kutta) methods have their potential advantages compared with multi-step methods such as BDF methods. Based on the above considerations, the software PSIDE is a reasonable choice. This software is not only based on a four-stage RADAU IIA algorithm, which is one of the IRK methods, but also is suitable for solving the problem (1.5).

We develop the new interface of PSIDE with MOVOL and also explore the case of putting DDASSL and PSIDE with MOVCOL together. Numerical experiments have been done for a simple problem, heat conduction problem, the Gray-Scott problem and a scalar combustion problem.

An outline of this thesis is as follows. In Chapter 2, we present the fundamental ideas of moving mesh methods and a moving collocation method, the basic algorithm for MOVCOL and some issues related to the implementation of MOVCOL. In Chapter 3, DDASSL and its basic algorithm, a BDF method, is introduced. We discuss the possibility of using the alternative ODE solver PSIDE to replace DDASSL in Chapter 4. In Chapter 5, the resulting two versions of MOVCOL are applied to several bench mark problems. Finally, Chapter 6 contains conclusions and comments.

Chapter 2

Moving Mesh Method and MOVCOL Code

In this chapter, we focus on the software MOVCOL from Huang and Russell in 1996 [12]. We start with a review of moving mesh methods, based on the equidistribution principle, and several versions of moving mesh partial differential equations (MMPDEs). Then we describe the discretization process for MMPDEs and physical PDEs with a moving collocation method and introduce the basic algorithm for MOVCOL.

2.1 Moving mesh method

Moving mesh methods have been widely used in the last twenty years for solving PDEs which have rapidly solution change behaviors.

2.1.1 Equidistribution Principle

Most moving mesh methods are based on the idea of the equidistribution principle (EP), which is first presented by De Boor [1]. Since the EP plays a fundamental role in formulating MMPDEs, we first give a detailed description of it.

Consider a one dimensional time independent PDE

$$\mathbf{F}(x, \mathbf{u}, \mathbf{u}_x) = \frac{\partial}{\partial x}\mathbf{G}(x, \mathbf{u}, \mathbf{u}_x) \tag{2.1}$$

for $a = x^L < x < x^R = b$ supplemented with the boundary conditions

$$\mathbf{B}^L(x^L, x^L, \mathbf{u}(x^L), \mathbf{u}_x(x^L), \mathbf{u}_{xx}(x^L)) = 0 \tag{2.2}$$

$$\mathbf{B}^R(x^R, x^R, \mathbf{u}(x^R), \mathbf{u}_x(x^R), \mathbf{u}_{xx}(x^R)) = 0 \tag{2.3}$$

where $\mathbf{F}$, $\mathbf{G}$, $\mathbf{B}^L$, $\mathbf{B}^R$ and $\mathbf{u}$ are vector-valued functions.

We introduce a monitor function

$$M = M(x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}), \tag{2.4}$$

which provides some measurement of the difficulty of approximating $u(x)$ on the bounded interval $[a, b]$. Then a mesh is selected as follows:

$$\pi_h : a = x_1 < x_2 < \cdots < x_N = b \tag{2.5}$$

is chosen to evenly distribute M among the subintervals, i.e,

$$\int_{x_1}^{x_2} M(x)dx = \cdots = \int_{x_{N-1}}^{x_N} M(x)dx \tag{2.6}$$

In other words, the EP involves selecting the mesh points so that some measure of the solution error is equal over each subinterval. It also could be written down as

$$\int_a^{x_j} M(x)dx = \frac{j-1}{N-1}\sigma, \quad j = 1, \cdots, N \tag{2.7}$$

where

$$\sigma = \int_a^b M(x)dx. \tag{2.8}$$

We set $[0, 1]$ as the computational domain for a computational variable ξ in our future computations. For a mapping $x(\xi)$ from $[a, b]$ to $[0, 1]$, the mesh is defined in the following way:

$$x_j = x(\xi_j), \quad j = 1, \cdots, N \tag{2.9}$$

where

$$\xi_j = \frac{j-1}{N-1}, \quad j = 1, \cdots, N. \tag{2.10}$$

is a uniform mesh on [0,1].

To define $x(\xi)$, we expand the equidistribution principle to a continuous form and from the time-independent case to the time-dependent case under the above computational domain. Specifically, (2.7) is changed to

$$\int_0^{x(\xi,t)} M(\tilde{x}, t)d\tilde{x} = \xi\theta(t) \tag{2.11}$$

where

$$\theta(t) = \int_0^1 M(\tilde{x}, t) d\tilde{x} \tag{2.12}$$

The following figure shows the adaptive mesh points chosen for the Burgers equation at time $t = 0.4375$.

$$u_t = \epsilon u_{xx} - \frac{1}{2}(u^2)_x \quad 0 < x < 1, \quad t > 0, \quad \epsilon = 10^{-4}$$

$$u(0,t) = u(1,t) = 0 \quad t > 0$$

$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x), \quad 0 \le x \le 1.$$

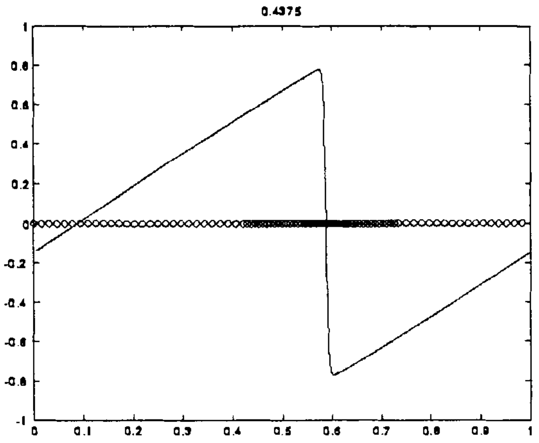

Figure 2.1: Numerical solution and mesh points for Burgers equation at time = 0.4375.

It indicates that there are more mesh points in the interval where the solution changes rapidly and fewer points in the interval where the solution is relative smooth. This result shows the reasonableness of the EP for adaptivity.

2.1.2 Moving Mesh Equations

For moving mesh methods, we solve the original partial differential equation and the mesh equation simultaneously to obtain both the physical solution approximation and the mesh results.

The above description of the EP indicates that the key in developing a moving mesh method lies in formulating a satisfactory mesh equation. In addition to the capability of concentrating a sufficient number of points in regions of rapid variation of the solution, a satisfactory mesh equation should be simple, easy to program, and reasonably insensitive to the choice of its adjustable parameters. As compared with the problem of discretizing the underlying physical equation, this task is somewhat artificial. That is, the construction of a moving mesh equation must be guided by both physical arguments and effective numerical principles.

The moving mesh partial differential equation (**MMPDE**) is the equation which shows how the mesh changes as the time and physical solution change, so that the nodes will keep concentrated in regions of rapid variation of the solution. It is based on the equidistribution principle we mentioned in section 2.1.1. Differentiating (2.11) with respect to ξ once, we can get

$$M(x(\xi,t),t)\frac{\partial}{\partial\xi}x(\xi,t)=\theta(t) \tag{2.13}$$

Differentiating (2.11) with respect to ξ twice, it yields

$$\frac{\partial}{\partial\xi}\{M(x(\xi,t),t)\frac{\partial}{\partial\xi}x(\xi,t)\}=0 \tag{2.14}$$

In [23], if we differentiate (2.13) with respect to t, it is the conservative form of an MMPDE

$$\frac{\partial}{\partial\xi}(M\dot{x})+\frac{\partial M}{\partial t}\frac{\partial x}{\partial\xi}=\dot{\theta}(t). \tag{2.15}$$

after dividing by $\frac{\partial x}{\partial\xi}$ then combining with (2.13), we can also get

$$\frac{\partial}{\partial x}(M\dot{x})+\frac{\partial M}{\partial t}=\frac{M\dot{\theta}(t)}{\theta}. \tag{2.16}$$

Since there is no term $\theta(t)$ in equation (2.14), we use the later time $t+\tau$ instead of t, so that the mesh satisfies

$$\frac{\partial}{\partial \xi}\{M(x(\xi,t+\tau),t+\tau)\frac{\partial}{\partial \xi}x(\xi,t+\tau)\} = 0. \tag{2.17}$$

We use Taylor expansions for the above equation,

$$\frac{\partial}{\partial \xi}x(\xi,t+\tau) = \frac{\partial}{\partial \xi}x(\xi,t) + \tau\frac{\partial}{\partial \xi}\dot{x}(\xi,t) + O(\tau^2), \tag{2.18}$$

and

$$\begin{aligned} M(x(\xi,t+\tau),t+\tau) = M(x(\xi,t),t) + \tau\dot{x}\frac{\partial}{\partial x}M(x(\xi,t),t) \\ +\tau\frac{\partial}{\partial t}M(x(\xi,t),t) + O(\tau^2). \end{aligned} \tag{2.19}$$

After dropping higher order terms, we can get

$$\frac{\partial}{\partial \xi}(M\frac{\partial \dot{x}}{\partial \xi}) + \frac{\partial}{\partial \xi}(\dot{x}\frac{\partial M}{\partial \xi}) = -\frac{\partial}{\partial \xi}(\frac{\partial M}{\partial t}\frac{\partial x}{\partial \xi}) - \frac{1}{\tau}\frac{\partial}{\partial \xi}(M\frac{\partial x}{\partial \xi}). \tag{2.20}$$

We call this MMPDE2. In principle, it is reasonable to drop out the term $\dot{x}\frac{\partial}{\partial \xi}M_{\xi}$ and $\frac{\partial x}{\partial \xi}\frac{\partial}{\partial t}M_{\xi}$ [10], thus we can get the following simplification:

$$\frac{\partial}{\partial \xi}\left(M\frac{\partial \dot{x}}{\partial \xi}\right) = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right). \tag{2.21}$$

This is MMPDE4.

Since the integration of the monitor function represents an error measurement in the interval, we define this error measure as W, which is generally related to some monitor function by

$$W_i = \int_{x_i}^{x_i+1} M(\tilde{x},t)d\tilde{x}, \tag{2.22}$$

where M is a certain monitor function. After discretizing the above formula

$$W = M\frac{\partial x}{\partial \xi} \tag{2.23}$$

and using midpoint rule, we can get

$$W_i \approx M_{i+\frac{1}{2}}(x_{i+1} - x_i). \tag{2.24}$$

On the other hand, the node speed is determined by

$$\dot{x}_{i+1} - \dot{x}_i = -\lambda(W_i - \bar{W}) \tag{2.25}$$

where λ is a positive parameter, W_i is an error indicator on the subinterval (x_i, x_{i+1}) and $\bar{W}$ is the average of the W_i values.

Eliminating $\tilde{W}$ by substracting equation (2.25) on two consecutive intervals, we can get

$$\dot{x}_{i+1} - 2\dot{x}_i + \dot{x}_{i-1} = -\lambda(W_i - W_{i-1}) \tag{2.26}$$

If we denote λ by $\frac{1}{\tau}$ and use (2.23), we can regard (2.26) as a centered finite difference approximation of an MMPDE. We call this formula MMPDE6,

$$\frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right). \tag{2.27}$$

The constructions are very different and in their final forms, the moving mesh equations appear to be quite different from each other. Some versions of MMPDEs are derived from MMPDE4 and MMPDE6. We describe one from MMPDE4 in the next subsection.

2.1.3 Spatial Smoothing Moving Mesh Equation

There is another important version of an MMPDE. From (2.14), we notice that the initial transformation is assumed to be a smooth transformation [13]. For most problems which have large solution variations, their corresponding monitor functions are generally non-smooth in space. Thus we introduce a smoothing monitor $\widetilde{M}$ which satisfies the following equation which makes this transformation smooth:

$$\widetilde{M} = \frac{M}{1 - \lambda^2\Delta},$$

and

$$\frac{\widetilde{M}}{\partial \xi}(0,t) = \frac{\widetilde{M}}{\partial \xi}(1,t) = 0,$$

where λ is a positive number and $\Delta = \frac{\partial^2}{\partial \xi^2}$.
Thus we obtain a smooth verision of MMPDE4

$$\frac{\partial}{\partial \xi}(\frac{\dot{\hat{n}}}{M}) = -\frac{1}{\tau}\frac{\partial}{\partial \xi}(\frac{\dot{n}}{M}) \tag{2.28}$$

where $\hat{n} = (I - \lambda^2\Delta)(\frac{\partial}{\partial \xi})^{-1}$, $n = (\frac{\partial}{\partial \xi})^{-1}$ and τ is the smoothing parameter.

Numerical experiments in [13] indicate that the spatial smoothing is often essential if the solution to the physical PDE changes rapidly. Furthermore, the numerical experiments show this smoothing process at least will not deteriorate the computation results significantly.

For moving mesh methods, the physical PDE and moving mesh equation are often solved simultaneously for both the physical solution $u(x,t)$ and the coordinate transformation $x(\xi,t)$. Two advantages of this simultaneous solution approach are that the interpolation of the physical solution between different meshes is unnecessary and that, after discretizing the physical PDE and MMPDE in space, the method of lines approach can be employed using standard software to solve the resulting ODE system.

2.2 Moving Collocation Method

A new moving mesh method is introduced for solving the time dependent partial differential equations (PDE) in divergence form. The method uses a cell averaging cubic Hermite collocation discretization for the physical PDEs and a three point finite difference discretization for the moving mesh equations. The collocation method can offer higher order convergence, provide an approximating solution continuously, and be implemented easily while staying simple for general boundary conditions [12]. Therefore, it remains popular among a variety of ODE and PDE software, including the moving mesh software.

2.2.1 Physical PDE

We use a cell averaging cubic Hermite collocation method to discretize the physical PDE. The cubic Hermite polynomial $v(x,t)$ is the approximation for the solution $u(x,t)$ at time $t \in [t_a, t_b]$ on the fixed number of mesh points

$$X_1(t) := x^L(t) < \ldots < X_{npts}(t) := x^R(t) \tag{2.29}$$

where *npts* is the number of mesh points and

$$\begin{aligned} v(x,t) = {} & v_i(t)\phi_1(s^{(i)}) + v_{x,i}(t)H_i(t)\phi_2(s^{(i)}) \\ & + v_{i+1}(t)\phi_3(s^{(i)}) + v_{x,i+1}(t)H_i(t)\phi_4(s^{(i)}) \end{aligned} \tag{2.30}$$

for $x \in [X_i(t), X_{i+1}(t)]$, $i = 1, 2, \ldots, npts - 1$, where $v_i(t)$ and $v_{x,i}(t)$ are defined as the approximation of $u(X_i(t), t)$ and $u_x(X_i(t), t)$.
The local coordinate $s^{(i)}$ is defined by

$$s^{(i)} := (x - X_i(t))/H_i(t) \tag{2.31}$$

$$H_i(t) := X_{i+1}(t) - X_i(t), \tag{2.32}$$

and the respective shape functions are

$$\begin{aligned} \phi_1(s) &:= (1+2s)(1-s)^2 \\ \phi_2(s) &:= s(1-s)^2 \\ \phi_3(s) &:= (3-2s)s^2 \\ \phi_4(s) &:= (s-1)s^2 \end{aligned} \tag{2.33}$$

for $x \in [X_i(t), X_{i+1}(t)]$, $i = 1, 2, \ldots, npts - 1$. We have

$$v_x(x,t) = (1/H_i)(v_i \frac{\partial \phi_1}{\partial s} + H_i v_{x,i} \frac{\partial \phi_2}{\partial s} + v_{i+1} \frac{\partial \phi_3}{\partial s} + H_i v_{x,i+1} \frac{\partial \phi_4}{\partial s})$$

$$v_{xx}(x,t) = (1/H_i^2)(v_i \frac{\partial^2 \phi_1}{\partial s^2} + H_i v_{x,i} \frac{\partial^2 \phi_2}{\partial s^2} + v_{i+1} \frac{\partial^2 \phi_3}{\partial s^2} + H_i v_{x,i+1} \frac{\partial^2 \phi_4}{\partial s^2})$$

$$\begin{aligned} v_t(xt) = & \frac{\partial v_i}{\partial t}\phi_1 + (\frac{\partial v_{x,i}}{\partial t} H_i + v_{x,i}\frac{\partial H_i}{\partial t})\phi_2 \\ & + \frac{\partial v_{i+1}}{\partial t}\phi_3 + (\frac{\partial v_{x,i+1}}{\partial t} H_i + v_{x,i+1}\frac{\partial H_i}{\partial t})\phi_4 \\ & - v_x(x,t)(\frac{\partial X_i}{\partial_t} + s^{(i)}\frac{\partial H_i}{\partial_t}) \end{aligned}$$

$$\begin{aligned} v_{x,t}(x,t) = & (\frac{1}{H_i})(\frac{\partial v_i}{\partial t}\frac{\partial \phi_1}{\partial s} + (\frac{\partial v_{x,i}}{\partial t} H_i + v_{x,i}\frac{\partial H_i}{\partial t})\frac{\partial \phi_2}{\partial s}) \\ & (\frac{1}{H_i})(\frac{\partial v_{i+1}}{\partial t}\frac{\partial \phi_3}{\partial s} + (\frac{\partial v_{x,i+1}}{\partial t} H_i + v_{x,i+1}\frac{\partial H_i}{\partial t})\frac{\partial \phi_4}{\partial s}) \\ & - \frac{v_x(x,t)}{H_i}(\frac{\partial H_i}{\partial t}) - v_{x,x}(x,t)(\frac{\partial X_i}{\partial t} + s^{(i)}\frac{\partial H_i}{\partial t}) \end{aligned} \tag{2.34}$$

where ϕ_j, $(\frac{\partial \phi_j}{\partial s})$ and $(\frac{\partial^2 \phi_j}{\partial s^2})$ are functions of $s^{(i)}$.

Based on the conservation law, our physical PDE (1.1) satisfies

$$\int_{x^L}^{x^R} \mathbf{F} dx = \mathbf{G}|_{x=x^R,t} - \mathbf{G}|_{x=x^L,t} \tag{2.35}$$

for $t \in (t_a, t_b]$.

Then we can get the cell average for each half of $[X_i, X_{i+1}]$

$$\int_{X_i}^{\frac{X_i+X_{i+1}}{2}} \mathbf{F} dx = \mathbf{G}_{\frac{i+1}{2}}(t) - \mathbf{G}_i(t)$$

$$\int_{\frac{X_i+X_{i+1}}{2}}^{X_{i+1}} \mathbf{F} dx = \mathbf{G}_{i+1}(t) - \mathbf{G}_{\frac{i+1}{2}}(t) \tag{2.36}$$

for $i = 1, \ldots, npts - 1$ and $t \in (t_a, t_b]$, where

$$\mathbf{G}_i(t) := \mathbf{G}(t, x, \mathbf{v}, \mathbf{v}_x, \mathbf{v}_t, \mathbf{v}_{xt}) \,|_{x=X_i}$$

$$\mathbf{G}_{\frac{i+1}{2}}(t) := \mathbf{G}(t, x, \mathbf{v}, \mathbf{v}_x, \mathbf{v}_t, \mathbf{v}_{xt}) \,|_{x=\frac{X_{i+1}+X_i}{2}}$$

$$\mathbf{G}_{i+1}(t) := \mathbf{G}(t, x, \mathbf{v}, \mathbf{v}_x, \mathbf{v}_t, \mathbf{v}_{xt}) \,|_{x=X_{i+1}} \,. \tag{2.37}$$

After that, we use the piecewise linear approximation to get

$$\mathbf{F} \approx \mathbf{F}(X_{i,1}, t)(x - X_{i,2})/(X_{i,1} - X_{i,2}) + \mathbf{F}(X_{i,2}, t)(x - X_{i,1})/(X_{i,2} - X_{i,1}) \tag{2.38}$$

where $X_{i,j} := X_i + s_j H_j$, $j = 1, 2$.

The two Gauss Points on $[0, 1]$ are

$$s_1 = \frac{1}{2}(1 - 1/\sqrt{3}) \tag{2.39}$$

$$s_2 = \frac{1}{2}(1 + 1/\sqrt{3}). \tag{2.40}$$

After simplification, we can get

$$\mathbf{F}(X_{i,1}, t) = (\frac{1}{H_i})(-(1+2/\sqrt{3})\mathbf{G}_i(t) + (4/\sqrt{(3)})\mathbf{G}_{i+1/2}(t) + (1-2/\sqrt{3})\mathbf{G}_{i+1}(t)) \tag{2.41}$$

$$\mathbf{F}(X_{i,2}, t) = (\frac{1}{H_i})(-(1-2/\sqrt{3})\mathbf{G}_i(t) - (4/\sqrt{(3)})\mathbf{G}_{i+1/2}(t) + (1+2/\sqrt{3})\mathbf{G}_{i+1}(t)). \tag{2.42}$$

This conservative method is analyzed in [10].

2.2.2 MMPDE

Even though it has been proved that the above cubic Hermite collocation method can bring higher accuracy than a standard finite difference scheme [12], it has the price of more computation time and a more complicated computation. Our aim is to use a relatively simple method for the MMPDE since experience has shown that the accuracy of the mesh solution does not need to be as high as that of the physical solution [10]. Therefore, we consider using a finite difference scheme for the MMPDE.

We use the three point finite different method to discretize the moving mesh partial differential equations. Since MMPDE4 and MMPDE6 are the common choices in our further computation, we give the details of discretizing them by the centered finite difference scheme.

For MMPDE4,

$$\frac{\partial}{\partial \xi}\left(M\frac{\partial \dot{x}}{\partial \xi}\right) = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right), \tag{2.43}$$

we discretize it to obtain

$$\begin{aligned} &\frac{M_{i+1}+M_i}{2(\frac{1}{n})^2}(\dot{x}_{i+1}-\dot{x}_i) - \frac{M_i+M_{i-1}}{2(\frac{1}{n})^2}(\dot{x}_i-\dot{x}_{i-1}) \\ &= -\frac{1}{\tau}[\frac{M_{i+1}+M_i}{2(\frac{1}{n})^2}(x_{i+1}-x_i) - \frac{M_i+M_{i-1}}{2(\frac{1}{n})^2}(x_i-x_{i-1})]. \end{aligned} \tag{2.44}$$

For MMPDE6,

$$\frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right), \tag{2.45}$$

it yields

$$\frac{1}{(\frac{1}{n})^2}[\dot{x}_{i+1}-2\dot{x}_i+\dot{x}_{i-1}] = -\frac{1}{\tau}[\frac{M_{i+1}+M_i}{2(\frac{1}{n})^2}(x_{i+1}-x_i) - \frac{M_i+M_{i-1}}{2(\frac{1}{n})^2}(x_i-x_{i-1})]. \tag{2.46}$$

Third order convergence in space for the moving collocation method has been demonstrated numerically in terms of the rate of convergence. It is slower than the traditional (fourth order) cubic Hermite collocation on a fixed mesh but much faster than the second order of the commonly used moving finite difference methods. The moving collocation method can also produce results for small and moderate numbers of mesh points with more accuracy [12].

2.3 Implementation of MOVCOL

Several moving mesh codes have been used in a variety of research fields now. The one we focus on is named MOVCOL, which was published by Huang and Russell in 1996. MOVCOL is designed to be as easy to use as possible, while providing enough flexibility and control for solving a wide variety of problems. In this subsection, we outline what a user must to do to solve a problem with MOVCOL and describe the basic algorithm of MOVCOL.

We emphasize that MOVCOL is designed for solving second order problems of the form

$$\mathbf{F}(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_t, \mathbf{u}_{xt}) = \frac{\partial}{\partial x}\mathbf{G}(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_t, \mathbf{u}_{xt}) \tag{2.47}$$

for $x^L(t) < x < x^R(t)$ and $t_a < t \leq t_b$ supplemented with the boundary conditions

$$\mathbf{B}^L(t, x^L, x_t^L, \mathbf{u}(x^L, t), \mathbf{u}_x(x^L, t), \mathbf{u}_{xx}(x^L, t), \mathbf{u}_t(x^L, t), \mathbf{u}_{xt}(x^L, t)) = 0 \tag{2.48}$$

$$\mathbf{B}^R(t, x^R, x_t^R, \mathbf{u}(x^R, t), \mathbf{u}_x(x^R, t), \mathbf{u}_{xx}(x^R, t), \mathbf{u}_t(x^R, t), \mathbf{u}_{xt}(x^R, t)) = 0 \tag{2.49}$$

for $t_a < t \leq t_b$ and the initial condition

$$\mathbf{u}(x, t_a) = \mathbf{U}(x), \quad x^L(t_a) \leq x \leq x^R(t_a) \tag{2.50}$$

where $\mathbf{F}$, $\mathbf{G}$, $\mathbf{B}^L$, $\mathbf{B}^R$, $\mathbf{u}$ and $\mathbf{U}$ are vector-valued functions of this PDE systems.

This software includes 5 parts: EXAMPLE, MOVCOL, DDASSL, DLINPK and DAUX.

EXAMPLE is a set of subroutines which is written by the user to define the physical PDEs, MMPDEs, boundary conditions, initial condition, monitor function, physical domain and computational domain. We write down drivers for several numerical problems in the Appendix.

As the core part of the code, MOVCOL uses the moving collocation method we mentioned in the last section to discretize the physical PDEs and MMPDEs.

The call to MOVCOL is

CALL MOVCOL(npde, npts, atol, touta, ntouta, par, iflag, rwork, lrw, iwork, liw)

For these input values, **npde** means the number of physical PDEs in (2.47); **npts** is the number of mesh points used in the computation. **Atol** and **rtol** represent the absolute and relative tolerance for time integration which is used in DDASSL; **touta** is the real array for the time integration and **ntouta** is the length of this array. The option vector **par** is related to many features of MOVCOL. Several of them are worth mentioning.

MOVCOL can solve four types of MMPDEs, from the fixed mesh case, MMPDE4 to MMPDE6 and the spatial smoothing MMPDE (2.28). If we do not specify, it will default to solve the last type of MMPDE. MOVCOL also defines **rwork** and **iwork** as the space to save real and integer work variables. These two arrays are very important since they take control of the interface of MOVCOL with the ODE solvers. The specification of the length of these variables can be quite sensitive here. Either too large or too small of the value of these numbers could lead the interface to fail to work. This constraint brings some difficulties for changing the current ODE solver DDASSL to an alternative one since the user has to understand all the basic algorithms for the software. If MOVCOL fails, the scalar variable **iflag** should be examined to see what specific error caused the difficulty. Those details are mentioned in the documentation of MOVCOL.

The design of the monitor function also can affect the accuracy and efficiency of the code. A common choice for the monitor function is the arc-length function

$$M(x) = \sqrt{1 + (u_x)^2}. \tag{2.51}$$

Other possible choices are

$$M(x) = 1 + \mid u \mid \tag{2.52}$$

and the curvature monitor function

$$M(x) = (\alpha + u_{xx}^2)^{\frac{1}{4}} \tag{2.53}$$

where α is a positive number.

During the discretization, an ODE system.

$$\mathbf{F}(t, \mathbf{y}', \mathbf{y}') = 0 \tag{2.54}$$

is constructed. This code uses the Differential Algebraic Equations (DAEs) solver DDASSL to solve this system.

The call to DDASSL is

CALL DDASSL (RES, neq, t, y, yprime, tout, info, rtol, atol, idid, rwork, lrw, iwork, liw, rpar, ipar, JAC)

If DDASSL fails, the scalar variable **idid** will show the specific error message. The documentation in DDASSL gives a detailed explanation of all the error messages.

The user subroutines RES111 and JAC222 must be declared externally. Even though there is an option in DDASSL which could allow the user to specify the JAC111, it is not recommended to let DDASSL write JAC111 directly because this matrix is not easy to write and debug. We default it to be approximated by DDASSL with a finite difference scheme. More details about DDASSL are mentioned in the next chapter.

DLINPK and DAUX are auxiliary linear algebra routines when DDASSL is being used.

Finally the algorithm for MOVCOL1 is as follows:

Step 1: Check the input parameters and set the default values.

Step 2: Call MOVCOL1 (main subroutine of MOVCOL).

Step 3: MOVCOL1

Substep1: Define the parameters in the following way:

$$\begin{aligned}\mathbf{y} = (&x_1, u_1(x_1), u_{1x}(x_1), u_2(x_1), u_{2x}(x_1), \ldots, u_k(x_1), u_{kx}(x_1),\\ &x_2, u_1(x_2), u_{1x}(x_2), u_2(x_2), u_{2x}(x_2), \ldots, u_k(x_2), u_{kx}(x_2),\\ &x_3, u_1(x_3), u_{1x}(x_3), u_2(x_3), u_{2x}(x_3), \ldots, x_j, u_k(x_j), u_{kx}(x_j))\end{aligned}$$

where $k = 1, \ldots, npde$ and $j = 1, \ldots, npts$

Substep2: Start computing the initial mesh from a uniform mesh.

Substep3: Compute the initial value of $\mathbf{y}'$. Call DDASSL with a first-order backward differentiation formula and two time integration steps.

Substep4: Compute u, u_x, u_t at x_j where $j = 1, \ldots, npts$.

Substep5: External subroutine RES111. Use the cubic Hermite collocation discretization for the physical PDEs and a three point finite difference discretization for the MMPDEs.

Substep6: Call DDASSL to solve the ODE system.

Chapter 3

DDASSL

In this chapter, we describe DDASSL, the initial value solver which is used in MOVCOL and the backward differentiation formulas (BDF) algorithm which it is based on.

3.1 Theory of DAEs

A DAE is a system of differential-algebraic equations. There is algebraic constraint on the variable such as

$$x' = f(x, y, t) \tag{3.1}$$

$$G(x, y, t) = 0. \tag{3.2}$$

If we differentiate the constraint equation respect to t, we can get

$$x' = f(x, y, t) \tag{3.3}$$

$$G_x(x, y, t)x' + G_y(x, y, t)y' = -G_t(x, y, t). \tag{3.4}$$

If g_y is nonsingular, it means we can generate the continuous function satisfying

$$y' = G(t, y(t)), \tag{3.5}$$

and then we call this an index 1 DAE. Otherwise, with some algebraic manipulations and coordinate changes we can rewrite (3.4) to (3.2) and differentiate with respect to t again. The index of the DAE is the minimum number of times that the system needs to be differentiated to get an implicit ODE.

Recall that in the previous chapter, when the method of lines is used to discretize the MMPDE and the physical PDE, the DAE system we obtain is of index 0.

The Backward Differentiation Formula (BDF) [8] is one of the main numerical methods for solving DAEs.

3.2 Backward Differentiation Formula

The simplest first order BDF method is the implicit Euler method, which consists of replacing the derivative in $F(t, y_n, y_n') = 0$ by a backward difference

$$F(t_n, y_n, \frac{y_n - y_{n-1}}{h}) = 0 \tag{3.6}$$

where $h = t_n - t_{n-1}$.

The $k-$th order (constant step-size) BDF consists of replacing y' by the derivative of the polynomial which interpolates the computed solution at $k+1$ times $t_n, t_{n-1}, \ldots, t_{n-k}$, evaluated at t_n. It yields

$$F(t_n, y_n, \frac{\rho y_n}{h}) = 0 \tag{3.7}$$

where $\rho y_n = \sum_{i=0}^{k} \alpha_i y_{n-i}$ and $i = 0, 1, \ldots, k$ are the coefficients of the BDF method. The resulting system of nonlinear equations for y_n at each time step is usually solved by Newton's method. It has been proved in [4] that this method is stable for ODEs when $k < 7$. We are going to show more computational details about the fixed leading coefficients BDF methods which are used by DDASSL in the next section. This fixed leading coefficients formula is a compromise between the fixed coefficient and variable coefficient approaches, offering a stable and efficient computation.

3.3 Basic Algorithm for DDASSL

DDASSL is a Fortran code which is designed by L.Petzold [21] for solving DAE problems of index less than or equal to 1,

$$F(t, y, y') = 0 \tag{3.8}$$

$$y(t_0) = y_0 \tag{3.9}$$

$$y'(t_0) = y_0' \tag{3.10}$$

where F, y, and y' are N-dimensional vectors. In this section, we give a detailed description of the computational algorithm used by DDASSL.

At first it uses a variable order and variable step-size, but a fixed leading coefficient BDF method to approximate the derivatives; then Newton's method is used to solve the resulting nonlinear system at each time step. LINPACK is called to deal with the linear systems and the linear least square problems.

This code implements the backward differentiation formulas from order one through five to solve an implicit differential equation system for y and y'.

We assume that we have the approximations y_{n-i} to the exact solution $y(t_{n-i})$ for $i = 0, 1, \ldots, k$ where k is the order of the BDF we currently use, and we plan to find the approximation of the solution at the time t_{n+1}.
A *predictor polynomial* $\omega^P_{n+1}(t_{n-i})$ is defined to serve as an initial guess for y_{n-i}. It interpolates y_{n-i} at the last $k+1$ time steps, so that

$$\omega^P_{n+1}(t_{n-i}) = y_{n-i}, \quad i = 0, 1, \ldots, k. \tag{3.11}$$

The predicted values for y and y' at t_{n+1} are obtained by evaluating $\omega^P_{n+1}(t)$ and $\omega'^P_{n+1}(t)$ at t_{n+1},

$$y^{(0)}_{n+1} = \omega^P_{n+1}(t_{n+1}), \tag{3.12}$$

$$y'^{(0)}_{n+1} = \omega'^P_{n+1}(t_{n+1}). \tag{3.13}$$

The fixed leading coefficient form of the kth order BDF method is used to develop the *corrector formula*,

$$\omega^C_{n+1}(t_{n+1}) = y_{n+1}, \tag{3.14}$$

$$\omega^C_{n+1}(t_{n+1} - ih_{n+1}) = \omega^P_{n+1},(t_{n+1} - ih_{n+1}), \quad 1 \leq i \leq k \tag{3.15}$$

$$F(t_{n+1}, \omega^C_{n+1}(t_{n+1}), \omega'^C_{n+1}(t_{n+1})) = 0. \tag{3.16}$$

The solution to the corrector formula is the vector y_{n+1} such that the *corrector polynomial* $\omega^C_{n+1}(t)$ and its derivative satisfy the DAE at t_{n+1}, and the *corrector polynomial* interpolates the *predictor polynomial* at k equally spaced points behind t_{n+1}.

The value of the predictor $y^{(0)}_{n+1}$, $y'^{(0)}_{n+1}$ and the corrector y_{n+1} at t_{n+1} are defined in terms of polynomials which interpolate the solution at previous time steps. Thus the predictor polynomial is

$$\begin{aligned}\omega^P_{n+1}(t) = y_n + (t - t_n)[y_n, y_{n-1}] + (t - t_n)(t - t_{n-1})[y_n, y_{n-1}, y_{n-2}] + \ldots + \\ (t - t_n)(t - t_{n-1})(t - t_{n-k+1}) \cdots [y_n, y_{n-1}, y_{n-2}, \ldots, y_{n-k}]\end{aligned} \tag{3.17}$$

where

$$[y_0] = y_n, \tag{3.18}$$

$$[y_n, y_{n-1}, \ldots, y_{n-k}] = \frac{[y_n, y_{n-1}, \ldots, y_{n-k+1}] - [y_{n-1}, y_{n-2}, \ldots, y_{n-k}]}{t_n - t_{n-k}}. \tag{3.19}$$

Evaluating ω_{n+1}^P at t_{n+1}, we can obtain

$$y_{n+1}^{(0)} = \sum_{i=1}^{k+1} \phi_i^*(n) \tag{3.20}$$

$$y_{n+1}^{\prime(0)} = \sum_{i=1}^{k+1} \gamma_i(n+1)\phi_i^*(n) \tag{3.21}$$

where

$$\phi_i^*(n) = \beta_i(n+1)\phi_i(n), \quad i \geq 1 \tag{3.22}$$

$$\gamma_1(n+1) = 0 \tag{3.23}$$

$$\gamma_i(n+1) = \gamma_{i-1}(n+1) + \alpha_{i-1}(n+1)/h_{n+1}, \quad i > 1 \tag{3.24}$$

$$\beta_1(n+1) = 1 \tag{3.25}$$

$$\beta_i(n+1) = \frac{\phi_1(n+1)\phi_2(n+1)\ldots\phi_{i-1}(n+1)}{\phi_1(n)\phi_2(n)\ldots\phi_{i-1}(n)}, \quad i \geq 1 \tag{3.26}$$

$$\phi_1(n) = 1 \tag{3.27}$$

$$\phi_i(n) = \phi_1(n)\phi_2(n)\ldots\phi_{i-1}(n)[y_n, y_{n-1}, \ldots, y_{n-i+1}] \quad i \geq 1. \tag{3.28}$$

The corrector formula and the predictor polynomials satisfy the relationship

$$\omega_{n+1}^C(t) - \omega_{n+1}^P(t) = b(t)(y_{n+1} - y_{n+1}^{(0)}), \tag{3.29}$$

where

$$b(t_{n+1} - ih_{n+1}) = 0, \quad i = 1, 2, \ldots, k. \tag{3.30}$$

$$b(t_{n+1}) = 1 \tag{3.31}$$

Differentiating (3.29) and evaluating at t_{n+1} gives

$$\alpha_s(y_{n+1} - y_{n+1}^{(0)}) + h_{n+1}(y_{n+1}' - y_{n+1}^{\prime(0)}) = 0, \tag{3.32}$$

where the leading fixed coefficients α_s are

$$\alpha_s = -\sum_{j=1}^{k} \frac{1}{j}. \tag{3.33}$$

In order to solve for y'_{n+1}, the corrector iteration must solve

$$F(t_{n+1}, y_{n+1}, y'^{(0)}_{n+1} - \frac{\alpha_s}{h_{n+1}}(y_{n+1} - y^{(0)}_{n+1})) = 0. \tag{3.34}$$

The above equation must be solved for y_{n+1} at each time step. To simplify the notation, we rewrite it as

$$F(t, y, \alpha y + \beta) = 0 \tag{3.35}$$

where $\alpha = -\frac{\alpha_s}{h_{n+1}}$ and $\beta = y'^{(0)}_{n+1} - \alpha y^{(0)}_{n+1}$.
In (3.34), all variables are evaluated at t_{n+1}, α is a constant which changes whenever the step-size or order changes, and β is a vector which remains constant while we are solving the corrector equation. We use the modified Newton iteration to solve the corrector equation (3.33)

$$y^{(m+1)} = y^{(m)} - cG^{-1}F(t, y^{(m)}, \alpha y^{(m)} + \beta), \tag{3.36}$$

where $y^{(0)}_{n+1}$ is $W^P_{t_{n+1}}$ and G is the iteration matrix

$$G = \alpha\frac{\partial F}{\partial y'} + \frac{\partial F}{\partial y}. \tag{3.37}$$

The matrix G is factorized into a product of an upper and lower triangular matrix, $G = LU$. So (3.35) is then solved by

$$Ls^{(m)} = r^{(m)}$$

$$U\delta^{(m)} = s^{(m)}, \tag{3.38}$$

where $\delta^{(m)} = y^{(m+1)} - y^{(m)}$ and $r^{(m)} = -cF(t, y^{(m)}, \alpha y^{(m)} + \beta)$. In DDASSL, the matrix G may be dense or have a banded structure. The factorization of G and the solution of the system in (3.37) are performed by routines in the LINPACK software package.

Normally, especially for large systems, the work of computing and factoring G dominates the cost of the integration. Often the matrices $\frac{\partial F}{\partial y'}$ and $\frac{\partial F}{\partial y}$ change whenever the step-size or order of the method being used changes. The constant α depends on the choice of order or step-size. Whenever either the derivative matrix $\frac{\partial F}{\partial y'}$ or $\frac{\partial F}{\partial y}$, or α changes, we have to recalculate the iteration matrix G. Otherwise, we can fix the iteration matrix as the one in the previous step. We define $\widehat{\alpha}$ as the current constant related to the order and step-size, so the iteration matrix is

$$\widehat{G} = \widehat{\alpha}\frac{\partial F}{\partial y'} + \frac{\partial F}{\partial y'}. \tag{3.39}$$

If $\widehat{G}$ is close enough to G, then the algorithm will converge successfully. If DDASSL fails to converge after four iterations of (3.35), then it needs to rebuild a new iteration matrix G. In DDASSL, the matrix G is computed by finite difference methods. It is also worth mentioning that DDASSL can achieve the same order of convergence for this class of DAEs as it does for ODEs.

3.4 Limitations

In our experience, the majority of DAEs problems whose index are 0 or 1 can be solved successfully with DDASSL. However, we also notice DDASSL has limitations, some of which we summarize below.

3.4.1 Higher Index problems

Sometimes the failure of DDASSL in special situations could be due to the index problem. In particular, the error estimates used in DDASSL may fail to converge for higher index problems. As a response to a large integration error estimate, the code repeats reducing the step-size until the iteration matrix becomes ill-conditioned. For small enough step sizes, this condition problem causes the Newton iteration to not converge, and the code eventually fails due to multiple convergence test failures. It also deserves mentioning that the failures will not necessarily occur on the first step because sometimes DDASSL can start solving a smooth higher index system and fail after a step-size or order change or rapid change in the solution.

In this thesis, we mainly focus on MOVCOL combined with initial value solvers for solving 1-D in space second-order problems. However, we notice that people are paying increasingly more attention to solving fourth- or sixth-order problems with moving mesh software. The common way of solving these problems is converting it into a system of second-order PDEs.

We consider the sixth-order nonlinear diffusion equation

$$u_t = -\frac{\partial}{\partial x}(u^n \frac{\partial^5 u}{\partial x^5}) \tag{3.40}$$

where $u \geq 0$ is the thickness of a fluid film beneath an elastic plate and $p = \frac{\partial^5 u}{\partial x^5}$ is the pressure within the film [6].

We convert this equation to

$$u_t = -(u^n(v_2)_x)_x \tag{3.41}$$

$$v_1 = u_{xx} \tag{3.42}$$

$$v_2 = v_{1xx} \tag{3.43}$$

where (3.41) and (3.42) are the algebraic constraints of the DAE.

This is a DAE system of index 2. Since DDASSL has limitation for solving such problems, we may have to use other software to replace it if we want to solve higher order problem in this method.

3.4.2 Inconsistent Initial Values

Based on the basic algorithm we described before, DDASSL needs a consistent set of initial values $F(t_0, y^{(0)}, y'^{(0)}) = 0$. But it is possible that in some cases, we may not know both initial values for $y^{(0)}$ and $y'^{(0)}$, and thus DDASSL may fail in the first step. For some cases if we only know $y^{(0)}$, DDASSL has an algorithm [4] to compute $y'^{(0)}$ automatically. For MOVCOL, only $y^{(0)}$ is given, so we always need to start looking for $y'^{(0)}$. Frequently either the successful convergence test will fail or the Newton iteration may fail due to the poor initial estimation. Even initial guesses which are only slightly inconsistent could cause DDASSL to fail to complete the first step.

In this general code, there is another option in DDASSL to compute the starting guess for the initial value $y'^{(0)}$ if we know $y^{(0)}$. In this case, DDASSL takes a small implicit Euler step for its first step, and uses a damped Newton iteration to solve the nonlinear system. The error estimate for this step is different from the estimate which DDASSL usually uses because the initial derivatives are not available for use in an error estimate.

In the basic algorithm of MOVCOL, DDASSL does this work in the substep3 as we mentioned in section (2.3). We fix the first order of BDF method and a small time step-size with DDASSL in this substep. Otherwise DDASSL will use a variable order and variable step-size approach. It may bring more accuracy for the guess of the initial value, but cost relatively more computation time. We define $\widehat{y'^{(0)}} = 0$ first, then use the above process to get the initial guess $y'^{(0)}$.

Due to the limitations of DDASSL, we discuss the possibilities of using other ODE solvers to replace DDASSL in the next chapter.

Chapter 4

Alternative Solvers

There are a variety of ODE solvers. When we consider which kind of ODE solver should be used to replace DDASSL, we have to keep in mind which kind of ODE system is investigated. There are several reasons to consider (3.8) rather than trying to rewrite them as an explicit ODE system, so in this thesis, we will pay more attention to the potential software which solves (3.8).

As general-purpose software for solving second order partial differential equations, MOVCOL generates the DAE system $F(t, y, y') = 0$ automatically for any second order PDEs provided by the user. It is very likely that for a specific problem, one can rewrite the resulting DAE system into an explicit form of $B(t, y)y' = F(t, y)$; however, this would require the user to have a very good sense of the way MOVCOL discretizes for the physical PDEs and the MMPDE. Even if this is the case, for the sake of numerical efficiency, the change to an explicit form can destroy sparsity and prevent the exploitation of the system structure [4]. Most of all, this has to be done case by case, which conflicts with the concept of designing general-purpose software.

Recently, people have paid more attention to using IRK (Implicit Runge-Kutta) methods to solve DAE problems. Especially for the case of using moving mesh methods to solve PDEs, each interpolation of a variable onto a new mesh generates a discontinuity of that variable in time. Since IRK is a one-step method, it has a potential advantage compared to a multi-step method such as BDF in approximating rapidly changing functions. Due to their one step nature, IRK methods are potentially more efficient for these problems than multi-step methods. Even though both methods have to restart at the discontinuity point, the IRK method can restart at a higher order rather than at a lower order like multi-step methods.

4.1 Implicit Runge Kutta Methods

An S stage IRK method applied to the DAE (3.1)and (3.2) is

$$F(t_{n-1} + c_i h, y_{n-1} + h\sum_{j=1}^{M} a_{ij} Y_j', Y_i') = 0, \quad i = 1, 2, \ldots, M, \tag{4.1}$$

$$y_n = y_{n-1} + h\sum_{i=1}^{M} b_i Y_i'. \tag{4.2}$$

where $h = t_n - t_{n-1}$. It yields

$$y_n = y_{n-1} + h\varphi(t_{n-1}, y_{n-1}, h). \tag{4.3}$$

4.2 PSIDE

4.2.1 Introduction

PSIDE's full name is Parallel Software for IDEs (Implicit Differential Equations). It is a Fortran code for solving the implicit differential equations

$$g(t, y, y') = 0 \tag{4.4}$$

$$y(t_0) = y_0, \quad y'(t_0) = y_0' \tag{4.5}$$

on shared memory computers.

The algorithm for PSIDE is based on a four-stage Radau IIA method, which is one of the implicit Runge-Kutta methods. The linear systems are solved by a modified Newton process, in which every Newton iterate itself is computed by means of the Parallel Iterative Linear Solver for Runge-Kutta (PILSRK) proposed in [14]. A Fortran code CACM423 is also required as a substitute for the linear algebra routine LAPACK.

Implementing the RADAUII method requires high computational costs. PSIDE is designed for using 4 processors to compute those 4 stages in parallel to increase the speed of computation. Solving (4.4) with the four stage RADAUII methods means to solve for $\dot{Y}$ from the nonlinear system

$$G(X \otimes y_n + h(A \otimes I)\dot{Y}, \dot{Y}) = 0, \tag{4.6}$$

where

$$\dot{Y} = (\dot{Y}_1^T, \dot{Y}_2^T, \dot{Y}_3^T, \dot{Y}_4^T) = 0,$$

h is the step-size and the matrix A

$$\begin{pmatrix} 0.1129994793231 & -0.0403092207235 & 0.0258023774203 & -0.0099046765072 \\ 0.2343839957473 & 0.2068925739354 & -0.0478571280485 & 0.0160474228065 \\ 0.2166817846232 & 0.4061232638674 & 0.1890365181700 & -0.0241821048998 \\ 0.2204622111767 & 0.3881934688432 & 0.3288443199800 & 0.06250000000000 \end{pmatrix} \tag{4.7}$$

is the 4×4 RADAUII matrix. $X = (1,1,1,1)^T$ and I is an identity matrix.

4.2.2 Getting Start with PSIDE

There are three important subroutines we need to mention for PSIDE. The first one is GEVAL, which defines the IDE problem

$$G(t, y, y') = 0. \tag{4.8}$$

We must declare GEVAL as the external statement in our own program. IERR is an integer flag which is always equal to zero on input. Subroutine GEVAL should set $IERR = -1$ if GEVAL cannot be evaluated for the current values of y and dy. PSIDE will then try to prevent IERR = -1 by using a smaller step-size.

To solve the IDE, it is necessary to use the partial derivatives $J = \partial G/\partial y$. The solution will be more reliable if we provide J via the subroutine JEVAL.

The third subroutine is MNUM. To solve the IDE it is also necessary to use the partial derivatives $M = \partial G/\partial y'$. The solution will be more reliable if we provide M via MEVAL.

However, sometimes we can use dummy routines to take the place of those external subroutines. If we define MNUM = TRUE and JNUM = TRUE, PSIDE will approximate M and J by numerical differentiation automatically.

4.2.3 Limitations

Even though PSIDE is powerful software in scientific computation, it still has some limitations in practical applications. We summarize two of them here.

PSIDE is designed to run on a four processor workstation. Since the parallel computation can do the four stage computation at the same time, it will efficiently reduce the cost of the implicit Runge-Kutta method and also keep the good properties for this method such as relatively highly computational accuracy. However, not all computers have four processors. PSIDE can still work in a one processor computer and maintain the high computational accuracy, but cost a large amount of computation time.

We use LAPACK to solve the linear algebraic system for PSIDE. For the implementation of LAPACK, we need a machine with LAPACK and a machine with optimized BLAS. However, not all systems have both available. Downloading and installing both packages can be a complicated process. There is a Fortran routine called CACM423 which could replace a machine tuned LAPACK and machine optimized BLAS [24].

4.3 Other Possibilities

We have mentioned in the previous chapter that we can discretize the MMPDE and the physical PDE in the form

$$F(t, y, y') = 0 \tag{4.9}$$

using the method of lines. Even though we have been discussing a lot about the advantages of working directly with (4.10), we examine the possibilities of working with

$$B(t, y)y' = F(t, y) \tag{4.10}$$

In the appendix, we rewrite the system into an explicit form for Burgers' equation to facilitate a better understanding of the underlying collocation method for MOVCOL and the choice of different ODE/DAE solvers. Consider

$$u_t = \epsilon u_{xx} - \frac{1}{2}(u^2)_x \quad 0 < x < 1, \quad t > 0, \quad \epsilon = 10^{-4}$$
$$u(0,t) = u(1,t) = 0 \quad t > 0$$
$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x), \quad 0 \le x \le 1.$$

For simplicity, we give the reformulation for a three points mesh case

$$0 = x_1 < x_2 < x_3 = 1,$$

where the boundary condition and initial condition are

$$u(0,0) = u(1,0) = 0, \quad t > 0.$$

For the sake of simplicity, we discretize the equation on a moving mesh with only 3 mesh points. Details for discretizing MMPDE4 and MMPDE6 are also provided. As for the smooth version of MMPDE4, we show that it is not linearly implicit and thus can not be rewritten into the explicit form. In this case, one has to choose a DAE solver for $F(t, y, y') = 0$ over a DAE solver for $By' = F(t, y)$.

This theoretical work for Burgers' equation provides a useful guidance for writing the form of (4.1) for other problems. However, we notice that this task requires the user to have a good understanding of both MOVCOL and ODE solvers.

Besides the direct use of IVP solvers based on Runge-Kutta methods instead of using DDASSL, another alternative is that one may use implicit DAE solvers to generate consistent starting values for higher order BDF methods solvers, and then use solvers based on BDF methods to take care of other parts. Thus, we in principle exploit the advantages of both methods.

Chapter 5

Numerical Experiments

In this chapter, we consider two versions of MOVCOL, one with DDASSL and the other with PSIDE for several numerical experiments: a simple problem, a heat conduction problem, a scalar combustion model and the Gray-Scott problem. We choose these problems as our test examples because they have been used extensively in the literature and show qualitatively different solution behavior. We use the following notations for the error measurements:

$$e_i = e(x_i) = |u^e(x_i) - u^c(x_i)|,$$

$$||e||_\infty = \max(e_1, \ldots, e_n),$$

$$||e_R|| = |\frac{u^e(x_i) - u^c(x_i)}{u^e(x_i)}|,$$

$$||e||_2 = (\sum_{i=1}^{n} {e_i}^2 * (x_i - x_{i-1}))^{\frac{1}{2}}.$$

where n is the number of mesh points when we compute the solution, u^e is the exact or reference solution to the underlying problem and u^c is the computed solution. All computations presented in this thesis used a HITACHI-PC-UC5910A labtop in double precision.

Throughout, we use the arc-length monitor function

$$M(x,t) = \sqrt{1 + (\frac{\partial u}{\partial x})^2} \tag{5.1}$$

for all the problems. In order to have a clear understanding of the computation efficiency for both versions, we first consider the following two problems for which an analytic solution is available.

5.1 Simple Problem

We start with a relatively simple problem

$$u_t = u_{xx} \tag{5.2}$$

$$u(x,t) = \exp(-t)\cos(x).$$

The boundary conditions and initial condition are derived from the exact solution. In the following tables, P and D stand for the IVP solver PSIDE and DDASSL; *atol* and *rtol* stand for the absolute error tolerance and the relative error tolerance used in those IVP solvers; *npts* stands for the number of the mesh points when we compute the solution and the length of time integration is from 0 to 0.7.

Table 5.1: Computation time and error for a simple problem when $atol = rtol = 10^{-3}$, $npts = 41$.

	PSIDE		DDASSL	
	MMPDE4	MMPDE6	MMPDE4	MMPDE6
$\|\|e\|\|_\infty$	4.5216×10^{-6}	2.741×10^{-6}	6.338×10^{-4}	6.298×10^{-4}
$\|\|e_R\|\|_\infty$	0.2304	0.0046	0.1488	0.1490
TIME (seconds)	3.06	3.26	0.351	0.201

Here are some plots for the exact solutions, numerical solutions and errors.

A summary of the conclusions we get from the following tables is:

1. Table 5.1 shows that for each individual MOVCOL, MMPDE4 and MMPDE6 obtain similar results in terms of both accuracy and computational cost.
2. Deviations may be caused by inaccuracy in both the time integration and the discrete approximation to the problem.
3. Table 5.2 shows for this problem, in most of the cases, PSIDE with MOVCOL can get higher accuracy than the requirement, but it also will take much longer time.
4. PSIDE and DDASSL can obtain the same order of accuracy for roughly the same computational time if fewer mesh points are used for PSIDE than DDASSL.

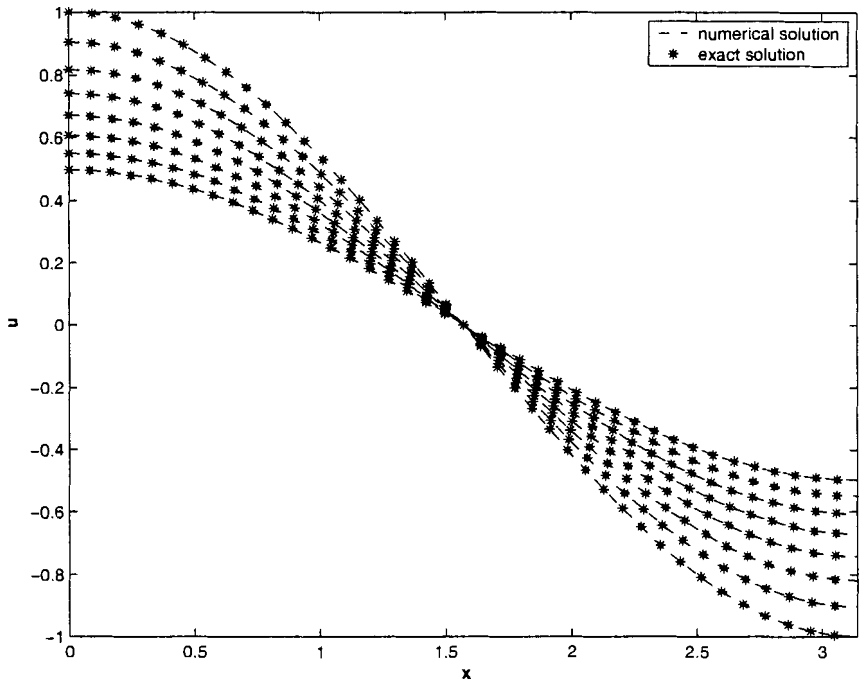

Figure 5.1: Numerical solution and exact solution for the simple problem by MOVCOL with PSIDE at $npts = 41$, $atol = rtol = 10^{-3}$ and with MMPDE6.

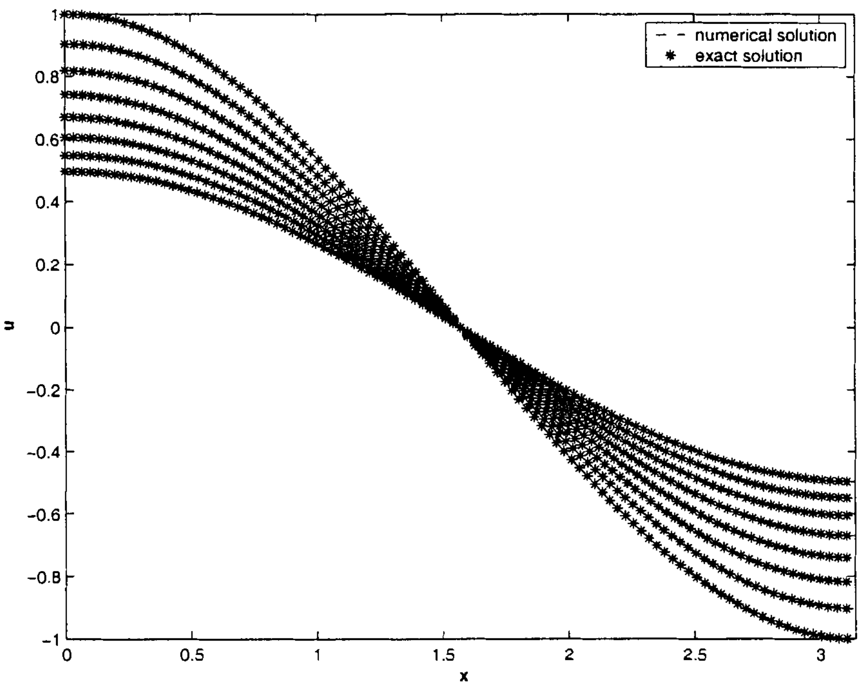

Figure 5.2: Numerical solution and exact solution for the simple problem by MOVCOL with PSIDE at $npts = 101$, $atol = rtol = 10^{-3}$ and with MMPDE6.

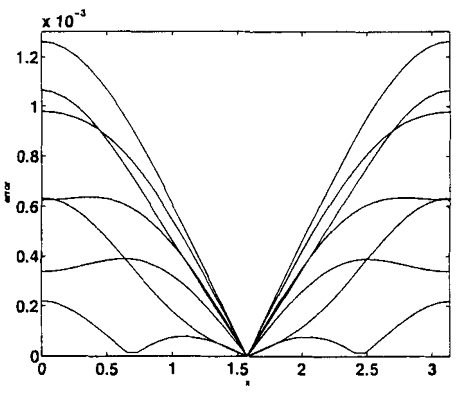

Figure 5.3: Computation error for the simple problem for MOVCOL with DDASSL at *atol* = *rtol* = 10^{-3}, *npts* = 41 and with MMPDE6.

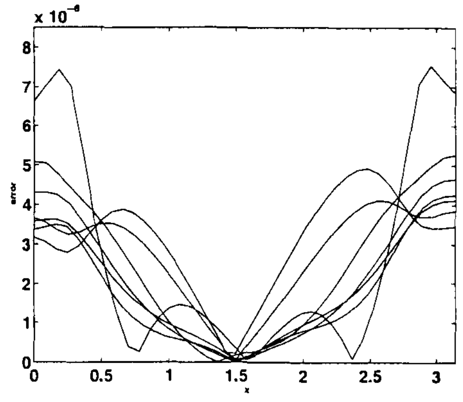

Figure 5.4: Computation error for the simple problem for MOVCOL with PSIDE at *atol* = *rtol* = 10^{-3}, *npts* = 41 and with MMPDE6.

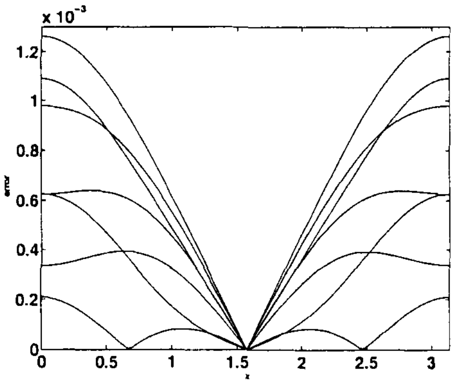

Figure 5.5: Computation error for the simple problem for MOVCOL with DDASSL at *atol* = *rtol* = 10^{-3}, *npts* = 101 and with MMPDE6.

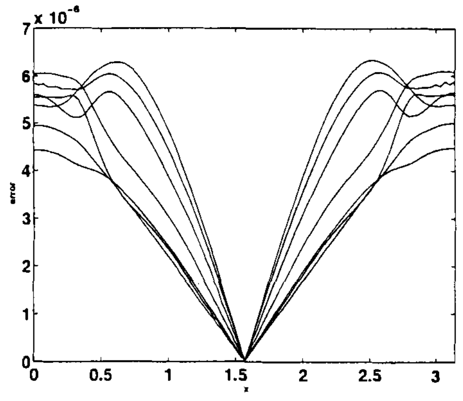

Figure 5.6: Computation error for the simple problem for MOVCOL with PSIDE at *atol* = *rtol* = 10^{-3}, *npts* = 101 and with MMPDE6.

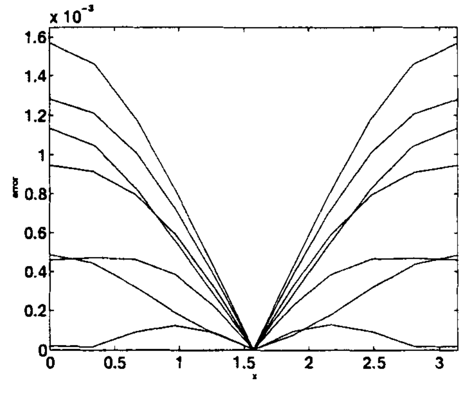

Figure 5.7: Computation error for the simple problem for MOVCOL with DDASSL at $atol = rtol = 10^{-3}$, $npts = 11$ and with MMPDE6.

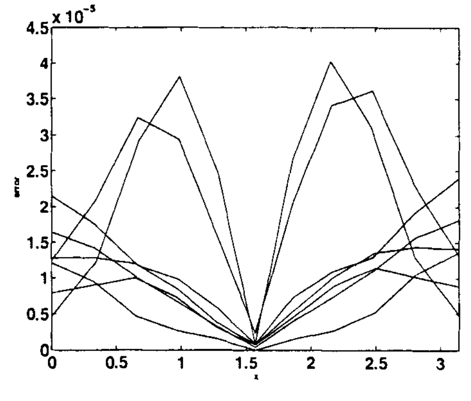

Figure 5.8: Computation error for the simple problem for MOVCOL with PSIDE at $atol = rtol = 10^{-3}$, $npts = 11$ and with MMPDE6.

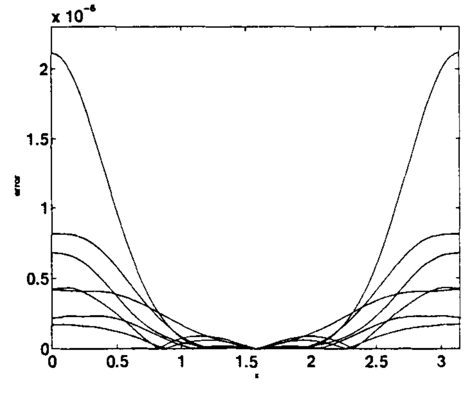

Figure 5.9: Computation error for the simple problem for MOVCOL with DDASSL at $atol = rtol = 10^{-6}$, $npts = 101$ and with MMPDE6.

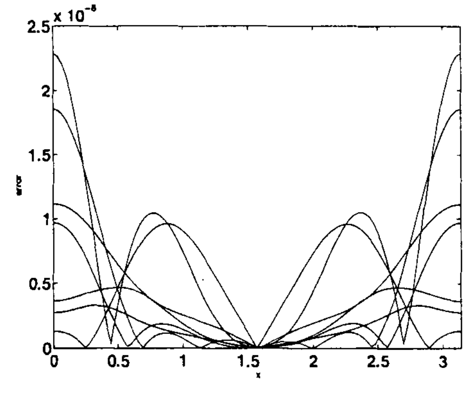

Figure 5.10: Computation error for the simple problem for MOVCOL with PSIDE at $atol = rtol = 10^{-6}$, $npts = 101$ and with MMPDE6.

Table 5.2: Computation time and error for a simple problem using MOVCOL with MMPDE6.

$atol = rtol$	10^{-3}					
npts	101		41		11	
Method	P	D	P	D	P	D
$\|\|e\|\|_\infty$	1.129×10^{-5}	6.177×10^{-3}	2.741×10^{-6}	6.298×10^{-4}	6.483×10^{-6}	4.864×10^{-4}
$\|\|e_R\|\|_\infty$	0.0435	0.1652	0.0046	0.1490	0.0090	0.078
Time (seconds)	18.435	0.621	3.26	0.201	0.301	0.05
$atol = rtol$	10^{-6}					
npts	101		41		11	
Method	P	D	P	D	P	D
$\|\|e\|\|_\infty$	9.849×10^{-9}	9.35×10^{-6}	3.832×10^{-9}	1.0203×10^{-6}	3.356×10^{-6}	3.378×10^{-6}
$\|\|e_R\|\|_\infty$	4.846×10^{-4}	0.0053	3.356×10^{-5}	0.0072	8.227×10^{-5}	5.277×10^{-4}
Time (seconds)	23.833	0.841	4.206	0.340	0.321	0.130
$atol = rtol$	10^{-9}					
npts	101		41		11	
Method	P	D	P	D	P	D
$\|\|e\|\|_\infty$	5.584×10^{-10}	5.548×10^{-9}	2.097×10^{-8}	1.451×10^{-8}	3.845×10^{-6}	4.088×10^{-6}
$\|\|e_R\|\|_\infty$	12.866×10^{-8}	4.422×10^{-5}	4.505×10^{-6}	2.013×10^{-6}	2.045×10^{-4}	2.639×10^{-4}
Time (seconds)	36.502	1.833	6.530	0.681	0.461	0.190

5.2 Heat Conduction Problem

The second problem, a heat conduction problem which has been considered by [18], is

$$\frac{\partial u}{\partial t} = \mu u_{xx} + (r_2 + 2r_1^2 \mu u)(1 - u^2) \tag{5.3}$$

$$u(x,t) = \tanh(r_1 x + r_2 t), \quad -3 \le x \le 3. \tag{5.4}$$

The boundary conditions

$$u(-3,t) = \tanh(-3r_1 + r_2 t), \quad u(3,t) = \tanh(3r_1 + r_2 t),$$

and initial conditions

$$u(x,0) = \tanh(r_1 x), \quad -3 \le x \le 3 \tag{5.5}$$

are set from the given exact solution (5.4). In our experiments, we choose the small diffusion term $\mu = 10^{-3}$ and $r_1 = r_2 = 5.0$ and $atol = 10^{-6}$ and $rtol = 10^{-6}$. The wave velocity is $c = \frac{-r_2}{r_1} < 0$. The solution has a steep wave front propagating to the left side, which will reach the left boundary at time $t = 3$. We test the problem with and $atol = rtol = 10^{-6}$ and 21 nodes and 61 nodes, and the moving mesh equation MMPDE6.

From the numerical results,Table 5.3, Table 5.4 and Figures 5.11 - 5.20, we conclude the following:

1. Using the maximum norm, we see the computational error is primarily when the solution changes rapidly.

2. For the heat conduction problem, the solution starts to oscillate at t = 0.7. The oscillation becomes strong after t = 1.0. With the presence of numerical instabilities, DDASSL will have difficulty to converge, and the numerical error goes out of control; however, this oscillation doesn't affect the ODE solver PSIDE dramatically. Compared to MOVCOL with DDASSL, MOVCOL with PSIDE does a better job in suppressing the numerical instability, and uses substantially less time to get comparable results after the strong oscillation point t=1.0. However, there is still numerical insatiability.

3. After using more mesh points for the heat conduction problem, the numerical solutions becomes smoother and there are no obvious oscillations. In this case, PSIDE uses more time than DDASSL, but DDASSL gets a better approximation than PSIDE.

4. When we require more computational accuracy, (i.e., increase the absolute tolerance and relative tolerance from 10^{-3} to 10^{-6}), the computation time for both PSIDE and DDASSL increase correspondingly, but the approximation error in the oscillatory part doesn't change significantly.

5. The heat conduction problem is provided as a numerical experiment which gives problems for moving mesh methods in [18], since the numerical solution has an oscillatory part. We see that under the same conditions as above, MOVCOL with PSIDE reduces this oscillation and improves the computational efficiency.

* once instability occurs, comparing these number will no longer make sense. However, we see that MOVCOL with PSIDE does a better job than MOVCOL with DDASSL in suppressing the instability.

Table 5.3: Computation time and error for the heat conduction problem with MMPDE6 when $npts = 21$.

	$atol = rtol = 10^{-3}$					
	computation time		$\|\|e\|\|_\infty$		$\|\|e\|\|_2$	
	P	D	P	D	P	D
t=0.5	0.821	0.19	0.0664	0.0695	0.0742	0.0727
t=0.7	0.91	0.19	0.1318	0.1372	0.1670	0.2246
t=0.8	0.97	0.32	0.3052	0.5572	0.3526	0.6946
t=0.9	1.17	0.41	0.6493	1.3945	0.8327	2.1813
t=1.0	1.63	0.66	1.0512	6.7274	1.5210	12.2155*
t=1.1	2.74	18.52	1.2738	100.62	2.3294	282.5901*
	$atol = rtol = 10^{-6}$					
	computation time		$\|\|e\|\|_\infty$		$\|\|e\|\|_2$	
	P	D	P	D	P	D
t=0.5	1.262	0.44	0.0705	0.0843	0.0775	0.0892
t=0.7	1.33	0.51	0.1406	0.1552	0.1869	0.2311
t=0.8	1.66	0.55	0.4043	0.5729	0.4474	0.6560
t=0.9	2.10	0.72	0.8402	1.3894	1.1441	1.9923
t=1.0	2.80	0.91	1.7817	4.9629	2.5389	9.0255*
t=1.1	4.37	72.58	4.1556	103.9800	7.3102	234.3755*

Table 5.4: Computation time and error for the heat conduction problem with MMPDE6 when $npts = 61$.

	$atol = rtol = 10^{-3}$					
	computation time		$\|\|e\|\|_\infty$		$\|\|e\|\|_2$	
	P	D	P	D	P	D
t=0.5	11.81	0.56	0.0012	0.0243	0.0019	0.0534
t=0.7	11.49	0.56	0.0019	0.0509	0.0035	0.1376
t=0.8	12.14	0.58	0.0019	0.0569	0.0037	0.1668
t=0.9	12.41	0.66	0.0021	0.0646	0.0040	0.1908
t=1.0	12.99	0.69	0.0024	0.0759	0.0046	0.2195
t=1.1	13.21	0.67	0.0025	0.0856	0.0051	0.2509
	$atol = rtol = 10^{-6}$					
	computation time		$\|\|e\|\|_\infty$		$\|\|e\|\|_2$	
	P	D	P	D	P	D
t=0.5	15.26	0.92	0.0012	0.0018	0.0021	0.0032
t=0.7	15.78	1.02	0.0023	0.0026	0.0044	0.0049
t=0.8	16.39	1.20	0.0026	0.0027	0.0050	0.0053
t=0.9	17.09	1.26	0.0027	0.0029	0.0055	0.0056
t=1.0	17.79	1.60	0.0030	0.0031	0.0060	0.0061
t=1.1	18.39	1.36	0.0033	0.0033	0.0065	0.0066

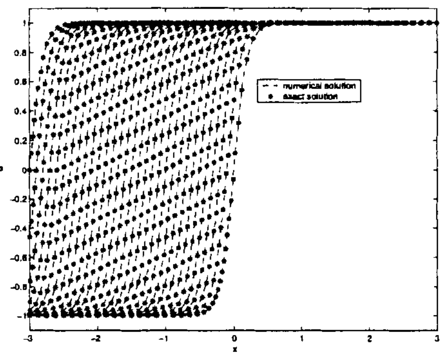

Figure 5.11: Numerical solution and exact solution for the heat conduction problem for MOVCOL with PSIDE from time=0 to time= 3.0, when $atol = rtol = 10^{-6}$ and $npts = 61$

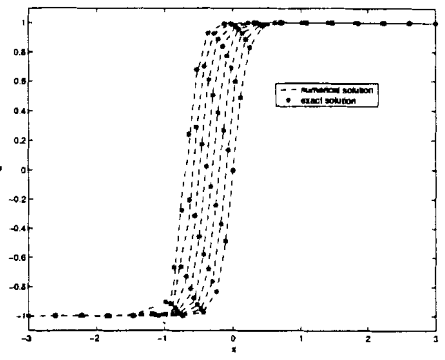

Figure 5.12: Numerical solution and exact solution for the heat conduction problem for MOVCOL with DDASSL from time=0 to time= 0.7, when $atol = rtol = 10^{-6}$ and $npts = 21$

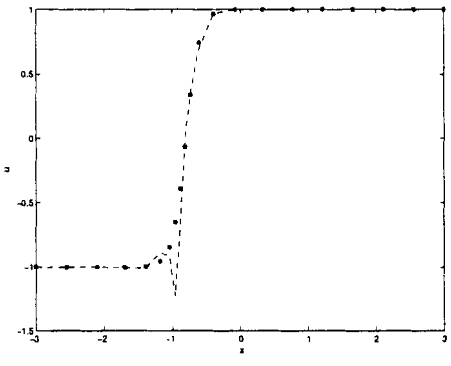

Figure 5.13: Numerical solution and exact solution for the heat conduction problem for MOVCOL with DDASSL at time=0.8, when $atol = rtol = 10^{-6}$ and $npts = 21$

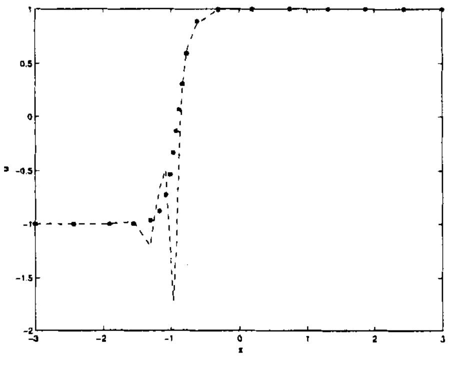

Figure 5.14: Numerical solution and exact solution for the heat conduction problem for MOVCOL with DDASSL at time= 0.9, when $atol = rtol = 10^{-6}$ and $npts = 21$

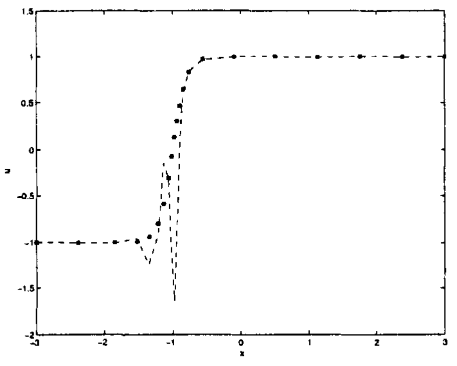

Figure 5.15: Numerical solution and exact solution for the heat conduction problem for MOVCOL with DDASSL at time=1.0, when $atol = rtol = 10^{-6}$ and $npts = 21$

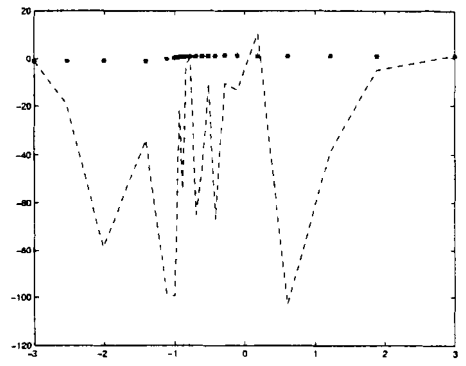

Figure 5.16: Numerical solution and exact solution for the heat conduction problem for MOVCOL with DDASSL at time= 1.1, when $atol = rtol = 10^{-6}$ and $npts = 21$

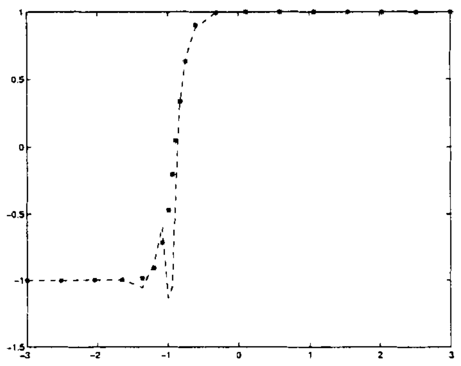

Figure 5.17: Numerical solution and exact solution for the heat conduction problem for MOVCOL with PSIDE at time=0.8, when $atol = rtol = 10^{-6}$ and $npts = 21$

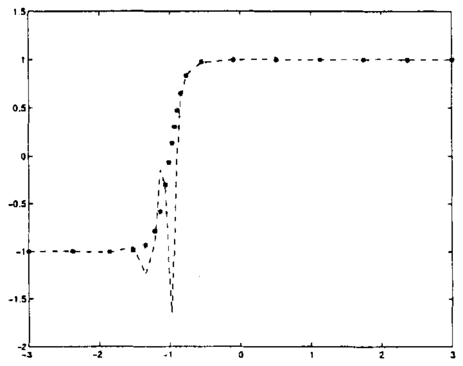

Figure 5.18: Numerical solution and exact solution for the heat conduction problem for MOVCOL with PSIDE at time= 0.9, when $atol = rtol = 10^{-6}$ and $npts = 21$

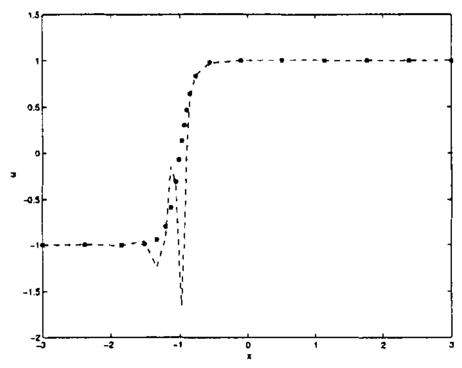

Figure 5.19: Numerical solution and exact solution for the heat conduction problem for MOVCOL with PSIDE at time=1.0, when *atol* = *rtol* = 10^{-6} and *npts* = 21

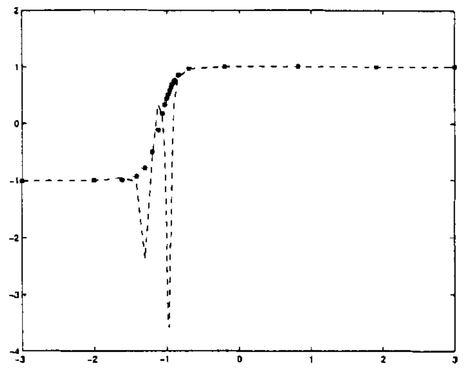

Figure 5.20: Numerical solution and exact solution for the heat conduction problem for MOVCOL with PSIDE at time= 1.1, when *atol* = *rtol* = 10^{-6} and *npts* = 21

5.3 Scalar Combustion Model

The third problem we present is a reaction-diffusion equation which models a problem from combustion theory. This problem is described in [7], [22] as a model of a single-step reaction diffusion and reads

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + D(1 + a - u)e^{(-d/u)}, \quad 0 < x < 1, \quad 0 < t, \tag{5.6}$$

$$\frac{\partial u}{\partial t}(0,t) = 0 \quad u(1,t) = 1, \quad 0 < t, \tag{5.7}$$

$$u(x,0) = 1, \quad 0 \leq x \leq 1, \tag{5.8}$$

where $D = Re^d/(ad)$, and $a = 1$, $d = 20$, $R = 5$. We use *atol* = *rtol* = 10^{-6} and the number of points=41. The solution represents the temperature of a reaction in a chemical system. For small times the temperature gradually increases from unity with a "hot spot" at $x = 0$.

Figures 5.21 and 5.22 show that the numerical solution reaches a steady state at $t = 2.9$. Table 5.5 shows that with the same order of error tolerance and the same number of mesh points, MOVCOL with PSIDE works way more slowly than MOVCOL with DDASSL.

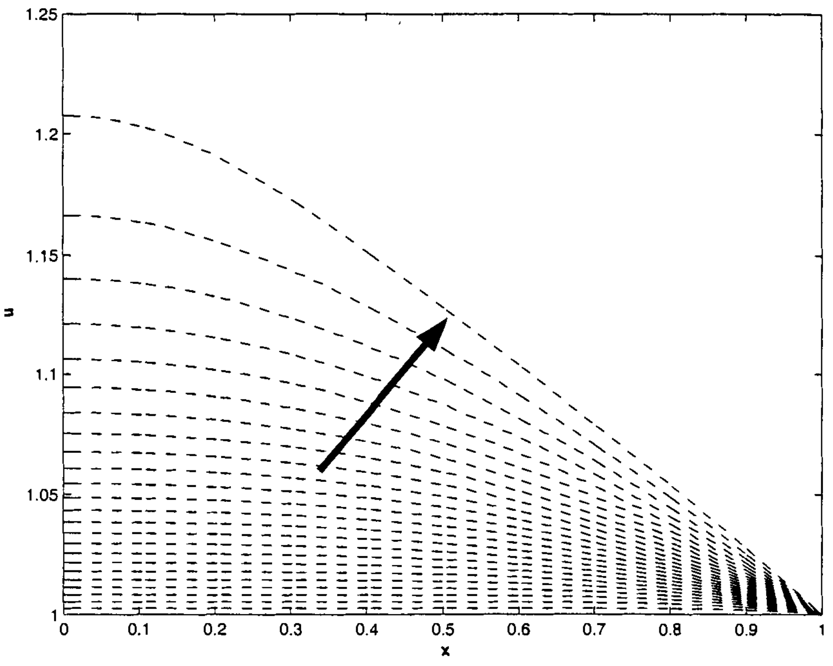

Figure 5.21: Numerical solution for the Scalar Combustion Model for MOVCOL with PSIDE from time 0 to time 2.5. The arrow shows the direction in which time increases.

5.4 Gray-Scott Problem

As a final example, we consider a reaction diffusion system for a chemical species. The Gray-Scott system is one such classical model [26]. The PDE models a chemical reaction in the following way:

$$U + 2V \rightarrow 3V,$$

$$V \rightarrow P,$$

where U, V and P are chemical species. The details of this system are given by

$$\frac{\partial u}{\partial t} = D_u u_{xx} - uv^2 + F(1-u) \tag{5.9}$$

$$\frac{\partial v}{\partial t} = D_v v_{xx} + uv^2 - (F+K)v \tag{5.10}$$

where D_u and D_u represent the diffusion rates, K is the rate of conversion of V to P, and F is the rate of the process that feeds U and drains U, V and P. In this system, U and V react with each other and produce some spikes.

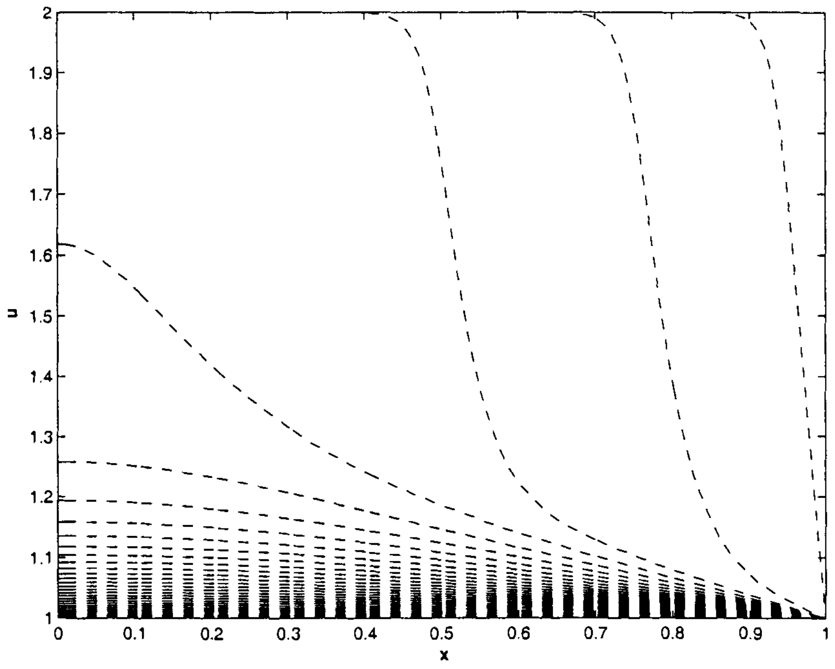

Figure 5.22: Numerical solution for the Scalar Combustion Model for MOVCOL with PSIDE from time 0 to time 2.9

In our experiments, we consider the following choices for the parameters: $D_u = 10^{-4}$, $D_v = 10^{-6}$, $F = 0.024$ and $K = 0.06$. The initial conditions are

$$u(x,0) = 1 - 0.5\sin^{100}(\pi x)$$

$$v(x,0) = 0.25\sin^{100}(\pi x).$$

The boundary conditions are Dirichlet boundary conditions on the domain $[0,1]$.

After comparing the mesh trajectory and the solution, from Figure 5.23 to Figure 5.30, it is not hard to see that the moving mesh code does an efficient and reliable job. The mesh distribution is almost uniform when the physical solution is smooth. But once the G-S model produces some spikes, the mesh trajectory will concentrate in the corresponding areas. This verifies the basic idea of the moving mesh method: the mesh concentrates in the areas where the physical solution changes fast.

These two equation systems show the version of MOVCOL with PSIDE can work as well as MOVCOL with DDASSL for reaction diffusion systems where the equations

Table 5.5: The computation time for the scalar combustion problem using MOVCOL with DDASSL and MOVCOL with PSIDE when $MMPDE = 6$.

	npts=21				npts=41			
atol=rtol	10^{-3}		10^{-6}		10^{-3}		10^{-6}	
	P	D	P	D	P	D	P	D
t=2.5	1.729	0.471	2.13	0.781	5.918	0.751	8.342	0.941
t=2.6	1.903	0.671	2.233	0.821	6.229	0.872	9.163	1.012
t=2.7	2.463	0.761	3.395	1.051	10.605	1.161	15.252	1.492
t=2.8	2.784	0.791	3.855	1.092	12.468	1.212	18.305	1.713
t=2.9	3.184	0.841	4.466	0.121	15.241	1.372	21.121	1.952

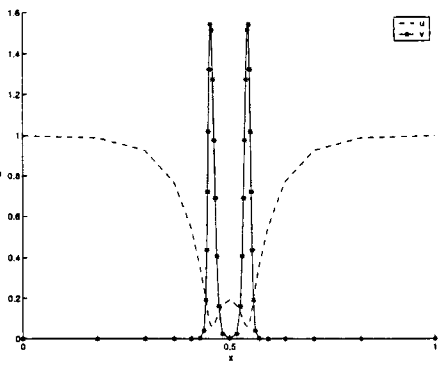

Figure 5.23: Numerical solution for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=200, when $atol = rtol = 10^{-6}$, $npts = 41$ and with MM-PDE6

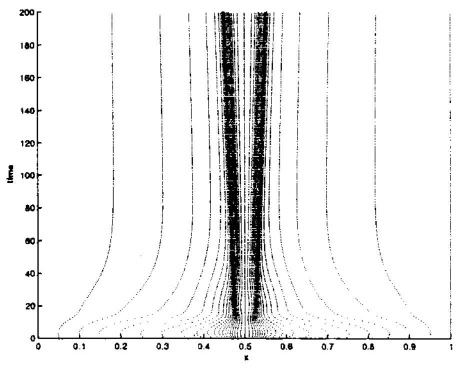

Figure 5.24: Mesh trajectory for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=200, when $atol = rtol = 10^{-6}$, $npts = 41$ and with MM-PDE6

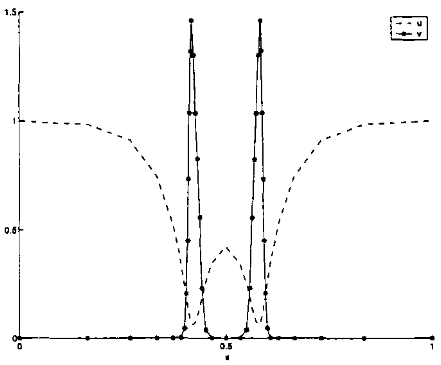

Figure 5.25: Numerical solution for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=500, when *atol* = *rtol* = 10^{-6}, *npts* = 41 and with MMPDE6

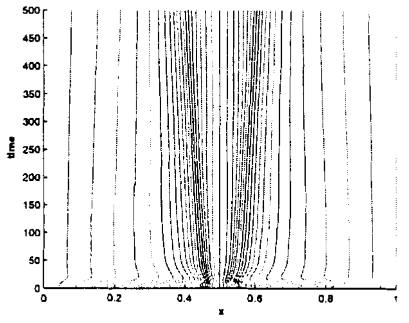

Figure 5.26: Mesh trajectory for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=500, when *atol* = *rtol* = 10^{-6}, *npts* = 41 and with MMPDE6

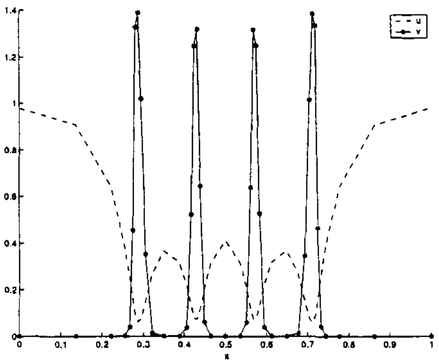

Figure 5.27: Numerical solution for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=1500, when *atol* = *rtol* = 10^{-6}, *npts* = 41 and with MMPDE6

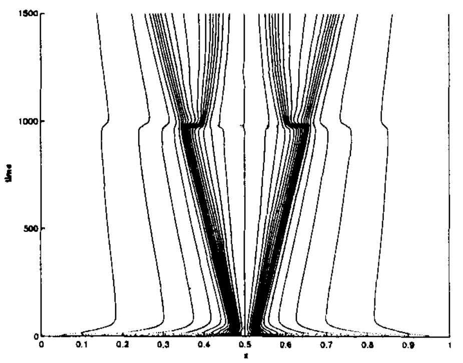

Figure 5.28: Mesh trajectory for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=1500, when *atol* = *rtol* = 10^{-6}, *npts* = 41 and with MMPDE6

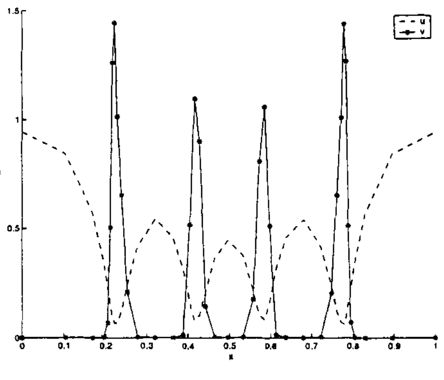

Figure 5.29: Numerical solution for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=2000, when *atol* = *rtol* = 10^{-6}, *npts* = 41 and with MM-PDE6

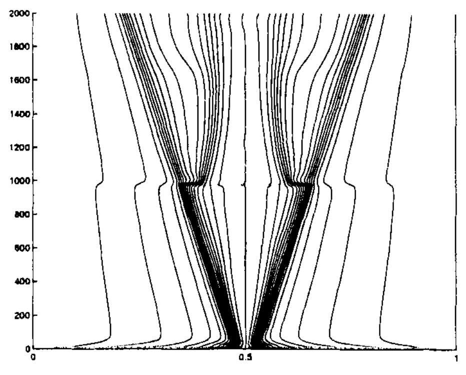

Figure 5.30: Mesh trajectory for the 1D Gray-Scott problem for MOVCOL with PSIDE at time=2000, when *atol* = *rtol* = 10^{-6}, *npts* = 41 and with MM-PDE6

strongly affect each other.

The workstation we use to do these experiments only has one processor, which is a possible reason that the cost of computation for PSIDE is very large here [25] [2].

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we study the role of two different IVP solvers – DDASSL and PSIDE – for solving second-order partial differential equations. The IVP equation system is obtained by using moving mesh software which uses the method of lines as well as a moving collocation scheme.

We start by briefly describing how MOVCOL discretizes second-order partial differential equations of a general form based on moving mesh methods and collocation. Basic algorithms as well as limitations of this software are then discussed. By successfully implementing PSIDE with MOVCOL, which originally works with DDASSL, we are able to compare the two IVP solvers when incorporated into MOVCOL.

Numerical experiments show that for a simple problem with smooth solution profiles, DDASSL works much faster; however PSIDE can achieve a higher order of accuracy. PSIDE and DDASSL can obtain the same order of accuracy for roughly the same computational time if fewer mesh points are used for PSIDE than DDASSL. For different choices of moving mesh equations, e.g., MMPDE4 and MMPDE6, similar results are obtained in terms of accuracy and computational cost.

For the heat conduction problem mentioned in [18] as a problem for which moving mesh methods have difficulty, DDASSL indeed has difficulty in the time integration when the numerical solution has strong spurious oscillations. PSIDE is doing much better than DDASSL but still has strong oscillations. After we increase the number of mesh points, the oscillation in the solution disappear and then both PSIDE and

DDASSL work well. The resulting accuracy and efficiency comparison is then similar to the smooth case.

For the scalar combustion problem for which the moving mesh method works very well [18], both versions indeed work well. However, PSIDE is much slower than DDASSL in this case. The successful application of PSIDE to the Gray-Scott model proves that our current version can also work well for systems which have more than one physical equations.

As we see in those experiments, using a one processor workstation, DDASSL can reach the required accuracy in reasonable time. PSIDE can achieve higher accuracy for the same tolerance but takes a longer time to finish its task. It is our conclusion that while choosing one ODE solver over another in practical can depend on the user's requirement for both computational cost and accuracy, DDASSL is in general very competitive.

6.2 Future Work

While we have successfully implemented PSIDE with MOVCOL and done the comparison with the combination of DDASSL and MOVCOL for several numerical problems, a lot of work could still be done in order to enable us to further investigate the potential of PSIDE.

First, further numerical comparisons of the two versions could be done for the last two problems. We didn't test the computational accuracy of the scalar combustion problem and the Gray Scott problem, which would require computing a reference solution with large N. Further work is also needed for testing PSIDE on problems for which DDASSL has difficulties getting started.

Second, we have showed in this thesis that an DAE system of the following form

$$By' = F(t, y) \tag{6.1}$$

could be obtained for second-order PDEs case by case with MOVCOL. The explicit form of (6.1) for Burgers' equation is reformulated which has only three mesh points for simplicity. This would provide us some guidance on how to expand this specific case to other cases in the future.

Finally, and perhaps most important of all, it would be interesting to run the version of PSIDE with MOVCOL using a machine tuned LAPACK and machine optimized

BLAS and in parallel. As a parallel code, it has been showned that running PSIDE with four processors reduces the computation time dramatically especially when a higher order of accuracy is required [25], [2]. We believe those changes will increase the computational efficiency for this new version and also will be helpful in investigating other properties of PSIDE combined with MOVCOL.

Appendix A

Reformulation of ODE systems by example

Consider Burgers' Equation

$$u_t = \epsilon u_{xx} - \frac{1}{2}(u^2)_x, \quad 0 < x < 1, \quad t > 0, \quad \epsilon = 10^{-4},$$

$$u(0,t) = u(1,t) = 0, \quad t > 0,$$

$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x), \quad 0 \leq x \leq 1.$$

For sake of simplicity, we investigate the case where the mesh has only three points in the physical domain $[0,1]$,

$$0 = x_1 < x_2 < x_3 = 1.$$

The equation can be written as

$$F(t, x, u, u_x, u_t, u_{xt}) = \frac{\partial}{\partial x} G(t, x, u, u_x, u_t, u_{xt}),$$

where

$$F = u_t \tag{A.1}$$

$$G = \epsilon u_x - \frac{1}{2}u^2 \tag{A.2}$$

We follow the procedure of how MOVCOL forms the DAE system using cubic Hermite collocation for the physical PDE and a finite different scheme for the moving mesh equation. We aim to obtain the following linearly implicit DAE system that can be handle by RADAU5 or other ODE solvers:

$$B(t,Y)Y' = F(t,Y), \tag{A.3}$$

where

$$Y = [u_x(x_1), x_2, u(x_2), u_x(x_2)u_x(x_3)]^T. \tag{A.4}$$

For the physical PDE, we have

$$\begin{aligned}
F(x_{11},t) &= \frac{1}{H_1}[-W_1G_1(t) + W_2G_{1.5}(t) + W_3G_2(t)],\\
F(x_{12},t) &= \frac{1}{H_1}[-W_3G_1(t) - W_2G_{1.5}(t) + W_1G_2(t)],\\
F(x_{21},t) &= \frac{1}{H_2}[-W_1G_2(t) + W_2G_{2.5}(t) + W_3G_3(t)],\\
F(x_{22},t) &= \frac{1}{H_2}[-W_3G_2(t) - W_2G_{2.5}(t) + W_1G_3(t)],
\end{aligned} \tag{A.5}$$

where

$$W_1 = 1 + \frac{2}{\sqrt{3}} \quad W_2 = \frac{4}{\sqrt{3}} \quad W_3 = 1 - \frac{2}{\sqrt{3}}.$$

Substituting (A.1) and (A.2) to (A.5) yields

$$\begin{aligned}
u(x_{11})_t = \tfrac{1}{H_1}\Big\{ &- W_1[\epsilon u_x(x_1) - \tfrac{1}{2}u(x_1)^2]\\
&+W_2[\epsilon u_x(x_{1.5}) - \frac{1}{2}u(x_{1.5})^2] + W_3[\epsilon u_x(x_2) - \frac{1}{2}u(x_2)^2]\Big\}
\end{aligned}$$

$$\begin{aligned}
u(x_{12})_t = \tfrac{1}{H_1}\Big\{ &- W_3[\epsilon u_x(x_1) - \tfrac{1}{2}u(x_1)^2]\\
&-W_2[\epsilon u_x(x_{1.5}) - \frac{1}{2}u(x_{1.5})^2] + W_1[\epsilon u_x(x_2) - \frac{1}{2}u(x_2)^2]\Big\}
\end{aligned}$$

$$\begin{aligned}
u(x_{21})_t = \tfrac{1}{H_2}\Big\{ &- W_1[\epsilon u_x(x_2) - \tfrac{1}{2}u(x_2)^2]\\
&+W_2[\epsilon u_x(x_{2.5}) - \frac{1}{2}u(x_{2.5})^2] + W_3[\epsilon u_x(x_3) - \frac{1}{2}u(x_3)^2]\Big\}
\end{aligned}$$

$$\begin{aligned}
u(x_{22})_t = \tfrac{1}{H_2}\Big\{ &- W_3[\epsilon u_x(x_2) - \tfrac{1}{2}u(x_2)^2]\\
&-W_2[\epsilon \mu_x(x_{2.5}) - \frac{1}{2}u(x_{2.5})^2] + W_1[\epsilon u_x(x_3) - \frac{1}{2}u(x_3)^2]\Big\}.
\end{aligned}$$

Imposing the boundary conditions

$$x_1 = 0, \quad x_3 = 1, \quad u(x_1) = 0, \quad u(x_3) = 0$$

to the above formulas, they are simplified to

$$u(x_{11})_t = \frac{1}{H_1}\left\{-W_1[\epsilon u_x(x_1)] + W_2[\epsilon u_x(x_{1.5}) - \frac{1}{2}u(x_{1.5})^2] + W_3[\epsilon u_x(x_2) - \frac{1}{2}u(x_2)^2]\right\}$$
$$u(x_{12})_t = \frac{1}{H_1}\left\{-W_3[\epsilon u_x(x_1)] - W_2[\epsilon u_x(x_{1.5}) - \frac{1}{2}u(x_{1.5})^2] + W_1[\epsilon u_x(x_2) - \frac{1}{2}u(x_2)^2]\right\}$$
$$u(x_{21})_t = \frac{1}{H_2}\left\{-W_1[\epsilon u_x(x_2) - \frac{1}{2}u(x_2)^2] + W_2[\epsilon u_x(x_{2.5}) - \frac{1}{2}u(x_{2.5})^2] + W_3[\epsilon u_x(x_3)]\right\}$$
$$u(x_{22})_t = \frac{1}{H_2}\left\{-W_3[\epsilon u_x(x_2) - \frac{1}{2}u(x_2)^2] - W_2[\epsilon u_x(x_{2.5}) - \frac{1}{2}u(x_{2.5})^2] + W_1[\epsilon u_x(x_3)]\right\}.$$

Since

$$\begin{aligned} u_t(x,t) &= \frac{du_i}{dt}\Phi_1 + (\frac{du_{x,i}}{dt}H_i + u_{x,i}\frac{dH_i}{dt})\Phi_2 \\ &+ \frac{du_{i+1}}{dt}\Phi_3 + (\frac{du_{x,i+1}}{dt}H_i + u_{x,i+1}\frac{dH_i}{dt})\Phi_4 \\ &- u_x(x,t)(\frac{dx_i}{dt} + s^{(i)}\frac{dH_i}{dt}), \end{aligned} \tag{A.6}$$

$$u_x(x,t) = \frac{1}{H_i}\left(u_i\frac{d\Phi_1}{ds} + u_{x,i}H_i\frac{d\Phi_2}{ds} + u_{i+1}\frac{d\Phi_3}{ds} + u_{x,i+1}H_i\frac{d\Phi_4}{ds}\right). \tag{A.7}$$

Thus

$$\begin{aligned} u_t(x_{11}) &= x_2\Phi_2(s_1)\cdot u_{tx}(x_1) \\ &+ [u_x(x_1)\Phi_2(s_1) + u_x(x_2)\Phi_4(s_1) \\ &+ s_1(u_x(x_1)\frac{d\Phi_2}{ds_1} + \frac{1}{x_2}u(x_2)\frac{d\Phi_3}{ds_1} + u_x(x_2)\frac{d\Phi_4}{ds_1})](x_2)_t \\ &+ \Phi_3(s_1)u_t(x_2) + x_2\Phi_4(s_1)u_{xt}(x_2), \end{aligned} \tag{A.8}$$

$$\begin{aligned} u_t(x_{12}) &= x_2\Phi_2(s_2)\cdot u_{tx}(x_1) \\ &+ [u_x(x_1)\Phi_2(s_2) + u_x(x_2)\Phi_4(s_2) \\ &+ s_2(u_x(x_1)\frac{d\Phi_2}{ds_1} + \frac{1}{x_2}u(x_2)\frac{d\Phi_3}{ds_1} + u_x(x_2)\frac{d\Phi_4}{ds_1})](x_2)_t \\ &+ \Phi_3(s_2)u_t(x_2) + x_2\Phi_4(s_2)u_{xt}(x_2), \end{aligned} \tag{A.9}$$

$$\begin{aligned} u_t(x_{21}) &= \Phi_1(s_1)u_t(x_2) + \Phi_2(s_1)(1-x_2)u_{xt}(x_2) \\ &- [u_x(x_2)\Phi_2(s_1) + u_x(x_3)\Phi_4(s_1) \\ &+ (1-s_1)(\frac{u(x_2)}{1-x_2}\frac{d\Phi_1}{ds_1} + u_x(x_2)\frac{d\Phi_2}{ds_1} + u_x(x_3)\frac{d\Phi_4}{ds_1})](x_2)_t \\ &+ \Phi_4(s_1)(1-x_2)u_{xt}(x_3), \end{aligned} \tag{A.10}$$

and

$$\begin{aligned} u_t(x_{22}) &= \Phi_1(s_2)u_t(x_2) + \Phi_2(s_1)(1-x_2)u_{xt}(x_2) \\ &- [u_x(x_2)\Phi_2(s_2) + u_x(x_3)\Phi_4(s_2) \\ &+ (1-s_2)(\frac{u(x_2)}{1-x_2}\frac{d\Phi_1}{ds_2} + u_x(x_2)\frac{d\Phi_2}{ds_2} + u_x(x_3)\frac{d\Phi_4}{ds_2})](x_2)_t \\ &+ \Phi_4(s_2)(1-x_2)u_{xt}(x_3) \end{aligned} \quad (A.11)$$

The two Gaussian points are

$$s_1 = \frac{1}{2}(1-\frac{1}{\sqrt{3}}), \quad (A.12)$$

and

$$s_2 = \frac{1}{2}(1+\frac{1}{\sqrt{3}}), \quad (A.13)$$

so

$$\Phi_1(s_1) = 1 - 3 * \frac{1}{4}(1-\frac{1}{\sqrt{3}})^2 - \frac{1}{4}(1-\frac{1}{\sqrt{3}})^3 = 0.8471506$$

$$\Phi_1(s_2) = 1 - 3 * \frac{1}{4}(1+\frac{1}{\sqrt{3}})^2 - \frac{1}{4}(1+\frac{1}{\sqrt{3}})^3 = -1.847151$$

$$\Phi_2(s_1) = \frac{1}{2}(1-\frac{1}{\sqrt{3}}) + \frac{1}{2}(1-\frac{1}{\sqrt{3}})^2 + \frac{1}{8}(1-\frac{1}{\sqrt{3}})^3 = 0.3100786$$

$$\Phi_2(s_2) = \frac{1}{2}(1+\frac{1}{\sqrt{3}}) + \frac{1}{2}(1+\frac{1}{\sqrt{3}})^2 + \frac{1}{8}(1+\frac{1}{\sqrt{3}})^3 = 2.523255$$

$$\Phi_3(s_1) = \frac{3}{4}(1-\frac{1}{\sqrt{3}})^2 - \frac{1}{2}(1-\frac{1}{\sqrt{3}})^3 = 0.09622504$$

$$\Phi_3(s_2) = \frac{3}{4}(1+\frac{1}{\sqrt{3}})^2 - \frac{1}{2}(1+\frac{1}{\sqrt{3}})^3 = -0.09622504$$

$$\Phi_4(s_1) = \frac{1}{8}(1-\frac{1}{\sqrt{3}})^3 - \frac{1}{4}(1-\frac{1}{\sqrt{3}})^2 = -0.03522081$$

$$\Phi_4(s_2) = \frac{1}{8}(1+\frac{1}{\sqrt{3}})^3 - \frac{1}{4}(1+\frac{1}{\sqrt{3}})^2 = -0.1314459$$

$$\frac{d\Phi_1}{ds_1} = -6s_1 - 6s_1^2 = -3(1-\frac{1}{\sqrt{3}}) - \frac{3}{2}(1-\frac{1}{\sqrt{3}})^2 = -1.535898$$

$$\frac{d\Phi_1}{ds_2} = -6s_2 - 6s_2^2 = -3(1+\frac{1}{\sqrt{3}}) - \frac{3}{2}(1+\frac{1}{\sqrt{3}})^2 = -8.464102$$

$$\frac{d\Phi_2}{ds_1} = 1 - 4s_1 + 3s_1^2 = 1 - 2(1-\frac{1}{\sqrt{3}}) + \frac{3}{4}(1-\frac{1}{\sqrt{3}})^2 = 0.2886751$$

$$\frac{d\Phi_2}{ds_2} = 1 - 4s_2 + 3s_2^2 = 1 - 2(1+\frac{1}{\sqrt{3}}) + \frac{3}{4}(1+\frac{1}{\sqrt{3}})^2 = -0.2886751$$

$$\frac{d\Phi_3}{ds_1} = 3(1-\frac{1}{\sqrt{3}}) - \frac{3}{2}(1-\frac{1}{\sqrt{3}})^2 = 1$$
$$\frac{d\Phi_3}{ds_2} = 3(1+\frac{1}{\sqrt{3}}) - \frac{3}{2}(1+\frac{1}{\sqrt{3}})^2 = 1$$
$$\frac{d\Phi_4}{ds_1} = \frac{3}{2}(1-\frac{1}{\sqrt{3}})^2 - (1-\frac{1}{\sqrt{3}}) = -0.1547005$$
$$\frac{d\Phi_4}{ds_2} = \frac{3}{2}(1+\frac{1}{\sqrt{3}})^2 - (1+\frac{1}{\sqrt{3}}) = 2.154701.$$

We choose arc-length function as monitor function

$$M = \sqrt{1+u_x^2}$$

From MMPDE4 (2.43)

$$[(M_1 + 2M_2 + M_3)]\,(x_2)_t = \frac{1}{\tau}[M_2 + M_3 - (M_1 + 2M_2 + M_3)x_2], \tag{A.14}$$

we obtain

$$\begin{aligned}
&(\sqrt{1+u_x(x_1)^2} + 2\sqrt{1+u_x(x_2)^2} + \sqrt{1+u_x(x_3)^2})(x_2)_t \\
= \; & \frac{1}{\tau}\Big[\sqrt{1+u_x(x_3)} + \sqrt{1+u_x(x_2)} - \\
& - (\sqrt{1+u_x(x_3)} + 2\sqrt{1+u_x(x_2)} + \sqrt{1+u_x(x_1)})x_2\Big].
\end{aligned}$$

From MMPDE6 (2.45)

$$(x_2)_t = \frac{1}{4\tau}[M_3 + M_2 - (M_1 + 2M_2 + M_3)x_2],$$

which yields

$$\begin{aligned}
(x_2)_t \;=\; & \frac{1}{4\tau}\Big[\sqrt{1+u_x(x_3)^2} + \sqrt{1+u_x(x_2)^2} \\
& + \left(\sqrt{1+u_x(x_1)^2} + 2\sqrt{1+u_x(x_2)^2} + \sqrt{1+u_x(x_3)^2}\right) x_2\Big].
\end{aligned}$$

So for the case with MMPDE4

$$B_4 = \begin{bmatrix}
0.3101x_2 & Q_1 & 0.0962 & -0.0352 & 0 \\
2.5232x_2 & Q_2 & -0.0962 & -0.1314x_2 & 0 \\
0 & Q_3 & 0.8472 & 0.3101(1-x_2) & -0.0352(1-x_2) \\
0 & Q_4 & -1.8471 & 0.3101(1-x_2) & -0.1314(1-x_2) \\
0 & Q_5 & 0 & 0 & 0
\end{bmatrix}.$$

For the case with MMPDE6

$$B_6 = \begin{bmatrix} 0.3101x_2 & Q_1 & 0.0962 & -0.0352 & 0 \\ 2.5232x_2 & Q_2 & -0.0962 & -0.1314x_2 & 0 \\ 0 & Q_3 & 0.8472 & 0.3101(1-x_2) & -0.0352(1-x_2) \\ 0 & Q_4 & -1.8471 & 0.3101(1-x_2) & -0.1314(1-x_2) \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

where

$$\begin{aligned} Q_1 &= u_x(x_1)\Phi_2(s_1) + u_x(x_2)\Phi_4(s_1) \\ &\quad + s_1\left(u_x(x_1)\frac{d\Phi_2}{ds_1} + \frac{1}{x}u(x_2)\frac{d\Phi_3}{ds_1} + u_x(x_2)\frac{d\Phi_4}{ds_1}\right) \\ &= 0.3711u_x(x_1) + \frac{0.2113}{x}u(x_2) - 0.0679u_x(x_2), \end{aligned}$$

$$\begin{aligned} Q_2 &= u_x(x_1)\Phi_2(s_2) + u_x(x_2)\Phi_4(s_1) \\ &\quad + s_2\left(u_x(x_1)\frac{d\Phi_2}{ds_1} + \frac{1}{x}u(x_2)\frac{d\Phi_3}{ds_1} + u_x(x_2)\frac{d\Phi_4}{ds_1}\right) \\ &= 2.7510u_x(x_1) + \frac{0.7887}{x}u(x_2) - 0.1572u_x(x_2), \end{aligned}$$

$$\begin{aligned} Q_3 &= (1-s_1)(\frac{u(x_2)}{1-x_2}\frac{d\Phi_1}{ds_1} + u_x(x_2)\frac{d\Phi_2}{ds_1} + u_x(x_3)\frac{d\Phi_4}{ds_1}) \\ &\quad -(u_x(x_2)\Phi_2(s_1) + u_x(x_3)\Phi_4(s_1)) \\ &= -1.2114\frac{u(x_2)}{1-x_2} - 0.0824u_x(x_2) - 0.0868u_x(x_3), \end{aligned}$$

$$\begin{aligned} Q_4 &= (s_2-1)(\frac{u(x_2)}{1-x_2}\frac{d\Phi_1}{ds_2} + u_x(x_2)\frac{d\Phi_2}{ds_2} + u_x(x_3)\frac{d\Phi_4}{ds_2}) \\ &\quad -(u_x(x_2)\Phi_2(s_2) + u_x(x_3)\Phi_4(s_2)) \\ &= 1.7885\frac{u(x_2)}{1-x_2} - 2.4623u_x(x_2) - 0.3239u_x(x_3), \end{aligned}$$

and

$$Q_5 = \sqrt{1+u_x(x_1)^2} + 2\sqrt{1+u_x(x_2)^2} + \sqrt{1+u_x(x_3)^2}$$

.

Next,

$$\begin{aligned} u(x_{1.5}) &= u_x(x_1)x_2\Phi_2(s_1) + u(x_2)\Phi_3(s_1) + u_x(x_2)x_2\Phi_4(s_1) \\ &= 0.3101u_x(x_1)x_2 + 0.0962u(x_2) - 0.0352u_x(x_2)x_2, \end{aligned}$$

$$\begin{aligned}
u(x_{2.5}) &= u(x_2)\Phi_1(s_2) + u_x(x_2)\Phi_2(s_2) + u_x(x_3)(1-x_2)\Phi_4(S_2) \\
&= -1.8471u(x_2) + 2.5233u_x(x_2) - 0.1314u_x(x_3)(1-x_2),
\end{aligned}$$

$$\begin{aligned}
u_x(x_{1.5}) &= u_x(x_1)\frac{d\Phi_2}{dS_1} + \frac{u(x_2)}{x_2}\frac{d\Phi_3}{dS_1} + u_x(x_2)\frac{d\Phi_4}{dS_1} \\
&= -1.5359u_x(x_1) + \frac{u(x_2)}{x_2} - 0.1547u_x(x_2),
\end{aligned}$$

$$\begin{aligned}
u_x(x_{2.5}) &= \frac{1}{1-x_2}u(x_2)\frac{d\Phi_1}{dS_2} + u_x(x_2)\frac{d\Phi_2}{dS_2} + u_x(x_3)\frac{d\Phi_4}{dS_2} \\
&= \frac{-8.4641}{1-x_2}u(x_2) - 0.2887u_x(x_2) + 2.1547u_x(x_3),
\end{aligned}$$

$$\begin{aligned}
f_1 = {} & -0.0481u_x(x_1)^2x_2 - 0.0006u_x(x_2)^2x_2 - 0.0109u_x(x_1)u(x_2)x_2 \\
& -0.0298u_x(x_1)u(x_2) - 0.0017u_x(x_2)u(x_2) + \frac{0.0727u(x_2)^2}{x_2} \\
& -\frac{0.512\epsilon u_x(x_2)}{x_2} - \frac{2.1547\epsilon u_x(x_1)}{x_2} - \frac{3.5470\epsilon u(x_1)}{x_2} + \frac{2.3094\epsilon u(x_2)}{x_2^2},
\end{aligned}$$

$$\begin{aligned}
f_2 = {} & 0.0481u_x(x_1)^2x_2 + 0.0006u_x(x_2)^2x_2 + 0.0109u_x(x_1)u(x_2)x_2 \\
& +0.0298u_x(x_1)u(x_2) + 0.0017u_x(x_2)u(x_2) - \frac{1.0727u(x_2)^2}{x_2} \\
& -\frac{1.7974\epsilon u_x(x_2)}{x_2} - \frac{2.1547\epsilon u_x(x_1)}{x_2} - \frac{3.5470\epsilon u(x_1)}{x_2} + \frac{2.3094\epsilon u(x_2)}{x_2^2},
\end{aligned}$$

$$\begin{aligned}
f_3 = {} & \frac{-2.9198u(x_2)^2}{1-x_2} - \frac{7.3520u_x(x_2)^2}{1-x_2} - \frac{0.02u_x(x_3)^2}{1-x_2} + \frac{10.7637u(x_2)u_x(x_2)}{1-x_2} \\
& -0.5605u(x_2)u_x(x_3) + 0.7658u_x(x_2)u_x(x_3) - \frac{2.1547\epsilon u_x(x_2)}{1-x_2} \\
& -\frac{19.5470\epsilon u(x_2)}{(1-x_2)^2} - \frac{0.6667\epsilon u_x(x_2)}{1-x_2} + \frac{4.8214\epsilon u_x(x_3)}{1-x_2} + \frac{0.0774}{1-x_2},
\end{aligned}$$

$$\begin{aligned} f_4 &= \frac{3.9198u(x_2)^2}{1-x_2} + \frac{7.3520u_x(x_2)^2}{1-x_2} + \frac{0.02u_x(x_3)^2}{1-x_2} - \frac{10.7637u(x_2)u_x(x_2)}{1-x_2} \\ &+0.5605u(x_2)u_x(x_3) - 0.7658u_x(x_2)u_x(x_3) + \frac{2.1547\epsilon u_x(x_2)}{1-x_2} \\ &+\frac{19.5470\epsilon u(x_2)}{(1-x_2)^2} + \frac{0.1547\epsilon u_x(x_2)}{1-x_2} - \frac{2.8214\epsilon u_x(x_3)}{1-x_2} + \frac{1.07735}{1-x_2}, \end{aligned}$$

$$\begin{aligned} f_5 &= \frac{1}{\tau}\Big[\sqrt{1+u_x(x_3)} + \sqrt{1+u_x(x_2)} \\ &\quad -(\sqrt{1+u_x(x_3)} + 2\sqrt{1+u_x(x_2)} + \sqrt{1+u_x(x_1)})x_2\Big], \end{aligned}$$

and

$$\begin{aligned} f_6 &= \frac{1}{4\tau}\Big[\sqrt{1+u_x(x_3)^2} + \sqrt{1+u_x(x_2)^2} \\ &\quad + \left(\sqrt{1+u_x(x_1)^2} + 2\sqrt{1+u_x(x_2)^2} + \sqrt{1+u_x(x_3)^2}\right) x_2\Big]. \end{aligned}$$

Thus for MMPDE4, we have

$$B(t,Y)_4 Y' = F(t,Y') =: \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix};$$

we also have

$$B(t,Y)_6 Y' = F(t,Y') =: \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_6 \end{pmatrix}$$

for MMPDE6.

For the smooth version of MMPDE4

$$\frac{\partial}{\partial \xi}\left\{\frac{\partial \dot{x}}{\partial \xi}\tilde{M}\right\} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left\{\frac{\partial x}{\partial \xi}\tilde{M}\right\}. \tag{A.15}$$

The discretizations are given by a finite difference scheme

$$M_{i-\frac{1}{2}} z_{i+\frac{1}{2}} = M_{i+\frac{1}{2}} z_{i-\frac{1}{2}}, \tag{A.16}$$

where

$$z_{i+\frac{1}{2}} = y_{i+\frac{1}{2}} - \frac{1}{\lambda^2 h^2}(y_{i+\frac{1}{2}} - 2y_{i+\frac{1}{2}} + y_{i-\frac{1}{2}}), \tag{A.17}$$

and

$$y_{i+\frac{1}{2}} = \frac{\frac{1}{N}}{\dot{x}_{i+1} - \dot{x}_i + \frac{1}{\tau}(x_{i+1} - x_i)}. \tag{A.18}$$

After substituting (A.17) and (A.18) into (A.16), we obtain a nonlinear term like $\dot{x}_{i+1}\dot{x}_i$. Software like RADAU which is designed for linear implicit differential equations will not be able to solve this type of DAEs. Therefore we can not apply this form to the smooth version of MMPDE4.

Bibliography

[1] De.Boor, Good approximation by splines with variable knots II, in Springer Lecture Notes Series 363, Springer-Verlag, Berlin, 1973.

[2] J. Bruder , Numerical results for a parallel linearly-implicit Runge-Kutta method, Computing, 59, 139-151, 1997.

[3] RRichard L. Burden, J. Douglas Faires, Numerical Analysis.(eighth edition), 2004.

[4] KK.E.Brenhan, S.L.Campbell, and L.R.Petzold, Numerical Solution of Initial-Value Problems in Differential- Algebraic Equations, SIAM, (second edition), 1996.

[5] EE.A.Dorfi and L.O'c.Drury, Simple adaptive grids for 1-D initial value problems, J. Comput. Phys, 69, 175-195, 1987.

[6] A.Flitton and J.King, Moving-boundary and fixed-domain problems for a sixth-order thin-film equation, European Journal on Applied Mathematics 15, 713-754, 2004.

[7] RR.M.Furzeland, J.G.Verwer, and P.A.Zegeling, A Numerical study of three moving grid methods for one-dimensional partial differential equations which are based on the method of lines, J. Comput. Phys., 69, 175-195,1987.

[8] CC.W.Gear, The simultaneous numerical solution of differential-algebraic equations, IEEE Trans.Circuit Theory, CT-18, 89-95, 1971.

[9] EE.Hairer, S.P.Nørsett and G.Wanner, Solving Ordinary Differential Equations I–Nonstiff Problems, Springer-Verlag, 1987.

[10] WWeizhang Huang, Yuhe Ren, Robert D Russell, Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle, SIAM J. Numer, 31, 709-730, 1994.

[11] WWeizhang Huang, Yuhe Ren, Robert D Russell, Moving mesh methods based upon moving mesh partial differential equations, J.Comp.Phys, 113, 279-290, 1994.

[12] WWeizhang Huang, Robert D Russell, A moving collocation method for solving time dependent partial differential equations, J IMACS, 20,101-116, 1996.

[13] WWeizhang Huang, Robert D Russell, Analysis of Moving mesh partial differential equations with spatial smoothing, J.Siam. anal, 34,1106-1126, 1997.

[14] PP.J.van der Houwen and J.J.B.de Swart, Parallel Linear system solver for Runge-Kutta methods, Advances in Computational Mathematics, 7, (157-181), 1997.

[15] EE.Hairer, and G.Wanner, Solving Ordinary Differential Equations II–stiff and Differential-Algebraic Problems , Springer-Verlag, 1991.

[16] KK.R.Jackson and R.Sacks-Davis. An alternative implementation of variable step-size multistep formulas for stiff ODEs, ACM Trans.Math.Software, 6, 295-318,1980.

[17] S.Li and L.Petzold, Moving mesh Methods with Upwinding Schemes for Time-Dependent PDEs, J.Comp.Phys, 131, 368-377, 1997.

[18] S.Li, L.Petzold and Y.Ren, Stability of Moving Mesh Systems of Partial Differential Equations, SIAM J.Sci.Comput. 20, (719-738), 1998.

[19] KK.Miller. Moving finite elements II, SIAM J. Numer. Anal, 18, (1033-1057), 1981.

[20] KK.Miller, R.N. Miller, Moving finite elements I, SIAM J. Numer. Anal, 18, 1019-1032, 1981.

[21] L.Petzold, Differential /algebraic equations are not ODEs, SIAM J.Sci.Statist.Comput. 3, 367-384, 1982.

[22] L.Petzold , Obervations on an adpative moving grid method for one-dimensional systems of partial differential equations. Appl. Numer.Math, 3, 347-360, 1987.

[23] A.B.White, JR, On selection of equidistributing meshes for two-point boundary-value problems, SIAM J.Numer.Anal. 16, 472-502, 1979.

[24] http://www.cwi.nl/archive/projects/PSIDE/.

[25] http://www.cwi.nl/archive/projects/PSIDE/user.pdf.

[26] P.Zegeling and H.Kok, Adaptive moving mesh computations for reaction-diffusion systems.Journal. Comp. Appl Math, 168, 519-528, 2004.