Numerical Laplace Transformation Methods for Integrating Linear Parabolic Partial Differential Equations

by

Edgard NGOUNDA

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Applied Mathematics at the University of Stellenbosch

Applied Mathematics, Department of Mathematical Sciences
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Professor J.A.C Weideman

January 2009

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: ....
E. NGOUNDA

Date: ....

Abstract

Numerical Laplace Transformation Methods for Integrating Linear Parabolic Partial Differential Equations

E. NGOUNDA

Applied Mathematics, Department of Mathematical Sciences
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: MSc (Mathematical Sciences)

January 2009

In recent years the Laplace inversion method has emerged as a viable alternative method for the numerical solution of PDEs. Effective methods for the numerical inversion are based on the approximation of the Bromwich integral.

In this thesis, a numerical study is undertaken to compare the efficiency of the Laplace inversion method with more conventional time integrator methods. Particularly, we consider the method-of-lines based on MATLAB's **ODE15s** and the Crank-Nicolson method.

Our studies include an introductory chapter on the Laplace inversion method. Then we proceed with spectral methods for the space discretization where we introduce the interpolation polynomial and the concept of a differentiation matrix to approximate derivatives of a function. Next, formulas of the numerical differentiation formulas (NDFs) implemented in **ODE15s**, as well as the well-known second order Crank-Nicolson method, are derived. In the Laplace method, to compute the Bromwich integral, we use the trapezoidal rule over a hyperbolic contour. Enhancement to the computational efficiency of these methods include the LU as well as the Hessenberg decompositions.

In order to compare the three methods, we consider two criteria: The number of linear system solves per unit of accuracy and the CPU time per unit of accuracy. The numerical results demonstrate that the new method, i.e., the Laplace inversion method, is accurate to an exponential order of convergence compared to the linear convergence rate of the **ODE15s** and the Crank-Nicolson methods. This exponential convergence leads to high accuracy with only a few linear system solves. Similarly, in terms of computational cost, the Laplace inversion method is more efficient than **ODE15s** and the Crank-Nicolson method as the results show.

Finally, we apply with satisfactory results the inversion method to the axial dispersion model and the heat equation in two dimensions.

Uittreksel

Numeriese Laplace Transformasiemetodes vir die Integrasie van Lineêre Paraboliese Parsiële Differensiaalvergelykings

("Numerical Laplace Transformation Methods for Integrating Linear Parabolic Partial Differential Equations")

E. NGOUNDA

Toegepaste Wiskunde, Departement Wiskundige Wetenskappe
Universiteit van Stellenbosch
Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MSc (Wiskundige Wetenskappe)

Januarie 2009

In die afgelope paar jaar het die Laplace omkeringsmetode na vore getree as 'n lewensvatbare alternatiewe metode vir die numeriese oplossing van PDVs. Effektiewe metodes vir die numeriese omkering word gebasseer op die benadering van die Bromwich integraal.

In hierdie tesis word 'n numeriese studie onderneem om die effektiwiteit van die Laplace omkeringsmetode te vergelyk met meer konvensionele tydintegrasie metodes. Ons ondersoek spesifiek die metode-van-lyne, gebasseer op MATLAB se **ODE15s** en die Crank-Nicolson metode.

Ons studies sluit in 'n inleidende hoofstuk oor die Laplace omkeringsmetode. Dan gaan ons voort met spektraalmetodes vir die ruimtelike diskretisasie, waar ons die interpolasie polinoom invoer sowel as die konsep van 'n differensiasie-matriks waarmee afgeleides van 'n funksie benader kan word. Daarna word formules vir die numeriese differensiasie formules (NDFs) ingebou in **ODE15s** herlei, sowel as die welbekende tweede orde Crank-Nicolson metode. Om die Bromwich integraal te benader in die Laplace metode, gebruik ons die trapesiumreël oor 'n hiperboliese kontoer. Die berekeningskoste van al hierdie metodes word verbeter met die LU sowel as die Hessenberg ontbindings.

Ten einde die drie metodes te vergelyk beskou ons twee kriteria: Die aantal lineêre stelsels wat moet opgelos word per eenheid van akkuraatheid, en die sentrale prosesseringstyd per eenheid van akkuraatheid. Die numeriese resultate demonstreer dat die nuwe metode, d.i. die Laplace omkeringsmetode, akkuraat is tot ’n eksponensiële orde van konvergensie in vergelyking tot die lineêre konvergensie van **ODE15s** en die Crank-Nicolson metodes. Die eksponensiële konvergensie lei na hoë akkuraatheid met slegs ’n klein aantal oplossings van die lineêre stelsel. Netso, in terme van berekeningskoste is die Laplace omkeringsmetode meer effektief as **ODE15s** en die Crank-Nicolson metode.

Laastens pas ons die omkeringsmetode toe op die aksiale dispersiemodel sowel as die hittevergelyking in twee dimensies, met bevredigende resultate.

Acknowledgments

I would like to express my sincere gratitude to the following people and organizations.

- The almighty God for the strength he instills in me to keep going.
- My sponsor, the Gabonese government through the Bourses et Stages for the financial support.
- Prof JAC Weideman, for his guidance and support throughout the duration of this study. He has provided me with many valuable insights and references and of course suggested most of the research topic considered in this thesis. Also his meticulous proof reading of several draft of this thesis has been extremely helpful.
- The Division of Applied Mathematics of the Mathematical Sciences department for the working facilities provided.
- Special thanks goes to "le collegue" Edson Pindza with whom I shared the joy and frustrations of this thesis during the past two years.
- Then I wish to thank the Gabonese students for the social life during these years of studying, especially to friends Stephane and Marlyse. The unconditional support and prayer of a special person named Aurixiane V. Komba is also acknowledged.
- And finally, loving thanks to my family for inspiring me: Honorine Mbouri (mother), Mr and Mme Ompiguy, Jules Anoumba. David, Zita, Sandra, Aude and Ralph.

Contents

Declaration **i**

Abstract **ii**

Uittreksel **iv**

Acknowledgments **vi**

Contents **vii**

List of Figures **ix**

List of Tables **xi**

Nomenclature **xii**

1 Introduction **1**

1.1 Laplace transform 2

1.2 Trapezoidal rule 3

1.3 Problem statement 4

1.4 Model problem 5

1.5 Thesis layout 6

2 Space discretization: Spectral Methods **8**

2.1 Polynomial interpolation 9

2.2 Differentiation matrices 10

3 Time integration **16**

3.1 The Method-of-lines 16

3.2 The Laplace inversion method 23

4 Numerical comparison **37**

4.1 Criterion I: Accuracy per linear systems solve 37

4.2 Criterion II: Accuracy per CPU time 43

5 Applications **47**

5.1 Axial dispersion model 47
5.2 Heat equation in two dimensions 55

6 Conclusion 61
6.1 Topics for further work 62

Appendices 63

A Matrix Decomposition 64

List of References 67

List of Figures

1.1 Solution of problem (1.4.1)–(1.4.3) computed by (1.4.4)–(1.4.5) . . . 6

2.1 Chebyshev approximation to derivatives of $f(x) = e^x$. 13

3.1 CPU time (in seconds) of one step of the Crank-Nicolson (3.1.12) with and without the LU factorization. 22

3.2 Hyperbolic contour (3.2.3). 24

3.3 CPU time of the inversion formula (3.2.17) with and without the Hessenberg decomposition. 29

3.4 Convergence curve obtained when solving the semi-discrete problem (2.2.17) with the inversion formula (3.2.12) for different values of t. The dash-dot curve shows the error of the approximation (3.2.16), and the dashed line the theoretical error (3.2.35). The error norm is the infinite norm $||\boldsymbol{u}_M(t) - \boldsymbol{u}(t)||_\infty$. 32

3.5 Actual error (dashed curve) and theoretical error (continuous curve) for $\Lambda = 2$, 10 and 50. We observe a good agreement between the numerical and theoretical results. Here $t_0 = 1/\Lambda$ and the interval is $[1/\Lambda,\, 1]$. 35

4.1 Comparison of the error of the inverse Laplace transform method (ILT), **ODE15s** (ODE15s) and Crank-Nicolson method (CN). M is the number of linear system solves used to compute the solution of the model problem (3.2.13) at $t = 1$. 38

4.2 Comparison of the error (4.1.8) of the Method 1; Method 2; **ODE15s** and the Crank-Nicolson method. M is the number of linear system solves to compute the solution of the model problem (3.2.13). The results were computed for $\Lambda = 2$ (a), $\Lambda = 3$ (b), $\Lambda = 5$ (c) and $\Lambda = 50$ (e), on the interval $[1/\Lambda,\, 1]$ and $\ell = 8$. 42

4.3 Convergence error of **ODE15s** and Method 2 for $\Lambda = 10^{15}$. 43

4.4 Convergence error of **ODE15s** and Method 2 for $\Lambda = 10^{20}$. 44

4.5 CPU time vs error of Method 1 (ILT1), Method 2 (ILT2) **ODE15s** and Crank Nicolson, on the interval interval $[0.02,\, 1]$, $\ell = 8$. Here the values of the CPU time of Method 1 should be multiplied by 8. 45

4.6 CPU time vs error for **ODE15s** and Method 2 on the interval $[1/\Lambda, 1]$, for huge values of Λ. 46

5.1 Axial distribution of the roots of the equation (5.1.8) 49
5.2 Exact solution (5.1.11) 50
5.3 Solution at the exit point, i.e., $x = 1$ of equation (5.1.16) with Laplace inversion method. Here $P_e = 20$. 53
5.4 Convergence error rate of the Laplace inversion 53
5.5 Solution of equation (5.1.16) in 2-D with Laplace transform method. Here $P_e = 20$. 54
5.6 Solution of the 2D problem at $t = 0$ 58
5.7 Solution of the 2D problem at $t = 2$ 58
5.8 Solution of the 2D problem at $t = 5$ 59
5.9 Solution of the 2D problem at $t = 15$ 59
5.10 Solution of the 2D problem at $t = 20$ 60

List of Tables

3.1 Stiffness ratio of (2.2.17) for $N \times N$ matrix $D_N^{(2)}$, s_1 is the ratio obtained numerically whereas s_2 is the theoretical estimate derived [1]. 18
3.2 Optimal parameters of the contour (3.2.3) for different Λ. 34
3.3 Convergence rate of the method (3.2.12) on an interval $[1/\Lambda, 1]$ for a few values of Λ. The first column indicates the number of points M in the trapezoid rule. 35

4.1 Comparison of the error defined by (4.1.8) of Method 1, Method 2, **ODE15s** and the Crank-Nicolson method on the interval $[t_0/\Lambda,\ 1]$ for $\Lambda = 50$. M is the number of functions evaluations (linear system solves) executed by each method. Here $\ell = 8$. 41
4.2 Optimal parameters for $\Lambda = 10^{15}$ and 10^{20}. 43

Nomenclature

$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
$[a, b]$	Closed interval $[a, b] := \{x \in \mathbb{R} : a \leq x \leq b\}$
(a, b)	Open interval $(a, b) := \{x \in \mathbb{R} : a < x < b\}$
MOL	Methods-of-lines
ODE	Ordinary differential equation
PDE	Partial differential equation
BDF	Backward differentiation formula
BDFk	Backward differentiation formula of order k
NDF	Numerical differentiation formula
NDFk	Numerical differentiation formula of order k
max	Maximum of
Pe	Peclet number
δ	Dirac delta function
δ_{jk}	Kronecker delta-symbol
λ_j	eigenvalue of the second order differentiation matrix $D_N^{(2)}$
N	Order of the differentiation matrix $D_N^{(m)}$
M	Number of linear system solves
m	mth order differentiation matrix $D_N^{(m)}$
x	Space coordinate
I	$N \times N$ identity matrix
m_{ij}	Matrix entry ij
t	Time
$O\left(g\left(x\right)\right)$	a function $f\left(x\right)$ for which there exist constants $x_0 > 0$ and $C > 0$ where $\lvert f\left(x\right)\rvert \leq Cg\left(x\right)$ for all $x \geq x_0$.
$\mathrm{erfc}(x)$	$= 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-s^2} ds$

Chapter 1

Introduction

Parabolic equations are second-order partial differential equations that describe a variety of problems in science and engineering, including heat diffusion and stock option pricing. These problems, also known as evolution problems, describe physical or mathematical systems with a time variable. They behave essentially like heat diffusing through a medium like a metal plate. In this thesis, we consider as model equation the one dimensional heat equation

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}, \tag{1.0.1}$$

where κ is a constant. The solutions give the temperature u at distance x and time t. For such problems the boundary conditions are usually known as well as the initial temperature. The efficient computation of this PDE is the main topic of this thesis.

Our approach to solving (1.0.1) follows two steps: first the PDE is discretized with respect to space, thereby generating a system of ordinary differential equations in time t. Second, the resultant system of ODEs is solved by applying a suitable time integrating method.

To semi-discretize the PDE, we consider spectral methods. Given a set of points, the idea behind spectral methods is as follows. Interpolate the unknown solution, then differentiate the interpolant polynomial at the mesh points. This discretization process can be converted into matrix form as we show in the next chapter. As a result, the approximation is reduced to the solution of a matrix equation. Note, however, since we are concerned with parabolic PDEs, the resulting differentiation matrix has its eigenvalues all real and negative [1].

For the numerical solution of the semi-discrete PDE, we shall consider in this thesis two integrating methods:

- the well-known method-of-lines, based on MATLAB's built-in code **ODE15s** and the Crank-Nicolson method, and

- the Laplace inversion method.

The latter method is relatively new in this context; see [2], [3], [4]. The objective of this thesis is therefore to compare this method with the more classical method-of-lines.

1.1 Laplace transform

The Laplace transformation method plays a significant role in application areas such as physics and engineering, with a growing interest in areas such as computational finance. It is a powerful tool for the solution of ordinary differential equations as well as partial differential equations.

The Laplace transform of a function $u(t)$ defined on $[0, \infty)$ is given by

$$U(z) = \int_0^\infty e^{-zt} u(t) dt, \qquad \operatorname{Re} z > \sigma_0. \tag{1.1.1}$$

We assume that $u(t)$ is a function of exponential order as $t \to +\infty$, i.e., there exist a σ_0 such that $e^{-\sigma_0 t}|u(t)| \leq +\infty$. In addition, if $u(t)$ is absolutely integrable for $t > 0$, then the Laplace integral (1.1.1) converges for all $\operatorname{Re} z > \sigma_0$. It then defines a single valued analytic function for $\operatorname{Re} z > \sigma_0$. Here, σ_0 is the so-called convergence abscissa of $U(s)$. We have defined $u(t) = 0$ for $t < 0$. A review of the properties of the Laplace transform as well as it applications can be found in [5; 6; 7].

The main difficulty in using the Laplace transform is finding the inverse. Unless it is given in a table, the Bromwich integral has to be evaluated, namely

$$u(t) = \frac{1}{2\pi i} \int_{\sigma - i\infty}^{\sigma + i\infty} e^{zt} U(z) dz, \qquad \sigma > \sigma_0, \ t > 0. \tag{1.1.2}$$

This formula is valid if all singularities of $U(z)$ are all located to the left of the vertical line $x = \sigma$, i.e., $\operatorname{Re} z < \sigma$. When all complex integration techniques fail to evaluate (1.1.2) analytically, one has to rely on numerical methods.

For the numerical evaluation of (1.1.2), we consider the parameterisation $z = \sigma + iy$, $-\infty < y < \infty$. The integral (1.1.2) can then be rewritten as

$$u(t) = \frac{e^{\sigma t}}{2\pi i} \int_{-\infty}^{\infty} e^{iyt} U(\sigma + iy) dy. \tag{1.1.3}$$

Unfortunately, in most cases, this problem is difficult to solve numerically for many reasons: firstly, the integrand is highly oscillatory on the infinite line, i.e., as $y \to \pm\infty$. Secondly, the transform $U(\sigma + iy)$ may decay slowly as $y \to \pm\infty$ [8].

There are many methods for the numerical inversion of the Laplace transform. Davis and Martin [9], Duffy [8], and Narayanan [10] review some methods developed in the last three decades. These algorithms have been classified by Abate and Valko [11] according to the basic method used as: (1) Fourier series expansion, (2) Laguerre function expansion, (3) combination of Gaver functionals and (4) deformation of the Bromwich contour. We shall only consider methods in class (4).

To get around the oscillatory nature of the integrand of (1.1.3), Talbot [2] developed a method based on the trapezoidal rule. He suggested the deformation of the contour of integration so that it starts and ends in the left half-plane where the integrand converges rapidly as $\operatorname{Re} z \to -\infty$.

This deformation of the contour is possible by Cauchy's integral theorem [7, p. 141]. Cauchy's theorem is applicable provided that all singularities of the transform $U(z)$ are all contained in the interior of the new contour and that $|U(z)| \to 0$ as $|z| \to \infty$ in the half-plane $\operatorname{Re} z < \sigma_0$ [12]. Such contours are used in [2; 3; 4], all of which are of the form

$$z = z(\ell), \quad -\infty < \ell < \infty,$$

with the property that $\operatorname{Re} z \to -\infty$ as $\ell \to \pm\infty$. On these contours, the integral (1.1.2) therefore becomes

$$u(t) = \frac{1}{2\pi i} \int_{-\infty}^{\infty} e^{z(\ell)t} U\left(z(\ell)\right) z'(\ell) d\ell. \tag{1.1.4}$$

The efficiency of the Talbot approach depends on the choice of the contour, as well as the number of function evaluations in the trapezoidal rule. As a contour of integration we shall use a hyperbolic contour (see Section 3.2).

1.2 Trapezoidal rule

Under certain conditions, the trapezoidal rule is a natural candidate to approximate the integral (1.1.4) [12]. We recall a few well known formulas related to the trapezoidal rule. Consider the definite integral

$$I(f) = \int_a^b f(x)dx,$$

where we assume $f(x)$ is integrable on the interval $[a, b]$. The integral $I(f)$ can be approximated by the trapezoidal rule $T_M(f)$ defined by

$$T_M(f) = \frac{h}{2}\left((f(a) + f(b)) + 2\sum_{j=1}^{M-1} f(x_j) \right). \tag{1.2.1}$$

Here M is the number of subintervals in the partitioning of $[a, b]$, and $h = (b-a)/M$, the uniform width. The trapezoidal rule is well known to have a convergence rate of $O(h^{-2})$. However, for smooth periodic functions integrated over one period, the convergence rate is faster than the usual $O(h^{-2})$; an exponential convergence rate is routinely achieved [12].

The Euler-Maclaurin summation formula, stated below, provides a method for estimating the convergence rate of the trapezoidal rule for a smooth function $f(x)$.

Theorem 1.2.1 *Let $m \geq 0$, $M \geq 1$, and define $h = (b-a)/M$, $x_j = jh$ for $j = 0, 1, ..., M$. Further assume that $f(x)$ is $2m+2$ times continuously differentiable on $[a, b]$ for some $m \geq 0$. Then, the error in the trapezoidal rule is given by*

$$\begin{aligned} I(f) - T_M(f) &= -\sum_{k=1}^{m} \frac{b_{2k}}{(2k)!} h^{2k} \left[f^{(2k-1)}(b) - f^{(2k-1)}(a) \right] \\ &+ \frac{h^{2m+2}}{(2m+2)!} \int_a^b \bar{B}_{2m+2}\left(\frac{x-a}{h}\right) f^{(2m+2)}(\xi) d\xi. \end{aligned} \tag{1.2.2}$$

The b_k are the Bernoulli numbers, and $\bar{B}_{2m+2}(x)$ is the periodic extension of the Bernoulli polynomials $B_{2m+2}(x)$.

Proof: see [13, p. 285].

From Theorem 1.2.1, one can see that the trapezoidal rule has an accuracy of $O(h^{-2})$ when $f^{'}(b) - f^{'}(a) \neq 0$. But, when $f^{'}(b) - f^{'}(a) = 0$, a higher order accuracy can be achieved. This happens, in particular, when the function is periodic with period $b - a$. In such cases, an exponential convergence rate can be achieved as stated in the following theorem.

Theorem 1.2.2 *Let $f : \mathbb{R} \to \mathbb{R}$, be analytic and 2π-periodic. Then there exists a strip $D = \mathbb{R} \times i(-c, c) \subset \mathbb{C}$ with $c > 0$ such that f can be extended to a bounded analytic and 2π-periodic function $f : D \to \mathbb{C}$. The error for the trapezoidal rule can be estimated by*

$$|I(f) - T_M(f)| \leq 4\pi \frac{K}{e^{cM} - 1}$$

where K denotes a bound for the analytic function $f(x)$ on D.

Proof: see [14, p. 211].

In the case of the improper integral (1.1.4) posed on the real line, a more convenient theorem proved by Martensen [15] guarantees an exponential convergence rate as we discuss in Section 3.2.

1.3 Problem statement

In the literature, the Laplace transform and the method-of-lines based on MATLAB's **ODE15s** and the Crank-Nicolson method are satisfactorily used to solve PDEs [16; 8; 11]. However, a direct comparison of these three methods lacks.

The purpose of this thesis is therefore to compare the efficiency of the Crank-Nicolson method, MATLAB's ODE solver **ODE15s** and the Laplace inversion method. In assessing the efficiency of these methods, we shall look at the performance of all three methods in terms of (*a*) accuracy as a function of number of linear systems solved, and (*b*) accuracy as a function of computational time.

1.4 Model problem

As model problem, we consider the standard heat equation with constant coefficients. This equation models the flow of heat in a bar of length π, say. In nondimensional form this is given by

$$\frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2}, \qquad 0 < x < \pi. \tag{1.4.1}$$

As boundary conditions we take

$$v(0,t) = 0, \qquad v(\pi,t) = 1, \qquad t > 0, \tag{1.4.2}$$

and initial condition

$$v(x,0) = 0. \tag{1.4.3}$$

The exact solution can be expanded as a Fourier sine series [17, p. 91]

$$v(x,t) = \frac{x}{\pi} + \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^n}{n}\sin(nx)e^{-n^2 t}, \tag{1.4.4}$$

but for small t this series is slowly convergent. A more efficient solution for small t is an infinite series involving the complementary error function [17, p. 91]

$$v(x,t) = \sum_{n=0}^{\infty}\Big[\operatorname{erfc}\Big(\frac{\pi - x + 2\pi n}{2\sqrt{t}}\Big) - \operatorname{erfc}\Big(\frac{\pi + x + 2\pi n}{2\sqrt{t}}\Big)\Big]. \tag{1.4.5}$$

For a numerical solution of (1.4.1)–(1.4.3), we introduce a substitution

$$u(x,t) = v(x,t) - x/\pi, \tag{1.4.6}$$

and work with the new variable u. The problem is then rewritten as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < \pi. \tag{1.4.7}$$

The boundary conditions are now

$$u(0,t) = 0, \qquad u(\pi,t) = 0, \qquad t > 0, \tag{1.4.8}$$

and the initial condition is

$$u(x,0) = \frac{-x}{\pi}. \tag{1.4.9}$$

In Figure 1.1 we show the solutions at various times t. The solution approaches the steady state $u = 0$ or $v = x/\pi$ quickly.

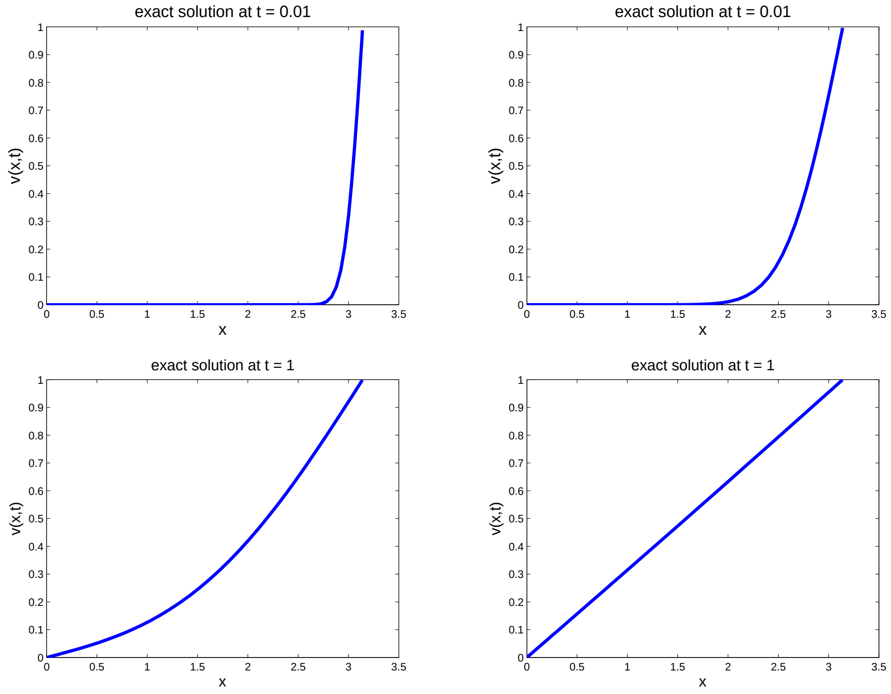


Figure 1.1: Solution of problem (1.4.1)–(1.4.3) computed by (1.4.4)–(1.4.5)

1.5 Thesis layout

The outline of this thesis is as follows.

In Chapter 2, we introduce polynomial interpolation, spectral differentiation, and the concept of a differentiation matrix. We also mention the MATLAB package DMSUITE, which is used to construct differentiation matrices.

Next, in Chapter 3, we introduce the time integration methods. In Section 3.1, we discuss the method-of-lines based on MATLAB's **ODE15s**, where we discuss the concept of stiffness as well as the mathematical background of the solver. Then we also introduce the well-known Crank-Nicolson method. This is followed by the Laplace inversion method in Section 3.2.

In Chapter 4, we perform numerical experiments with all these methods and discuss the convergence rates as well as the computational cost. The numerical results are presented by means of tables and figures.

In Chapter 5 we apply the Laplace inversion method to solve the axial dispersion model encountered in process engineering (Section 5.1), and the solution of a two dimension parabolic PDE (Section 5.2).

Finally, Chapter 6 summarizes our conclusions concerning the four methods. Subsequently, some issues for future research are proposed.

Chapter 2

Space discretization: Spectral Methods

In recent years, spectral methods have emerged as a viable alternative to finite elements and finite difference methods for the numerical solution of partial differential equations. This is mainly because of the higher accuracy it offers compared to the well-established finites differences and finite element methods. In areas such as turbulence flow modeling, numerical weather prediction, and seismic exploration, they have become popular for their accuracy and efficiency for numerical calculations [18, p. 127]. A survey of spectral methods in those areas is given in [18, Chapter 8] and [19, Section 15]. In this chapter, we present a spectral method based on Chebyshev polynomial interpolation [20] for solving the model problem given in Section 1.4.

Spectral methods use global representations of high degree over the entire domain. By contrast, methods such as finite elements or finite differences divide the domain into subintervals and use local polynomials of low degree.

For smooth solutions the results using spectral methods is of a degree of accuracy that local approximation methods cannot match. For such solutions spectral methods can often achieve an exponential convergence rate $O(e^{-cN})$, in contrast to the algebraic convergence rate $O(N^{-2})$ or $O(N^{-4})$ for finite differences or finite elements. Here N is the number of unknowns due to the discretization.

Nevertheless, in practice using spectral methods for boundary value problems may be troublesome. The presence of boundaries often introduces stability conditions that are both highly restrictive and often difficult to analyse. Thus, for a first order PDE one can get restrictive stability conditions of the form $\Delta t \leq O(N^{-2})$ for spectral methods where we would get $O(N^{-1})$ for finite differences or finite elements. The disparity increases for a second order PDE from $O(N^{-4})$ to $O(N^{-2})$ [1].

Moreover, matrices in spectral methods are neither sparse nor symmetric, in contrast to the situation in finite differences or finite elements where the sparsity structure of the matrices simplifies the computation. However, the point is that N is usually much smaller for spectral methods than for finite differences and finite elements. Therefore, a restriction of $O(N_1^{-4})$ can be more efficient compared to $O(N_2^{-2})$ if $N_2 \gg N_1$.

The implementation of spectral methods can be divided into three categories, namely the Galerkin, tau and the collocation (or pseudospectral) methods. The first two use the expansion coefficients of the global approximation and the latter can be viewed as a method of finding numerical approximations to derivatives at collocations points. In a manner similar to finite differences or finite elements methods the equation to be solved is satisfied in space at the collocations points. A comparison of these three methods is given in [18]. For its simplicity we restrict our discussion of spectral methods to the collocation method.

This chapter is organized as follows. In the first section, we shall briefly introduce polynomial interpolation, followed by the derivation of the differentiation matrices based on the spectral collocation method.

2.1 Polynomial interpolation

The spectral process involves seeking the solution to a differential equation by polynomial interpolation. In order to review the formulas of polynomial interpolation, we consider interpolating an arbitrary function $f(x)$ at $N+1$ distinct nodes $\{x_k\}_{k=0}^N$ in $[-1,1]$.

Definition 2.1.1 *Given a set $\{x_j\}_{j=0}^N$ of grid points, an interpolating approximation to a function $f(x)$ is a polynomial $f_N(x)$ of degree N, determined by the requirement that the interpolant agrees with $f(x)$ at the set $\{x_j\}_{j=0}^N$ of interpolation points, i.e.,*

$$f_N(x_i) = f(x_i), \qquad i = 0, 1, ..., N.$$

We define $L_k(x)$ as the Lagrange polynomial of degree N,

$$L_k(x) = \prod_{\substack{j=0 \\ j\neq k}}^{N} \frac{x - x_j}{x_k - x_j}, \qquad k = 0, 1, ..., N.$$

Note that $L_k(x)$ satisfies $L_j(x_k) = \delta_{jk}$, where δ_{jk} is the Kronecker delta-symbol. The interpolation polynomial, $f_N(x)$ is then given by

$$f_N(x) = \sum_{k=0}^{N} f(x_k) L_k(x). \tag{2.1.1}$$

From approximation theory we are aware of the fact that the set of points $\{x_k\}_{k=0}^N$ should not be chosen arbitrarily. For instance, polynomial interpolation on equally spaced points often fails because of divergence near the endpoints. This behaviour is known as the Runge phenomenon [21, p. 83].

In contrast, polynomial interpolation on unevenly spaced points which concentrate the nodes points toward the end of the domain is more efficient [21, p. 85]. This is the feature of grid points associated with the roots of the Jacobi polynomials; they have a nonuniform distribution in the interval $[-1,1]$ with a node density per unit length of [1]

$$\mu \approx \frac{N}{\sqrt{1-x^2}} \qquad \text{as} \qquad N \to \infty.$$

Examples of such points are:

- Chebyshev zeros: $x_j = \cos\left(\frac{2j+1}{2(N+1)}\pi\right), \qquad j = 0, ..., N.$
- Chebyshev extrema: $x_j = \cos\left(\frac{j\pi}{N}\right), \qquad j = 0, ..., N.$
- Legendre zeros: $x_j = j$th zero of $P_{N+1}(x), \qquad j = 0, ..., N.$

Here P_{N+1} is the Legendre polynomial of degree $N+1$. The Chebyshev points are the zeros and extrema of the Chebyshev polynomials T_{N+1} and T_N respectively. The Chebyshev extreme points are often described as the projection onto the interval $[-1,1]$ of the roots of unity along the unit circle $|z| = 1$ in the complex plane [20, p. 43].

For Chebyshev points fast algorithms such as the Fast Fourier Transform (FFT) exist for the implementation of the differentiation process at a cost of $O(N \log N)$. Legendre grids have some advantages because of its connection with Gauss quadrature but for the present thesis we shall only use the Chebyshev extreme points for its simplicity.

Finally, note that the canonical interval of the Chebyshev points is $[-1,1]$. Any problem posed on an arbitrary interval $[a,b]$ can be converted to $[-1,1]$ by the linear transformation $x \longleftrightarrow (1/2)((b-a)x + (b+a))$.

2.2 Differentiation matrices

In this section we shall use the Chebyshev grid introduced in the previous section to construct differentiation matrices. Then we shall use these matrices to derive an approximate matrix form of the model equation (1.4.7)–(1.4.9).

Associated with the interpolant polynomial $f_N(x)$ is the concept of collocation derivative. This is the derivative of $f_N(x)$ at the collocation points $\{x_k\}_{k=0}^N$. Thus the order m collocation derivative of (2.1.1) is

$$\frac{d^m f_N(x)}{dx^m} = \sum_{k=0}^{N} f(x_k) \frac{d^m L_k(x)}{dx^m}. \tag{2.2.1}$$

Evaluation at the nodes yields

$$\frac{d^m f_N(x_j)}{dx^m} = \sum_{k=0}^{N} f(x_k) \frac{d^m L_k(x_j)}{dx^m}, \qquad j = 0, ..., N. \tag{2.2.2}$$

Since differentiation is a linear process, (2.2.2) can be represented by the matrix formula

$$\boldsymbol{f}_N^{\boldsymbol{(m)}} = D_N^{(m)} \boldsymbol{f}_N, \tag{2.2.3}$$

where

$$\boldsymbol{f}_N = \begin{bmatrix} f_N(x_0) \\ \vdots \\ f_N(x_N) \end{bmatrix}, \quad \boldsymbol{f}_N^{\boldsymbol{(m)}} = \begin{bmatrix} f_N^{(m)}(x_0) \\ \vdots \\ f_N^{(m)}(x_N) \end{bmatrix},$$

and $D_N^{(m)}$ is the $(N+1) \times (N+1)$ differentiation matrix of order m with entries

$$(D_N^{(m)})_{j,k} = L_k^{(m)}(x_j), \qquad j,k = 0,...,N. \tag{2.2.4}$$

The computation of these differentiation matrices for an arbitrary order m has been considered in [22; 20; 23]. Following the approach of [24], Weideman and Reddy [23] developed a MATLAB algorithm that computes the Chebyshev grid points as well as the differentiation matrix of an arbitrary order m. This algorithm is implemented in the DMSUITE package [23]. The suite contains a function `chebdif` that computes the extreme points of the Chebyshev polynomial $T_N(x)$ and the differentiation matrix $D_N^{(m)}$. The code takes as input the size of the differentiation matrix N and the highest derivative order m and produces matrices $D_N^{(\ell)}$ of order $\ell = 1, 2, ..., m$. The next Theorem gives the formulas for the computation of the entries of $D_N^{(1)}$.

Theorem 2.2.1 *For any $N \geq 1$, let $i, j = 0, 1, ...N$. Then the entries of $D_N^{(1)}$ are given by*

$$(D_N^{(1)})_{00} = \frac{2N^2+1}{6}, \qquad (D_N^{(1)})_{NN} = \frac{2N^2+1}{6}, \tag{2.2.5}$$

$$(D_N^{(1)})_{jj} = \frac{-x_j}{2(1-x_j^2)}, \qquad j = 1, ..., N-1, \tag{2.2.6}$$

$$(D_N^{(1)})_{ij} = \frac{c_i}{c_j} \frac{(-1)^{i+j}}{(x_i - x_j)}, \qquad i \neq j, \qquad i,j = 0,...,N, \tag{2.2.7}$$

where

$$c_i = \begin{cases} 2, & i = 0 \textit{ or } N \\ 1, & \textit{otherwise.} \end{cases}$$

Proof: see Exercises 6.1 and 6.2 in [20, p. 58-59].

Higher order derivatives are evaluated by recurrences [24; 23], at a cost of $O(N^2)$ operations. This turns out to be cost effective compared to $O(N^3)$ if higher derivatives $D_N^{(m)}$ are obtained by taking powers of the first derivative $D_N^{(1)}$ [23].

The code is called from the command

```
»[x,DM] = chebdif(N,M);
```

The inputs N and M are respectively the size of the matrix and the highest derivative needed. The output is the column vector

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ \vdots \\ x_N \end{bmatrix} \tag{2.2.8}$$

with $x_0 = 1$ and $x_N = -1$. The trifold array `DM` contains the $(N+1) \times (N+1)$ matrices $D_N^{(m)}$, where $1 \leq m \leq M$.

As an example, we consider the differentiation of the function $f(x) = e^x$. For $N = 4$, the function `chebdif` computes the derivatives $f^{(1)}(x)$, $f^{(2)}(x)$, and $f^{(3)}(x)$, which are all equal to e^x, at the Chebyshev points x_j for $j = 0, ..., 4$ as follows.

```
N = 4; M = 3; [x,DM] = chebdif(N,M);

x =
   1.000000000000000
   0.707106781186547
                   0
  -0.707106781186547
  -1.000000000000000

y = exp(x);

DM(:,:,1)*y
   2.707988519849198
   2.032992745232904
   0.995682088901412
   0.496928746797088
   0.360490286407965

DM(:,:,2)*y
   2.601215133535465
   2.022972021706034
   0.998573422540361
   0.499701905915452
   0.446985876626792

DM(:,:,3)*y
   2.128168793535845
   1.820322155977870
   1.077114628454342
   0.333907100930835
```

```
0.026060463372817
```

The error in the spectral method is shown in Figure 2.1. The figure shows a graph of $f(x)$ alongside a plot of the error in $f^{(1)}(x)$, $f^{(2)}(x)$ and $f^{(3)}(x)$ respectively. With $N = 15$, we have about 13, 11 and 10 digits of accuracy for the computation of $f^{(1)}(x)$, $f^{(2)}(x)$ and $f^{(3)}(x)$ respectively.

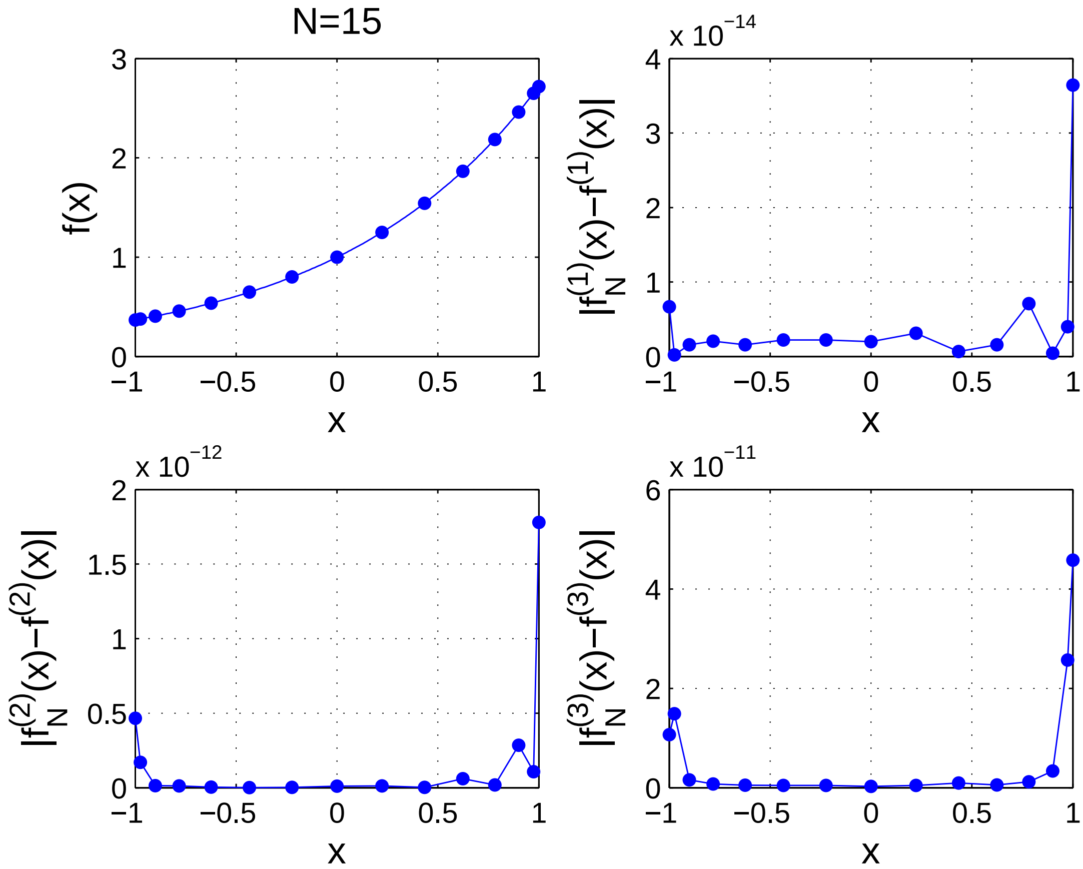


Figure 2.1: Chebyshev approximation to derivatives of $f(x) = e^x$.

Having defined the concept of differentiation matrix, we shall use it to approximate and solve PDEs. We consider the model problem (1.4.7)–(1.4.9)

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < \pi, \tag{2.2.9}$$

with boundary conditions

$$u(0, t) = 0, \qquad u(\pi, t) = 0, \qquad t > 0, \tag{2.2.10}$$

and the initial condition

$$u(x,0) = -x/\pi. \tag{2.2.11}$$

To convert the interval $[0,\pi]$ to $[-1,1]$, we apply the linear transformation

$$x \longleftrightarrow \frac{\pi}{2}\left(x+1\right).$$

The equations (2.2.9)–(2.2.11) become

$$\frac{\partial u}{\partial t} = \frac{4}{\pi^2}\frac{\partial^2 u}{\partial x^2}, \qquad -1 < x < 1, \tag{2.2.12}$$

with boundary conditions

$$u(-1,t) = 0, \qquad u(1,t) = 0, \qquad t > 0, \tag{2.2.13}$$

and initial condition

$$u(x,0) = -\frac{1}{2}\left(x+1\right). \tag{2.2.14}$$

We need to approximate the second partial derivative by the second order differentiation matrix $D_N^{(2)}$. Assume $u_N(x,t)$ is the interpolation polynomial in x of the unknown solution $u(x,t)$. Then, from subsection 2.2, equation (2.2.12) can be approximated in matrix form as

$$\frac{\partial \boldsymbol{u}_N}{\partial t} = \frac{4}{\pi^2} D_N^{(2)} \boldsymbol{u}_N \qquad j = 0,...,N, \tag{2.2.15}$$

where

$$\boldsymbol{u}_N = \begin{bmatrix} u_N(x_0,t) \\ \vdots \\ u_N(x_N,t) \end{bmatrix}.$$

From the boundary conditions (2.2.13) we have $u_N(x_0,t) = u_N(x_N,t) = 0$. This means that the first and last rows of the $D_N^{(2)}$ have no effect since they are not required and the first and last columns are multiplied by zero. This amounts to suppressing the first and last column as well as the first and last rows of $D_N^{(2)}$. As a result of this, the $(N-1)\times(N-1)$ differentiation matrix $D_N^{(2)}$ in (2.2.15) is obtained by removing the first and last rows and columns respectively.

Since MATLAB does not have a zero index, the first row and column of $D_N^{(2)}$ start with index $j = k = 1$, and consequently the last row and column are indexed $j = k = N+1$. Therefore removing the first and last rows and columns in MATLAB notation is achieved by

$$D_N^{(2)}(2:N,\, 2:N, 2). \tag{2.2.16}$$

The problem (1.4.7)−(1.4.9) is therefore approximated by the linear system of ODEs

$$\boldsymbol{u}_t = \frac{4}{\pi^2} D_N^{(2)} \boldsymbol{u}, \qquad \boldsymbol{u}_0 = -\frac{1}{2}\left(\boldsymbol{x} + \boldsymbol{x}_1\right), \tag{2.2.17}$$

and

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}, \quad \boldsymbol{x}_1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{2.2.18}$$

The approximation to $u(t)$ is represented by the $(N-1)\times 1$ vector $\boldsymbol{u} = \boldsymbol{u}(t)$. The column vector $\boldsymbol{u}_0$ is the representation of the initial condition (2.2.14).

It is known that the analytical solution to the semi-discrete equation (2.2.17) is

$$\boldsymbol{u}(t) = \exp(\alpha D_N^{(2)} t)\boldsymbol{u}_0, \qquad \alpha = \frac{4}{\pi^2},$$

which requires the computation of the matrix exponential. However, computing the matrix exponential of a large matrix can be a computational challenge, and therefore this formula for solving the problem is not used much in practice [25].

In the next chapter, we solve the system (2.2.17) using two time-integrating techniques, namely the method-of-lines in Section 3.1 and the Laplace transform in Section 3.2.

Chapter 3

Time integration

In this chapter, numerical methods used to solve the model problem (2.2.17) are discussed. We consider three time integrating methods, namely two versions of the method-of-lines (based on MATLAB's **ODE15s** and the Crank-Nicolson method) and the Laplace inversion method based on the trapezoidal rule.

3.1 The Method-of-lines

The method-of-lines is a well-known method for the numerical solution of time dependent PDEs. The process is divided into the following two steps. Firstly, discretize the PDE with respect to space, thereby generating a system of ODEs in time, such as (2.2.17) in the previous section. Secondly, solve the system by some discrete method in time [26], such as Runge-Kutta or linear multistep methods, or a software package that has been developed for ODEs.

The method is quite powerful and is widely used in conjunction with a wide range of high quality ODE integrators. As an example of such methods we consider, in this section, the MATLAB ODE solver **ODE15s** and the Crank-Nicolson method.

3.1.1 MATLAB ODE15s

MATLAB ODE solvers are a collection of codes for solving initial value problems. The solvers implement a variety of methods and are distinguished from each other by their ability to solve stiff problems as well as their available order. The solvers for stiff problems are **ODE23s**, **ODE15s**, whereas **ODE23**, **ODE45** and **ODE113** are used for non-stiff problems. We will consider the stiff solver **ODE15s** for the solution of the semi-discrete equation (2.2.17).

Stiffness is a property of an ODE that leads to instability when an explicit method is used, unless considerably small time steps are taken [16]. Implicit methods, in particular the backward differentiation methods, perform better than explicit ones [16; 27].

Consider a linear system of ODEs

$$\boldsymbol{u}_t = D\boldsymbol{u} + \boldsymbol{b}, \tag{3.1.1}$$

where D is a square matrix of order $(N-1)$. The Jacobian J of the system is given by the matrix D itself.

The eigenvalues of the Jacobian matrix characterize the stability of the system. In general, the system (3.1.1) is called stiff if the eigenvalues of J differ greatly in magnitude [28; 27]. The following definition occurs in the literature [28; 16].

Definition 3.1.1 *Assume that all eigenvalues $\lambda_1, ..., \lambda_N$ of the Jacobian matrix J are all real and negative and ordered as*

$$\lambda_N \leq \lambda_{N-1} \leq ... \leq \lambda_2 \leq \lambda_1 < 0. \tag{3.1.2}$$

Then the system (3.1.1) is said to be stiff if the stiffness ratio s, defined by

$$s = \frac{|\lambda_N|}{|\lambda_1|},$$

is such that

$$s \gg 1, \quad i.e., \quad |\lambda_N| \gg |\lambda_1|.$$

Taking the semi-discrete model equation (2.2.17), note that its Jacobian is the differentiation matrix $D_N^{(2)}$ itself. From [29; 1], we know that all eigenvalues of $D_N^{(2)}$ are distinct and lie on the negative real axis. This means that we can apply Definition 3.1.1 to measure its stiffness ratio.

The stiffness ratio s_1 of $D_N^{(2)}$, computed numerically, is shown in Table 3.1 for different values of N. Large values of the ratio s_1 indicate that the equation (2.2.17) is stiff. These numerical values agree well, in order of magnitude, with the theoretical estimate of the ratio s_2 obtained in [1]. In that paper, the authors derived the theoretical formulas to estimate the two extreme eigenvalues as

$$\lambda_1 \approx -\pi^2/2^2 \qquad \text{and} \qquad \lambda_N \approx -\sqrt{\frac{11}{4725}} N^4.$$

Effective methods for solving stiff problems have large stability regions [16]. The particular shape of the stability region required will depend on the problem to be solved. But, what is important is for the stability region to include a large part of the left complex plane, thus including the spectrum of the Jacobian matrix $D_N^{(2)}$. Our discussion on stability will be restricted to the following definition:

N	s_1	s_2
10	1.36×10^2	1.95×10^2
20	2.45×10^3	3.13×10^3
40	4.46×10^4	5.01×10^4
60	2.33×10^5	2.53×10^5
80	7.48×10^5	8.01×10^5

Table 3.1: Stiffness ratio of (2.2.17) for $N \times N$ matrix $D_N^{(2)}$, s_1 is the ratio obtained numerically whereas s_2 is the theoretical estimate derived [1].

Definition 3.1.2 *A linear multistep method is A-stable if the stability region includes the entire left half-plane $Re\,\lambda < 0$. It is $A(\alpha)$-stable if the stability region contains the infinite sector $|arg\,\lambda - \pi| < \alpha$ for some α in $[0,\, \pi/2]$.*

A class of $A(\alpha)$-stable method is known as backward differentiation formulas. Most of the software packages in use for stiff problems are based on these formulas.

3.1.1.1 Backward differentiation formulas

We start this subsection with the definition of the backward difference operator

$$\nabla u^n = u^n - u^{n-1}, \tag{3.1.3}$$

and the $(j)^{th}$-order backward difference operator

$$\nabla^j u^n = \sum_{k=0}^{j} (-1)^k \binom{j}{k} u^{n-k}.$$

The backward differentiation formula of order k, BDFk, approximate the solution $\boldsymbol{u}(t)$ of (3.1.1) by polynomial interpolation. The interpolation polynomial $\boldsymbol{p}_k(t)$ of degree k satisfies the solution $\boldsymbol{u}(t)$ at step $t_{n+1} = t_n + h$, with a constant step size h, using the previously computed solution values $\boldsymbol{u}^{n-k}$,...,$\boldsymbol{u}^{n-1}$, $\boldsymbol{u}^n$. The formula is defined by

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j \boldsymbol{u}^{n+1} = h\left(D\boldsymbol{u}^{n+1} + \boldsymbol{b}\right). \tag{3.1.4}$$

At each time step a linear system involving the coefficient matrix D has to be solved [28, p. 70]. This can be done efficiently with a LU decomposition, which is computed prior to the time stepping. Details are given in the next subsection. The local discretization error introduced at each step is given by the term [16]

$$\frac{1}{k+1} h^{k+1} \boldsymbol{u}^{n+1}_{(k+1)}, \tag{3.1.5}$$

Note that in (3.1.5) the subscript $(k+1)$ is a derivative w.r.t. t, whereas the superscript indicates a time level, i.e.,

$$\frac{d^k \boldsymbol{u}^n}{dt^k} = \boldsymbol{u}^n_{(k)}. \tag{3.1.6}$$

The relationship of the difference operator ∇ with respect to derivatives is given by

$$u^{n+1}_{(k+1)} \approx \frac{\nabla^{k+1} u^{n+1}}{h^{k+1}}. \tag{3.1.7}$$

From (3.1.7), the approximation (3.1.5) can be rewritten as

$$\frac{1}{k+1} \nabla^{k+1} \boldsymbol{u}^{n+1}. \tag{3.1.8}$$

In practice, the implementation of the BDF formula (3.1.4) is based on a quasi-constant step-size, which means that the step-size h is held constant to the extent possible [16; 28, p. 64]. When it appears necessary to change the step size to a new step-size h_{new}, previous values computed at a spacing of h are interpolated to obtain values at h_{new}. As a result, **ODE15s** continuously controls the accuracy of the solution at each time step and adaptively changes the time step to maintain a consistent level of accuracy. Larger time steps are used for slowly varying regions and smaller time steps are taken for rapidly varying regions.

In addition to adaptive step size, the BDFs codes have formulas of varying order. In particular, BDFs are A-stable up to and including order 2 and $A(\alpha)$-stable, for some $\alpha \in (0, \pi/2)$, up to order 6 [28, p. 68]. As the order increases, the stability region decreases with the angle α. As a consequence, the stability of the formula (3.1.4) worsens [28, p. 68]. In the next section, we describe a new family of formulas based on BDFs called Numerical Differentiation Formulas [16] that present some advantages over the BDFs.

3.1.1.2 Numerical differentiation formulas

Numerical differentiation formulas (NDFs) [16] were developed in an attempt to improve the stability of higher order BDFs but at reasonable computational cost. The NDF formula is given by the equation

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j \boldsymbol{u}^{n+1} = h\left(D\boldsymbol{u}^{n+1} + \boldsymbol{b}\right) + \kappa\gamma_k \left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^0\right), \tag{3.1.9}$$

where $\boldsymbol{u}^0 = \boldsymbol{u}_0$, κ is a scalar parameter and γ_k is given by

$$\gamma_k = \sum_{j=1}^{k} \frac{1}{j}.$$

For $\kappa = 0$, this formula reduces to the backward difference formula (3.1.4).

To derive the local error at each step, Shampine [16] considered the identity

$$\boldsymbol{u}^{n+1} - \boldsymbol{u}^0 = \nabla^{k+1}\boldsymbol{u}^{n+1},$$

and from the local discretization error (3.1.8) deduced

$$\left(\kappa\gamma_k + \frac{1}{k+1}\right)\nabla^{k+1}\boldsymbol{u}^{n+1}. \tag{3.1.10}$$

From (3.1.10) we deduce

$$\left(\kappa\gamma_k + \frac{1}{k+1}\right)h^{k+1}\boldsymbol{u}^{(n+1)}_{(k+1)}. \tag{3.1.11}$$

In the light of (3.1.11) and from (3.1.5), the leading error term of both BDFs and NDFs are of $O(h^{k+1})$. Thus NDFs have a smaller error constant than the BDFs if the inequality

$$\left|\kappa\gamma_k + \frac{1}{k+1}\right| < \frac{1}{k+1}, \qquad \text{i.e.,} \qquad \frac{-2}{k+1} < \kappa < 0,$$

holds. In [16; 30] the choice of the parameter κ is determined by two criteria, stability and efficiency. Klopfenstein [30] computed numerically κ values for orders 1 to 6 but only values for orders 1 and 2 could preserve both the accuracy and stability. Using a different approach to Klopfenstein, Shampine and Reichelt [16] computed successfully the best estimate of κ for orders 3 up to 5. These values can improve the efficiency of the NDFs by approximately 26% for the orders 1, 2, 3 and 12% for the fourth order [16].

The **ODE15s** solver is based on the NDFs of order 1 to 5 [16]. The order 5 NDF is used as default method in the code. As an option, it uses the BDFs. For stiff problems, the solver uses sufficiently small step-sizes in stiff regions and automatically increases the step-size in non-stiff regions. When the error becomes larger than a specified tolerance, the step-size is a failure. The process is repeated until a new step-size with error estimate less than the given error tolerance is obtained [16]. Although the code accurately approximates the solution, the repeated process and the choice of small step for stability comes at a computational cost [28, p. 66]. Despite this drawback the **ODE15s** solver exhibits efficient results and is cost effective on stiff problems compared to non-stiff solvers such as **ODE45**.

MATLAB's **ODE15s** is relatively easy to implement, and systems of ODEs can be solved in few lines of code. The minimum input includes a function handle for the time derivative, a time range, and an initial value. Optionally, the user can specify the desired relative and absolute errors through the command *abstol* and *reltol* respectively.

In the next section we discuss the second method-of-lines used in this thesis, namely the Crank-Nicolson method.

3.1.2 The Crank-Nicolson Method

An alternative method-of-lines is the well known Crank-Nicolson method. Let $h = \frac{t}{M}$, be a given step-size, where M is the number of steps, and t the final time. Then the Crank-Nicolson approximation of the equation (3.1.1) is

$$\left(I - \frac{h}{2}D_N^{(2)}\right)\boldsymbol{u}^{n+1} = \left(I + \frac{h}{2}D_N^{(2)}\right)\boldsymbol{u}^n + h\boldsymbol{b}. \tag{3.1.12}$$

This scheme is implicit, unconditionally stable, and is of second order of accuracy.

Let $A = \left(I - \frac{h}{2}D_N^{(2)}\right)$ and $B = \left(I + \frac{h}{2}D_N^{(2)}\right)$, then equation (3.1.12) becomes

$$A\boldsymbol{u}^{n+1} = B\boldsymbol{u}^n + h\boldsymbol{b}. \tag{3.1.13}$$

Thus, each step of (3.1.13) requires the solution of a system of linear equations involving the matrix A.

When using spectral methods the matrices A and B are dense, and the solution of the equation (3.1.13) is obtained at the cost of $O(N^3)$ operations at each step n. This can become inefficient as N increases.

To reduce the computational cost, we consider a transformation of A to some triangular system which is easier to solve. Such systems can be solved by forward or backward substitution [31, p. 88–89]. This process is referred to as LU factorization. Since A does not depend on the step index n, one can compute an LU factorization of this matrix once, beforehand, and apply it in all steps (3.1.13) to obtain $\boldsymbol{u}^{n+1}$.

The decomposition of A to an LU form yields

$$A = LU, \tag{3.1.14}$$

where U is an upper triangular matrix and L is a lower triangular matrix obtained by Gaussian elimination [31, p. 94]. Improved stability of the Gaussian elimination method can be achieved by introducing row interchanges during the elimination process. This strategy, known as *partial pivoting*, corresponds to the multiplication on the left of A in (3.1.14) by a permutation matrix P [31, p. 109–121]. The LU form with partial pivoting becomes

$$A = PLU. \tag{3.1.15}$$

The solution of the recurrence relation (3.1.13) is then obtained by solving the triangular systems [31, p. 94]

$$L\boldsymbol{y}^n = P^T(B\boldsymbol{u^n}) + hP^T\boldsymbol{b}, \tag{3.1.16}$$

and

$$U\boldsymbol{u^{n+1}} = \boldsymbol{y}^n, \tag{3.1.17}$$

in sequence (note that we have used $P^{-1} = P^T$). Step (3.1.16) involves forward substitution, and (3.1.17) a back substitution, both of which cost $O(N^2)$ operations. This is in contrast to the original $O(N^3)$ as in (3.1.13) where a full system is solved at each step.

The performance of the LU factorization as applied to the Crank-Nicolson method is shown in Figure 3.1. In this figure, we have plotted the CPU time for various matrices of different sizes N when solving the equation (3.1.13) with and without the LU factorization (3.1.15). From Figure 3.1, we observe an increase of the computational time (dotted curve) as the full matrices A and B in (3.1.13) become large. However, the application of the LU decomposition substantially reduces the CPU time (circled curve).

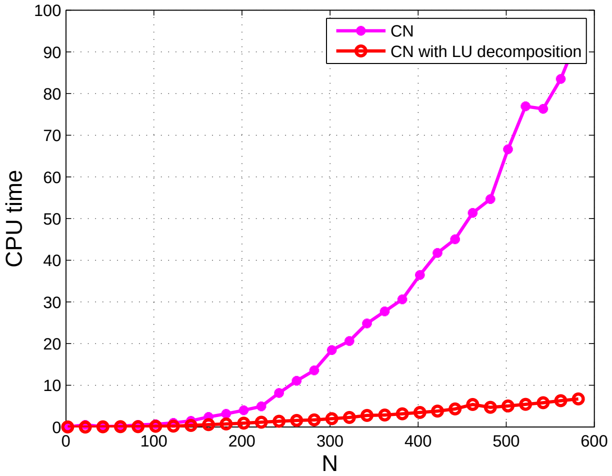


Figure 3.1: CPU time (in seconds) of one step of the Crank-Nicolson (3.1.12) with and without the LU factorization.

3.2 The Laplace inversion method

In this section we consider the Laplace inversion formula as an alternative to the methods of Section 3.1.

Recall from Chapter 1 that given a Laplace transform $U(z)$, the reconstruction of the original function $u(t)$ can be expressed by the Bromwich integral

$$u(t) = \frac{1}{2\pi i}\int_{\sigma - i\infty}^{\sigma + i\infty} e^{zt}U(z)dz, \qquad t > 0, \tag{3.2.1}$$

with $\sigma > \sigma_0$ and σ_0 the convergence abscissa of $U(z)$.

For numerical evaluation of $u(t)$ and from the Cauchy's integral formula, the Bromwich line is deformed to an equivalent contour Γ that starts and ends in the left half-plane. On the new contour Γ, we have

$$u(t) = \frac{1}{2\pi i}\int_{\Gamma} e^{zt}U(z)dz, \qquad t > 0. \tag{3.2.2}$$

In the next section, we discuss a family of hyperbolic type as contours of integration.

3.2.1 The hyperbolic contour

Different contours such as the parabola, hyperbola and a cotangent contour have been proposed in [32; 3; 33]. Optimal parameters for these contours have been derived in [4; 34]. For a pure parabolic problem, these three contours have a fast convergence rate of order roughly $O(10^{-M})$, where M is the number of sample points in the quadrature rule. We shall only consider the hyperbola as the integration contour.

Following [32; 3], we define the family of hyperbolic contours by

$$z = \mu\left(1 + \sin\left(i\ell - \alpha\right)\right), \qquad \ell \in \mathbb{R}, \tag{3.2.3}$$

where the real parameters $\mu > 0$ and $0 < \alpha < \pi/2$ determine the geometry of the contour. The positive parameter μ controls the width of the contour while α determines the geometric shape, i.e., the asymptotic angle.

The mapping z transforms the horizontal straight line $\ell \in \mathbb{R}$ into the left branch of the hyperbola

$$\left(\frac{\mu - x}{\sin\left(\alpha\right)}\right)^2 - \left(\frac{y}{\cos(\alpha)}\right)^2 = \mu^2, \qquad x = \operatorname{Re} z \quad \text{and} \quad y = \operatorname{Re} z.$$

The asymptotes of this hyperbola make angles $\pm\left(\pi/2 - \alpha\right)$ with the real axis. A typical contour is illustrated in Figure 3.2 below.

On the contour (3.2.3) the inversion formula (3.2.2) becomes

$$u(t) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} e^{z(\ell)t}U\left(z(\ell)\right)z'(\ell)d\ell, \tag{3.2.4}$$

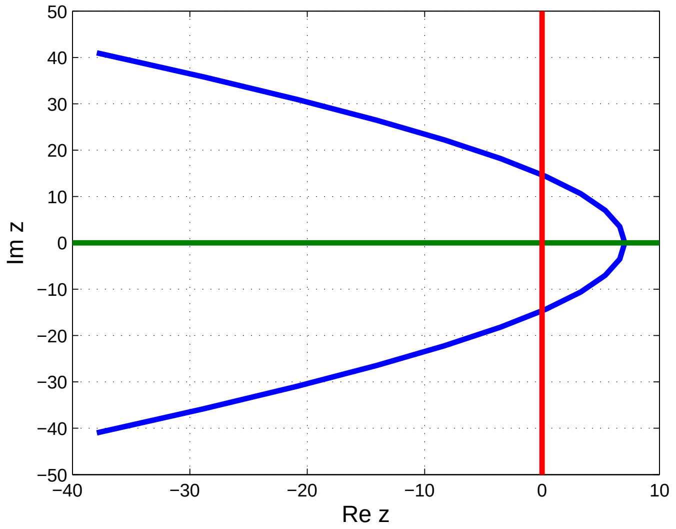


Figure 3.2: Hyperbolic contour (3.2.3).

where

$$z'(\ell) = \mu i \cos (i\ell - \alpha) .$$

In Subsection 3.2.5, we determine formulas for the optimal parameters μ and α.

3.2.2 Approximation by the trapezoidal rule

In Section 1.2, we saw that the trapezoidal rule converges exponentially on an interval $[a,\, b]$ for a smooth $(b-a)$-periodic function. In this subsection, we shall see that the trapezoidal rule is also suitable for improper integrals over $(-\infty,\, \infty)$ where an exponential convergence rate can be achieved.

Let us consider

$$I(f) = \int_{-\infty}^{\infty} f(x)dx;$$

in this case, the formula (1.2.1) becomes

$$T(f) = h \sum_{j=-\infty}^{\infty} f\left(x_j\right), \qquad x_j = jh. \tag{3.2.5}$$

For practical purposes, the infinite sum is truncated to a finite sum. The application of this formula is efficient only when the function $f(x)$ decays sufficiently fast. For an analytic function $f(x)$ defined on $(-\infty,\, \infty)$, a theorem similar to Theorem 1.2.1 in Chapter 1, which gives an exponential convergence estimate of the trapezoidal rule, is stated as follows

Theorem 3.2.1 *Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be an analytic function. Then there exists a strip $\mathbb{R} \times (-d,\, d)$ in the complex plane with $d > 0$ such that f can be extended to a complex analytic function $f : \mathbb{R} \times (-d,\, d) \longrightarrow \mathbb{C}$. Furthermore, the error for the trapezoidal rule is given by*

$$E_d = Re\left\{\int_{-\infty+i\sigma}^{\infty+i\sigma}\left(1 - i\cot\frac{\pi z}{h}\right)f(z)dz\right\},$$

where $0 < \sigma < d$. Moreover, it is bounded by

$$|E_d| \leq \frac{2}{\exp\left(\frac{2\pi d}{h}\right) - 1}\int_{-\infty+i\sigma}^{\infty+i\sigma}|f(z)|dz.$$

Proof: see [15].

From this theorem we see that for an analytic function defined on $(-\infty,\, \infty)$, the error is of exponential order. For further discussion of the convergence of the trapezoidal rule, we refer to [15].

Define $\ell_k = kh$, for $h > 0$, then the trapezoidal rule applied to (3.2.4) gives

$$u(t) \approx \frac{h}{2\pi i}\sum_{k=-\infty}^{\infty} e^{z(\ell_k)t}U\left(z(\ell_k)\right)z'(\ell_k). \qquad (3.2.6)$$

Because of the exponential factor $e^{z(\ell_k)t}$, the terms in the sum decrease exponentially as $k \rightarrow \infty$. In this case one commits an exponentially small error by truncating at a finite integer $k = M$.

We redefine the quadrature points as

$$\ell_k = \left(k + \frac{1}{2}\right)h, \qquad (3.2.7)$$

which we consider in the rest of this thesis for simplicity. With the quadrature points (3.2.7), the trapezoidal rule becomes the midpoint rule, an equally accurate method. In the next subsections we apply the inversion method based on the midpoint rule to solve a scalar problem as well as the matrix problem (2.2.17).

3.2.3 Scalar problem

We consider the scalar analogue of (2.2.17)

$$\frac{du}{dt} = \lambda u, \qquad \lambda < 0. \qquad (3.2.8)$$

A straightforward application of the Laplace transform (1.1.1) to (3.2.8) gives

$$U(z) = \frac{u(0)}{z - \lambda}, \qquad (3.2.9)$$

and the inversion formula (3.2.4) on the contour (3.2.3) yields

$$u(t) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} \frac{e^{z(\ell)t}u(0)}{z(\ell)-\lambda} z'(\ell)d\ell. \tag{3.2.10}$$

The approximation to (3.2.10) becomes

$$u(t) \approx \frac{h}{2\pi i}\left(u(0)\sum_{k=-\infty}^{\infty} e^{z(\ell_k)t}\left(z(\ell_k)-\lambda\right)^{-1} z'(\ell_k)\right). \tag{3.2.11}$$

For practical purposes, the infinite sum (3.2.11) has to be truncated, say to $2M$ points. And if symmetry is used, the above equation is equivalent to

$$u_M(t) = \frac{h}{\pi}\mathrm{Im}\left(u(0)\sum_{k=0}^{M-1} e^{z_k t}\left(z_k-\lambda\right)^{-1} z_k'\right). \tag{3.2.12}$$

where $z_k = z(\ell_k)$ and $z_k' = z'(\ell_k)$.

In the next section we shall show that the scalar λ represents an eigenvalue of the differentiation matrix D which is equal to the singularities of the Laplace transform.

3.2.4 Matrix problem

We now turn to the matrix problem (2.2.17)

$$\boldsymbol{u}_t = \frac{4}{\pi^2}D_N^{(2)}\boldsymbol{u}, \qquad \boldsymbol{u}_0 = \boldsymbol{u}(0) = \frac{1}{2}\left(\boldsymbol{x}+\boldsymbol{x}_1\right). \tag{3.2.13}$$

We recall that $D_N^{(2)}$ is the $(N-1)\times(N-1)$ second order Chebyshev differentiation matrix with boundary conditions incorporated, $\boldsymbol{u}(t)$ an $(N-1)\times 1$ vector, and $\boldsymbol{u}_0$ the initial condition. For simplicity we set $D = \frac{4}{\pi^2}D_N^{(2)}$.

The solution of (3.2.13) is approximated on the hyperbolic contour (3.2.3). A direct application of the Laplace transform (1.1.1) to (3.2.13) gives

$$(zI-D)\boldsymbol{U}(z) = \boldsymbol{u}_0, \tag{3.2.14}$$

where I is the $(N-1)\times(N-1)$ identity matrix. This formula is the matrix form of equation (3.2.9) and the scalar λ in (3.2.9) is the eigenvalue of the differentiation matrix D. To see this, suppose the matrix D has real and negative eigenvalues, $\lambda_1, ..., \lambda_n$ corresponding to the linearly independent eigenvectors, $\boldsymbol{v}_1, ..., \boldsymbol{v}_n$. If we expand the initial solution as a linear combination of eigenvectors, then

$$\boldsymbol{u}_0 = c_1\boldsymbol{v}_1 + ... + c_n\boldsymbol{v}_n.$$

The integrand $(zI-D)^{-1}\boldsymbol{u}_0$ can now be expanded as follows

$$(zI-D)^{-1}\boldsymbol{u}_0 = c_1(zI-D)^{-1}\boldsymbol{v}_1 + ... + c_n(zI-D)^{-1}\boldsymbol{v}_n.$$

Since the λ_j $(j = 1, ..., n)$ are the eigenvalues of D, this leads to

$$(zI - D)^{-1}\boldsymbol{u}_0 \;=\; \frac{c_1}{z-\lambda_1}\boldsymbol{v}_1 + ... + \frac{c_n}{z-\lambda_n}\boldsymbol{v}_n.$$

The application of the inversion method leads to

$$\boldsymbol{u}(t) \;=\; \frac{c_1}{2\pi i}\int_{-\infty}^{\infty}\frac{e^{z(\ell)t}}{z(\ell)-\lambda_1}d\ell + ... + \frac{c_n}{2\pi i}\int_{-\infty}^{\infty}\frac{e^{z(\ell)t}}{z(\ell)-\lambda_n}d\ell,$$

and the right-hand side is therefore equivalent to applying the inversion method to the scalar problem (3.2.10). But in this subsection we keep our attention to solving the problem with the matrix form (3.2.14).

To this end note from (3.2.14) that the solution of (3.2.13) may be written in the form

$$\boldsymbol{u}(t) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} e^{z(\ell)t}\left((zI-D)^{-1}\boldsymbol{u}_0\right)z'(\ell)d\ell. \tag{3.2.15}$$

As in [3; 4], (3.2.15) is approximated by the midpoint rule, on the set of points (3.2.7). The solution, after truncation and application of the symmetry, is of the form

$$\boldsymbol{u}_M(t) = \frac{h}{\pi}\text{Im}\left(\sum_{k=0}^{M-1} e^{z_k t}\left((z_kI-D)^{-1}\boldsymbol{u}_0\right)z_k'\right). \tag{3.2.16}$$

Because the differentiation matrix D is full, the solution of the linear systems

$$(z_kI - D)\boldsymbol{U}(z_k) = \boldsymbol{u}_0, \qquad k = 0, 1, ..., M, \tag{3.2.17}$$

represents the bulk of the computation in (3.2.16). For instance, a computational cost of $O(N^3)$ operations is required at each node z_k. Therefore, we are not going to use (3.2.17) directly, except for the comparison in Figure 3.3.

Speeding up the solution of the linear systems (3.2.17) requires the factorization of the coefficient matrices to a sparse form that is easier to solve. Because of the form of (3.2.17) the LU factorization cannot be used. As an alternative, we consider the Hessenberg reduction of D (see Appendix A). The decomposition yields

$$D = QHQ^T, \tag{3.2.18}$$

where $H = (h_{ij})$ is an upper Hessenberg matrix, i.e., $h_{ij} = 0$, $i > j+1$, and Q an orthogonal matrix. Then for each z_k, $k = 0, 1, ..., M-1$, the equation (3.2.17) becomes

$$(z_kI - QHQ^T)\boldsymbol{U}(z_k) = \boldsymbol{u}_0.$$

From this we have

$$Q(z_kQ^T - HQ^T)\boldsymbol{U}(z_k) = \boldsymbol{u}_0,$$

which yields

$$(z_kI - H)\boldsymbol{V}(z_k) = Q^T\boldsymbol{u}_0 \tag{3.2.19}$$

where $\boldsymbol{V}(z_k) = Q^T\boldsymbol{U}(z_k)$, so that

$$\boldsymbol{U}(z_k) = Q\boldsymbol{V}(z_k). \tag{3.2.20}$$

The solution $\boldsymbol{U}(z_k)$ for each z_k, is obtained by the computation of an almost triangular system (3.2.19) and combining the result in (3.2.20) at only $O(N^2)$ operations [31]. During this process, the Hessenberg reduction (3.2.18) is only computed once, beforehand. First, we compute

$$Q, \quad H, \quad \text{and} \quad \boldsymbol{V}_0 = Q^T\boldsymbol{u}_0.$$

Second, for each k, the system

$$(z_kI - H)\,\boldsymbol{V}(z_k) = \boldsymbol{V}_0,$$

is solved easily, since the $(z_kI - H)$ is an upper-Hessenberg matrix. Finally, the column vector

$$\boldsymbol{U}(z_k) = Q\boldsymbol{V}(z_k) \tag{3.2.21}$$

of the Laplace transform is computed and stored. Then (3.2.16) is computed as

$$\boldsymbol{u}_M(t) = \frac{h}{\pi}\text{Im}\left(\sum_{k=0}^{M-1} e^{z_kt}\boldsymbol{U}(z_k)z_k'\right),$$

to obtain the solution.

In Figure 3.3, we have plotted the CPU time versus N, the order of the differentiation matrix D in (3.2.17). That is, the CPU time with and without the Hessenberg decomposition. As N increases, we observe that the computational time of (3.2.17) increases drastically (dashed curve). However, the application of the Hessenberg decomposition (3.2.18) and (3.2.19)–(3.2.20) subsequently reduces the CPU time (circled curve).

3.2.5 Derivation of optimal contour parameters

We present theoretical formulas for the optimal parameters, α and μ, of the contour (3.2.3). We follow the approach of [4], where the scalar problem (3.2.8) was analysed. Optimal parameters are defined as the values that maximize the convergence rate of the midpoint/trapezoidal rule on the contour.

We shall distinguish two cases for the determination of the optimal contour parameters: parameters on a single contour t and on an interval $[t_0,\, t_1]$.

It is well-known that the total error estimate E_M of the midpoint rule on the real line is given by

$$E_M = E_d + E_t. \tag{3.2.22}$$

Here we have used E_d to represents the error introduced by the discrete approximation (3.2.6) of the continuous problem (3.2.4), and E_t that of the truncation error introduced in (3.2.16).

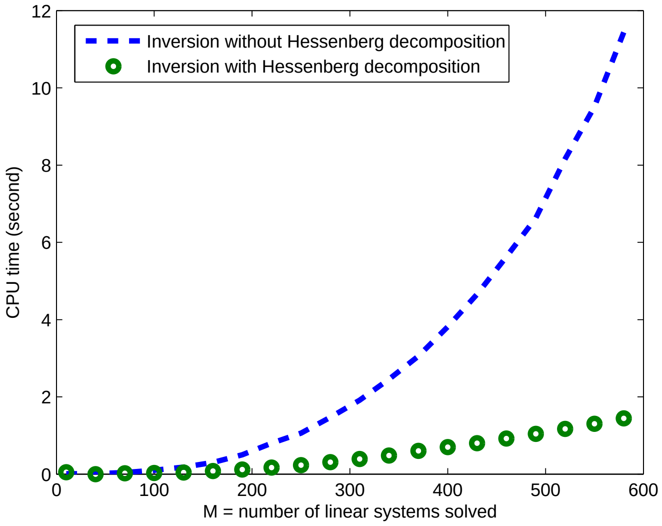


Figure 3.3: CPU time of the inversion formula (3.2.17) with and without the Hessenberg decomposition.

3.2.5.1 Inversion Laplace transform at a single time t

The optimal convergence rate of the hyperbolic family of contours (3.2.3) has been derived in [4]. The following were obtained:

$$E_{d+} = O(e^{-2\pi(\pi/2-\alpha)/h}), \quad E_{d-} = O(e^{\mu t-2\pi\alpha/h}), \quad h \to 0, \tag{3.2.23}$$

where

$$E_d = E_{d+} + E_{d-},$$

represents the discretization error. The truncation error was

$$E_t = O(e^{\mu t(1-\sin(\alpha)\cosh(hM))}), \quad M \to \infty. \tag{3.2.24}$$

The authors in [4] used an asymptotic approach to obtain these estimates. This requires that

$$E_{d+} = E_{d-} = E_t, \tag{3.2.25}$$

so that

$$-2\pi\left(\pi/2-\alpha\right)/h = \mu t - 2\pi\alpha/h = \mu t\left(1-\sin(\alpha)\cosh(hM)\right). \tag{3.2.26}$$

We solve these equations for h, μ and α in $(\pi/4, \pi/2)$. From the first equation in (3.2.26), we deduce that

$$\mu t = \pi\frac{4\alpha-\pi}{h}, \tag{3.2.27}$$

and from the last equation, it follows that

$$\cosh(hM) = \frac{2\alpha}{(4\alpha - \pi)\sin(\alpha)}. \tag{3.2.28}$$

Taking the inverse in (3.2.28) yields

$$A(\alpha) = hM = \cosh^{-1}\left(\frac{2\alpha}{(4\alpha - \pi)\sin(\alpha)}\right). \tag{3.2.29}$$

From (3.2.27) and (3.2.29), we now obtain

$$\mu = \frac{4\pi\alpha - \pi^2}{A(\alpha)}\frac{M}{t}, \quad \text{and} \quad h = \frac{A(\alpha)}{M}. \tag{3.2.30}$$

The corresponding convergence estimate can now be derived from h and μ in (3.2.30) as

$$E_M = O(e^{-B(\alpha)M}), \qquad \text{where} \qquad B(\alpha) = \frac{\pi^2 - 2\pi\alpha}{A(\alpha)}, \quad M \to \infty. \tag{3.2.31}$$

To derive the optimal contours parameters, note that as $B(\alpha)$ increases, E_M decreases exponentially. Thus E_M attains its optimum value at the maximum of $B(\alpha)$ over the interval $\left(\frac{\pi}{4}, \frac{\pi}{2}\right)$. A numerical computation of $B(\alpha)$ shows that the maximum is attained at

$$\alpha_\times = 1.1721. \tag{3.2.32}$$

With $\alpha_\times$ at hand, we can now compute $\mu_\times$ and $h_\times$ from (3.2.30). This yields

$$h_\times = \frac{1.0818}{M}, \qquad \text{and} \qquad \mu_\times = 4.4921\frac{M}{t}. \tag{3.2.33}$$

The optimal contour is then

$$z = 4.4921\frac{M}{t}\Big(1 + \sin(i\ell - 1.1721)\Big), \qquad \ell \in \mathbb{R}, \tag{3.2.34}$$

with the convergence rate

$$E_M = O(e^{-2.315M}) = O(10.2^{-M}), \qquad M \to \infty. \tag{3.2.35}$$

In Figure 3.4, we show the convergence rate of the Laplace inversion method at different times t. There, we have plotted the error

$$E_M = ||\boldsymbol{u}_M(t) - \boldsymbol{u}(t)||_\infty,$$

versus the number of function evaluations M (linear systems solved). Here $\boldsymbol{u}_M(t)$ is the numerical solution (3.2.16) and $\boldsymbol{u}(t)$ is the exact solution (1.4.4)–(1.4.5). The rapid convergence of the actual error (dash-dot curve) is in good agreement with the theoretical convergence (3.2.35) (dashed curve) obtained in [4].

The most important aspect of the Laplace inversion method is that it constantly achieves a high order of accuracy. As shown in Figure 3.4, it attains exponential accuracy to machine precision for only few linear system solves ($M \leq 13$). For $M \geq 14$ we observe an increase of the error in Figure 3.4, this is due to the ill-conditioning of the Laplace inversion. We shall discuss this in Subsection 3.2.6.

The drawback in using the Laplace inversion method is that since the optimal contour (3.2.34) depends on the time t, the evaluation of $\boldsymbol{u}(t)$ in (3.2.16) is carried out on a different contour for each t [4; 34]. For instance, suppose we want to evaluate $\boldsymbol{u}(t_j)$ for $t_1, ..., t_\ell$. To compute $\boldsymbol{u}(t_j)$ for each t_j, a new set of transforms $\boldsymbol{U}(z_k)$, $k = 0, ..., M-1$ in (3.2.17) is sampled on a different contour. This means $M \times \ell$ linear system are solved in total. For a matrix problem such as (3.2.16), this can be inefficient. As an example, in Figure 3.4, $3M$ systems were actually solved in total.

To overcome this drawback, the authors in [32] noted that the same evaluations of $\boldsymbol{U}(z_k)$, for $k = 0, ..., M-1$, can be used to approximate $\boldsymbol{u}(t)$ at different times t. The transforms $\boldsymbol{U}(z_k)$ in (3.2.17) can be computed once for the set of nodes $\{z_k, k = 0, 1, ..., M-1\}$. These evaluation of $\boldsymbol{U}(z_k)$ are then used to reconstruct the solution $\boldsymbol{u}(t)$ for any t in $[t_0,\, t_1]$. In addition, an exponential convergence rate, though with reduced decay rate compared to the case of a single contour, can be maintained on small intervals [32; 3; 4]. We discuss this topic in the next subsection.

3.2.5.2 Inversion Laplace transform on a interval $[t_0, \Lambda t_0]$

Assume that the solution is sought for t on the interval $[t_0, t_1]$, with $t_1 = \Lambda t_0$. An efficient implementation of the inversion method requires the use of the same contour (3.2.3) over the whole interval. Therefore, since the contour (3.2.3) depends on the parameter μ, this parameter should be held constant over the specified interval.

Various methods for finding such a contour exist [32; 3; 4]. In this thesis, we argue as follows: first note that only the equations (3.2.23) and (3.2.24) are time dependent. On one hand, the error E_t decreases when t increases and thus it attains its highest values at t_1. To see this, note that

$$1 - \sin(\alpha)\cosh(hM) \leq 0, \quad \text{for a fixed } h \neq 0 \quad \text{and} \quad M \to \infty,$$

and α is in $(\pi/4, \pi/2)$ which implies that the inequality $1/\sqrt{2} < \sin(\alpha) < 1$ holds. Multiplication of both side of the inequality by $\cosh(hM)$ yields

$$\frac{\cosh(hM)}{\sqrt{2}} < \sin(\alpha)\cosh(hM) < \cosh(hM),$$

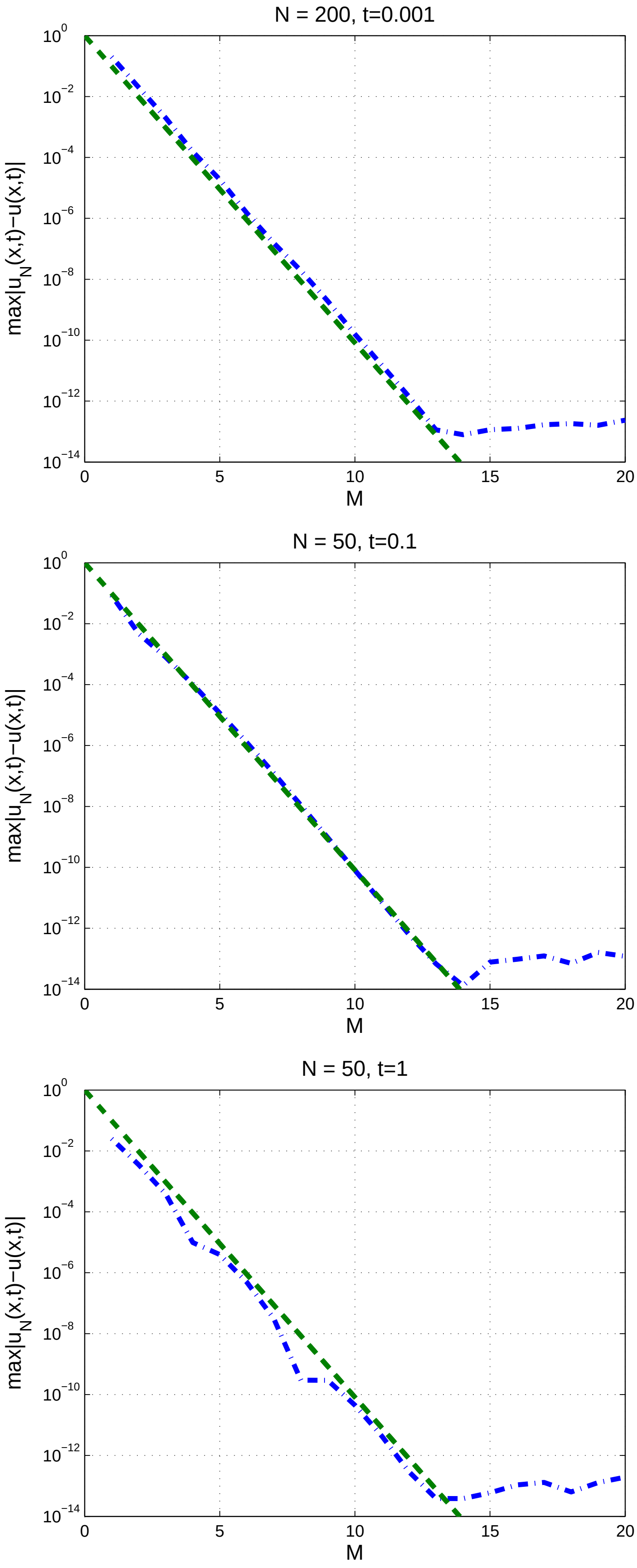


Figure 3.4: Convergence curve obtained when solving the semi-discrete problem (2.2.17) with the inversion formula (3.2.12) for different values of t. The dash-dot curve shows the error of the approximation (3.2.16), and the dashed line the theoretical error (3.2.35). The error norm is the infinite norm $||\boldsymbol{u}_M(t) - \boldsymbol{u}(t)||_\infty$.

but $\cosh(hM)/\sqrt{2} > 1$ for sufficiently large M, so that

$$1 - \sin(\alpha)\cosh(hM) < 0 \quad \text{for a fixed } h \neq 0 \quad \text{and} \quad M \to \infty.$$

On the other hand, the discretization error E_{d-} increases with t. This means the minimum of E_{d-} is at t_0.

To obtain high precision over the interval $[t_0,\, t_1]$, Weideman and Trefethen [4] considered the modification of the asymptotic equivalence (3.2.26) to

$$-2\pi\left(\pi/2 - \alpha\right)/h = \mu t_1 - 2\pi\alpha/h = \mu t_0\left(1 - \sin(\alpha)\cosh(hM)\right). \qquad (3.2.36)$$

We solve these equations for μ and h. From the first equation we have

$$\mu h = \frac{4\alpha\pi - \pi^2}{t_1}. \qquad (3.2.37)$$

The last equation in (3.2.36) together with with (3.2.37) yields

$$\cosh(hM) = \frac{(\pi - 2\alpha)\Lambda - \pi + 4\alpha}{(4\alpha - \pi)\sin\alpha},$$

where $\Lambda = t_1/t_0$. From this it follows that

$$A(\alpha) = hM = \cosh^{-1}\left(\frac{(\pi - 2\alpha)\Lambda - \pi + 4\alpha}{(4\alpha - \pi)\sin\alpha}\right). \qquad (3.2.38)$$

Therefore we obtain

$$h = \frac{A(\alpha)}{M} \qquad (3.2.39)$$

and

$$\mu = \frac{4\alpha\pi - \pi^2}{ht_1} = \frac{4\alpha\pi - \pi^2}{A(\alpha)}\frac{M}{t_1}. \qquad (3.2.40)$$

The contour parameters (3.2.40) and (3.2.39) are fixed and time independent. As a result, the corresponding contour (3.2.3) is also fixed over the interval $[t_0,\, t_1]$. Since the contour is fixed, we can now compute the set of transforms $\boldsymbol{U}(z_k)$ for $k = 0, 1, ..., M-1$ in (3.2.17). Then use the same set of computed values of $\boldsymbol{U}(z_k)$ to evaluate $\boldsymbol{u}(t)$ from the the midpoint rule (3.2.16) at different values of time t_j.

To sum up, the algorithm can be divided into two independent steps: First, the evaluation of $\boldsymbol{U}(z)$ at the required nodes. Second, the evaluation of $\boldsymbol{u}(t)$ in (3.2.16) at the required finite set of values t in $[t_0,\, t_1]$ [4; 32; 3].

More importantly, a relative exponential convergence rate is observed on the interval $[t_0, \Lambda t_0]$ [3]. But when Λ is large this exponential convergence rate is weak compared to the convergence rate obtained in the case of the single contour. To see this, note from the optimal parameters derived above, that the convergence rate as $M \to \infty$ is

$$E_M = O(e^{-B(\alpha)M}), \qquad (3.2.41)$$

with

$$B(\alpha) = \frac{\pi^2 - 2\pi\alpha}{\cosh^{-1}\left(\frac{(\pi-2\alpha)\Lambda+4\alpha-\pi}{(4\alpha-\pi)\sin(\alpha)}\right)}. \tag{3.2.42}$$

The maximum of $B(\alpha)$ can be computed numerically for different values of Λ. Some values of Λ are displayed in Table 3.2 as well as the corresponding optimal contour parameters and the decay rate $B(\alpha)$. Note the decay of $B(\alpha)$ (last column) as opposed to the value of Λ (first column). The convergence decreases as the interval increases.

Λ	α	$A(\alpha)$	$\mu t_1/M$	$B(\alpha)$
1	1.1721	1.0818	4.4921	2.3157
2	1.1431	1.5280	2.9417	1.7587
3	1.1181	1.8889	2.2134	1.5059
4	1.0969	2.1945	1.7838	1.3569
5	1.0791	2.4578	1.5013	1.2570
10	1.0236	3.3744	0.8871	1.0888
40	0.9464	5.2661	0.3842	0.7449
50	0.9381	5.5582	0.3452	0.7152

Table 3.2: Optimal parameters of the contour (3.2.3) for different Λ.

In Figure 3.5 we test the optimal parameters (3.2.38)–(3.2.40). The error is plotted in the maximum error norm

$$E_M = ||\boldsymbol{u}(t_1) - \boldsymbol{u}_M(t_1)||_\infty \tag{3.2.43}$$

for various Λ and $t_0 = 1/\Lambda$. Here $\boldsymbol{u}(t)$ is the exact solution (1.4.4) or (1.4.5) and $\boldsymbol{u}_M(t)$ the numerical solution (3.2.16). The values of E_M where all similar at $t = t_0$ and all values of t in between. The continuous curves represent the theoretical error as observed from the last column of Table 3.2. The dashed curves represent the actual convergence rate.

The good agreement between the theoretical and the numerical estimate can be observed from Figure 3.5, as both curves converge exponentially. As an example, let $\Lambda = 10$ and suppose we wish to attain an absolute error of, say, 10^{-10} over the interval [0.1, 1] when solving the model problem (1.4.1)–(1.4.3). From (3.2.41) this will require solving the equation

$$Ce^{-B(\alpha)M} \approx 10^{-10}, \tag{3.2.44}$$

where $B\left(\alpha\right)$ is given by (3.2.42) and C is an undetermined constant. Taking $C = 1$, and $\Lambda = 10$, results in

$$e^{-1.0888M} \approx 10^{-10} \Rightarrow M \approx 21,$$

which agrees with the middle set of curves in Figure 3.5.

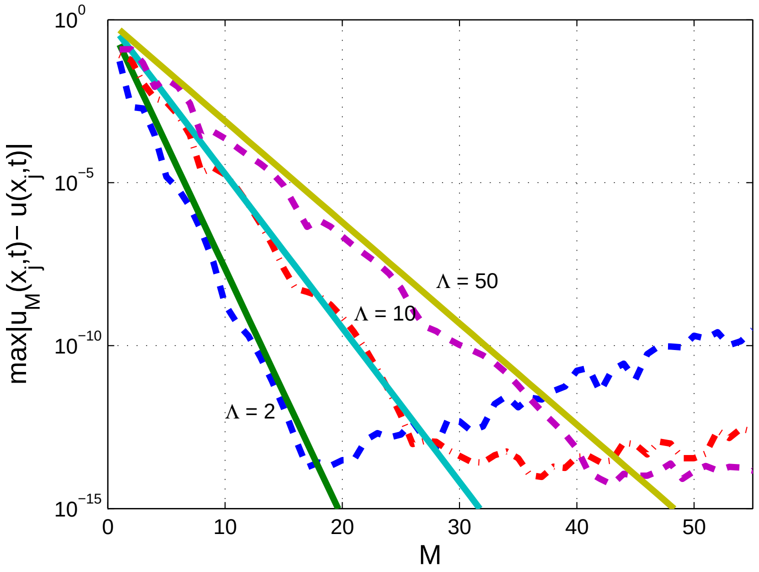


Figure 3.5: Actual error (dashed curve) and theoretical error (continuous curve) for $\Lambda = 2$, 10 and 50. We observe a good agreement between the numerical and theoretical results. Here $t_0 = 1/\Lambda$ and the interval is $[1/\Lambda, 1]$.

M	$\Lambda = 2$	$\Lambda = 4$	$\Lambda = 8$
2	2.10e-03	8.00e-02	4.77e-02
3	1.90e-03	1.90e-03	7.70e-03
4	2.84e-04	1.50e-03	3.20e-03
5	1.51e-05	3.95e-04	2.10e-03
6	6.52e-06	6.42e-05	7.35e-04
7	1.84e-06	4.28e-06	1.63e-04
8	3.21e-07	1.94e-06	1.41e-05
9	3.78e-08	1.87e-06	1.67e-05
10	1.84e-09	5.92e-07	9.61e-06
11	5.03e-10	1.46e-07	3.57e-06
12	1.97e-10	2.71e-08	1.02e-06
13	4.51e-11	2.85e-09	2.26e-07
14	8.23e-12	7.76e-10	3.46e-08
15	1.25e-12	5.03e-10	5.87e-09

Table 3.3: Convergence rate of the method (3.2.12) on an interval $[1/\Lambda, 1]$ for a few values of Λ. The first column indicates the number of points M in the trapezoid rule.

3.2.6 Ill-conditioning of the Laplace inversion

For values of $M > 13$, the convergence curves in Figure 3.4 start to increase. The exponential convergence is no longer valid. One might attribute this apparent contradictory behavior to either the spatial resolution or the ill-conditioning of the Laplace transform. However, since we have chosen the order N of the matrices D sufficiently large, the first claim is quickly discarded. In the floating-point evaluation of the transform $\boldsymbol{U}(z_k)$ at each node z_k, $k = 0, 1, ..., N$, a roundoff error is introduced and accumulates in the summation (3.2.16).

In practice, the system (3.2.17) is solved with a relative error $\boldsymbol{\rho}$ due to the floating-point in the computed value of $\boldsymbol{U}(z_k)$. Thus in (3.2.17) the computation taking place is that of

$$\boldsymbol{U}_k = \boldsymbol{U}(z_k)(\boldsymbol{x}_1 + \boldsymbol{\rho}_k), \qquad k = 0, 1, ..., M-1,$$

where

$$\boldsymbol{\rho}_k = \begin{bmatrix} \rho_{k1} \\ \vdots \\ \rho_{kN} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{x}_1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.$$

The entries of the $\boldsymbol{\rho}_k$ are such that $|\rho_{kj}| \leq \epsilon$ for $j = 1, ..., N$, and ϵ is the machine precision. As a result, the inverse $\boldsymbol{u}(t)$ is affected by the same rounding error $\boldsymbol{\rho}_k$ as

$$\boldsymbol{u}^{\times}(t) = \frac{h}{\pi}\text{Im}\left\{\sum_{k=0}^{M-1} e^{z_k t}\boldsymbol{U}(z_k)^T\Big(\boldsymbol{x}_1 + \boldsymbol{\rho}_k\Big)z_k'\right\},$$

that is

$$\boldsymbol{u}^{\times}(t) = \boldsymbol{u}_M(t) + \frac{h}{\pi}\text{Im}\left\{\sum_{k=0}^{M-1} e^{z_k t}\Big(\boldsymbol{U}(z_k)^T\boldsymbol{\rho}_k\Big)z_k'\right\}. \qquad (3.2.45)$$

As M increases the error term increases significantly because of the exponential term $e^{z_k t}$ and eventually becomes larger than the first term. We conclude that when M is small (≤ 13 as in this case), the convergence of the Laplace inversion dominates the conditioning error. However, for large M, the conditioning error increases and eventually dominates the other error. Because of this, we take values of $M \leq 13$ for the computation of (3.2.16).

In this chapter we have discussed three time integration methods for the numerical solution of the parabolic PDEs. In the next chapter we shall compare these methods numerically.

Chapter 4

Numerical comparison

In this chapter, we will be concerned with the evaluation of the performance of the following three methods: the Laplace inversion method, MATLAB's **ODE15s**, and the Crank-Nicolson method.

To this end, we will consider two criteria: the accuracy per linear system solve and the computational time based on a 1500 MHz laptop. Other implementations and other machines may give different results.

We consider the model problem (1.4.7)–(1.4.9) that we have been using throughout this thesis with the corresponding exact solutions (1.4.4) and (1.4.5).

4.1 Criterion I: Accuracy per linear systems solve

When solving parabolic PDEs, a common feature of the Laplace transforms and the methods-of-lines is that the major part of the computation involves solving linear systems as in (3.1.9), (3.1.16)–(3.1.17) and (3.2.17). The accuracy of the methods is dependent on the number of linear systems solved. Thus, we consider the number of linear systems solved for each method as a measure of comparison. The most accurate method is then the method which attains higher accuracy with the smallest number of linear systems solved.

We shall report the error in the L_∞ norm, defined as

$$E_M = ||\boldsymbol{u}(t) - \boldsymbol{u}_M(t)||_\infty, \tag{4.1.1}$$

where $\boldsymbol{u}(t)$ is the vector of exact solution values at the sample points x_j, and $\boldsymbol{u}_M(t)$ the numerical solution, which involves the solution of M linear systems at time t. We have chosen the $N \times N$ Chebyshev matrices D sufficiently large ($N = 50$) to fully resolve the solution.

As from the discussion in the last part of Section 3.2, the comparison will be considered on two different cases: comparison at a single value of t as well as on an interval $[t_0, t_1]$.

4.1.0.1 Comparison at a single value of t

In Figure 4.1, we have plotted the error versus the number of linear systems solves M at $t = 1$. Here "ILT" denotes the convergence rate of the Laplace inversion method; "CN" the convergence rate obtained by the Crank-Nicolson method; "ODE15s", the convergence rate of **ODE15s** and "Theor conv." denotes the theoretical convergence rate (3.2.35). The values in brackets next to the "ODE15s" data points represent the tolerance parameters (Retol, Abstol); see Section 3.1.1.

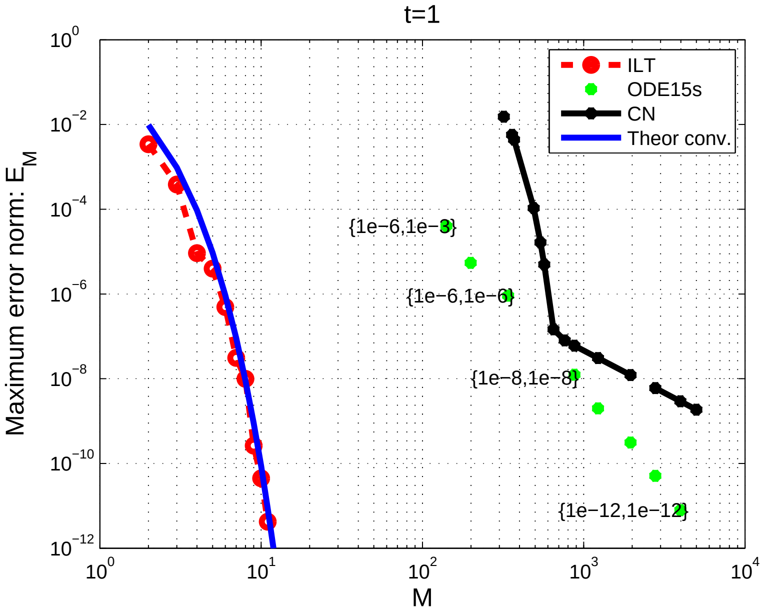


Figure 4.1: Comparison of the error of the inverse Laplace transform method (ILT), **ODE15s** (ODE15s) and Crank-Nicolson method (CN). M is the number of linear system solves used to compute the solution of the model problem (3.2.13) at $t = 1$.

A key feature to look for in Figure 4.1 is the relative positioning of the different curves. For a given accuracy, curves that are further to the left takes fewer linear systems to solve or fewer steps to achieve that accuracy. We observe that the Laplace transform method performs better than the other methods. It attains high accuracy with only a few function evaluations (i.e., linear system solves). This is in contrast to **ODE15s** and the Crank-Nicolson methods, for which a substantial number of steps is needed to attain the same level of accuracy.

Note that the convergence curve of the Crank-Nicolson method exhibits two types of behavior separated by a kink at about $M = 600$. This can be attributed to the non-smoothness of the initial condition (2.2.17)[35]. The straight line curve in the loglog plot for $M > 600$ indicates a power function relationship between the error and the step size as predicted by the order of the method. To see this, note that the linear relation is

$$\log E_M = m \log \Delta t + \log C,$$

where $\Delta t = 1/M$ (since $t = 1$) and for some constant C. This yields

$$\log E_M = \log C - m \log M.$$

Define $b = \log E_M$, $c = \log C$ and $e = \log M$, then

$$b = c - me. \tag{4.1.2}$$

We compute c and m by the least squares method. In matrix notation (4.1.2) becomes

$$\boldsymbol{b} = A\boldsymbol{x},$$

where

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_\ell \end{bmatrix}, \quad \text{and} \quad A = \begin{bmatrix} 1 & -e_1 \\ 1 & -e_2 \\ \vdots & \vdots \\ 1 & -e_\ell \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} c \\ m \end{bmatrix}.$$

To determine the value of m, we consider the data points of Figure 4.1. Using MATLAB, we find $m = 2.006$, which confirms that the Crank-Nicolson method is of order 2. We apply the same method to obtain the order of convergence of **ODE15s** and found $m = 4.985$ which confirms the default order 5 of **ODE15s**.

4.1.0.2 Comparison on an interval $[t_0,\, t_1]$

Next, we consider the inverse Laplace transform method on an interval $[t_0,\, t_1]$, where $t_1 = \Lambda t_0$ and $\Lambda > 1$. As discussed in the previous section, the computational effort of the inverse Laplace transform method comes from solving the linear system (3.2.17) for each z_k, $k = 0, 1, ..., M$. But because $\boldsymbol{U}(z_k)$ in (3.2.17) is independent of t, it can be evaluated once at a fixed set of quadrature points z_k. The same evaluations of $\boldsymbol{U}(z_k)$ can then be used to approximate $\boldsymbol{u}(t)$ at different t. This preserves the exponential convergence rate, but with a reduced decay constant, see Figure 3.5.

Suppose we wish to compute the solution at ℓ time levels τ_j in $[t_0,\, t_1]$, such that,

$$t_0 \leq \tau_1 < \tau_2 ... < \tau_\ell \leq t_1. \tag{4.1.3}$$

We propose the following two methods for the Laplace inversion method.

Method 1: Use a new contour optimal for each τ_j, $j = 1, ..., \ell$. The advantage of this method is that we have high accuracy as described by the convergence rate (3.2.35). The disadvantage is that, because the optimal contour (3.2.34) depends on t, this will require using ℓ different contours and therefore ℓM separate linear system solves (3.2.17).

Method 2: Use a single contour for all τ_j, $j = 1, ..., \ell$. The disadvantage is that we have lower accuracy as given by (3.2.41) with values in Table 3.2, but only M linear system solves.

Our aim is to decide which method is advantageous in terms of the number of linear systems solved. In other words, for which values of ℓ does Method 1 gives higher accuracy than Method 2.

Let us consider Method 1. Suppose we want to attain an accuracy ϵ on the interval $[t_0,\, t_1]$ for a minimum number of linear system solves M. Then, from the optimal convergence rate (3.2.35), we require for each τ_j, $j = 1, ..., \ell$, that

$$C_1 e^{-2.32 M_1} \approx \epsilon, \tag{4.1.4}$$

where M_1 is the number of function evaluations. We assume the constant C_1 is approximately the same for all τ_j. Thus the total number of linear system solves for Method 1 is

$$\ell M_1 \approx \frac{\ell}{2.32} \log{(C_1/\epsilon)}.$$

As for Method 2, to attain an accuracy ϵ over the interval $[t_0,\, t_1]$, we require

$$C_2 e^{-B(\alpha) M_2} \approx \epsilon,$$

where $B(\alpha)$ is given in (3.2.42) and C_2 a constant. The number of linear system solves is therefore

$$M_2 \approx \frac{1}{B(\alpha)} \log{(C_2/\epsilon)}. \tag{4.1.5}$$

Method 1 has the same amount of work as Method 2 if (4.1.4) is equal to (4.1.5), i.e.,

$$\frac{\ell}{2.32} \log{(C_1/\epsilon)} \approx \frac{1}{B(\alpha)} \log{(C_2/\epsilon)}.$$

C_1 and C_2 are difficult to estimate, so we shall assume $C_1 \approx C_2$. Then

$$\ell \approx \frac{2.32}{B(\alpha)}, \tag{4.1.6}$$

and when ℓ is larger than the number on the right Method 2 is more efficient, otherwise Method 1.

For example, to solve the problem on the interval $[1, 50]$ to an accuracy $\epsilon = 10^{-14}$, we deduce from Table 3.2 that

$$\ell > \frac{2.32}{0.7152} \quad \Longrightarrow \quad \ell > 4. \tag{4.1.7}$$

If the solution is required at 4 or more time levels in [0.02, 1], Method 2 is better than Method 1.

We illustrate this property in Table 4.1 and in Figure 4.2, where we have plotted the error on different intervals $[t_0, \Lambda t_0]$.

We have used the error norm

$$E_M = \max_{t \in \{\tau_1, \ldots, \tau_\ell\}} ||\boldsymbol{u}(t) - \boldsymbol{u}_M(t)||_\infty. \tag{4.1.8}$$

The numerical results in Table 4.1 were obtained on the interval $[t_0, \Lambda t_0]$ for $\Lambda = 50$ and $t_0 = 1/\Lambda$, i.e., the time interval is [0.02, 1]. We compute solutions at

$$[0.02,\ 0.04,\ 0.08,\ 0.3,\ 0.7,\ 0.8,\ 0.9,\ 1]\,,$$

i.e., $\ell = 8$.

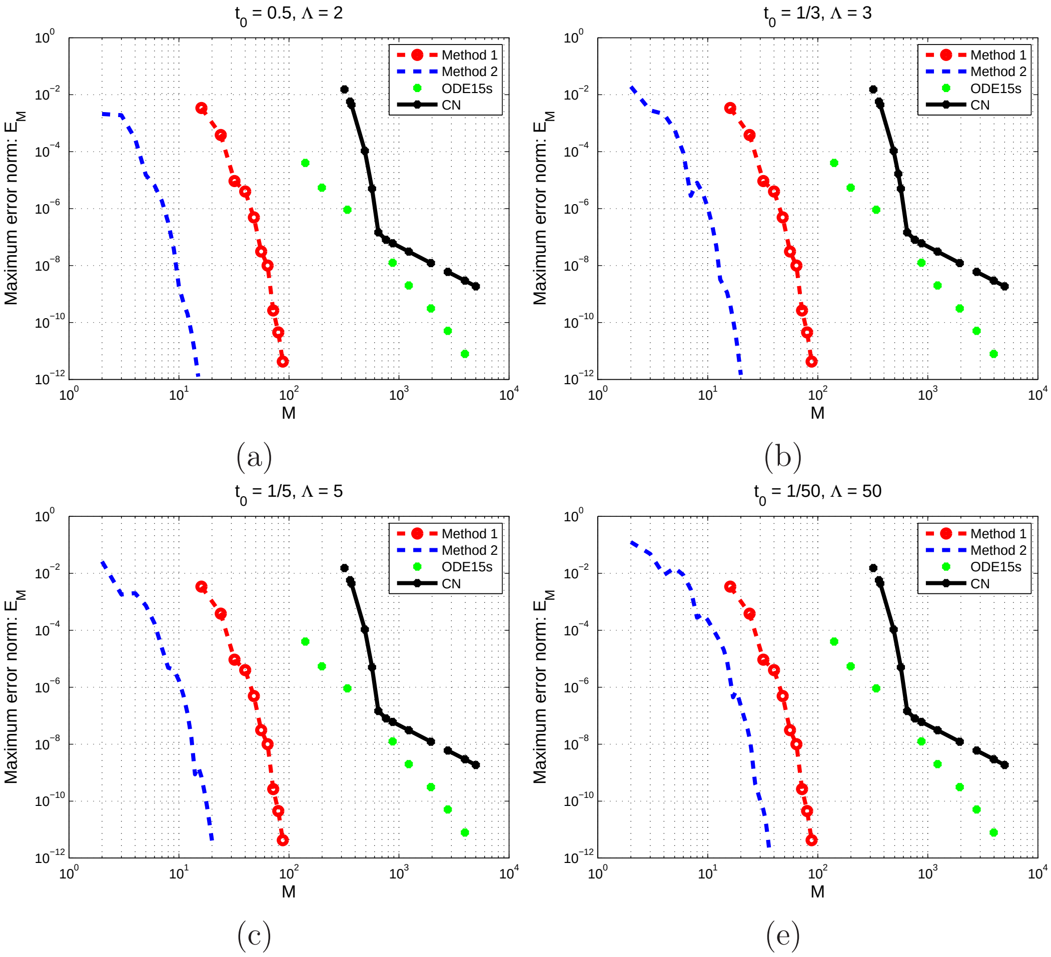
The value M of Method 1 should be multiplied by 8 to obtain the total number of linear systems solves to compute the solution in the specified interval. Thus, for example, in the first row the total number of linear system solves is 4 × 8 for Method 1.

We observe from these results that Method 2 gives better accuracy than Method 1. Method 1 is better than **ODE15s**, which is better than the Crank-Nicolson method. Method 1 becomes increasingly less efficient compared to Method 2 as ℓ increases.

The above observations are confirmed in Figure 4.2 where we have plotted the error for different values of Λ. We can observe that despite the deterioration of the convergence rate of Method 2 as Λ increases, it remains superior to **ODE15s** and CN over the interval $[t_0, \Lambda t_0]$.

$M\ell$	Method 1	M	Method 2	M	ODE15s	M	CN
24	9.17e-05	6	8.70e-03	141	8.77e-05	320	1.52e-02
40	1.05e-05	8	2.77e-04	199	8.57e-06	462	2.77e-04
48	1.14e-06	12	6.73e-05	340	1.95e-07	490	1.06e-04
56	1.21e-07	14	2.02e-05	462	8.74e-07	570	4.96e-06
64	1.24e-08	16	1.85e-06	878	2.04e-08	650	1.46e-07
72	1.19e-09	18	7.05e-07	1232	3.55e-09	762	8.27e-08
80	1.12e-10	26	1.35e-09	1952	3.11e-10	877	6.24e-08
88	1.19e-11	32	1.05e-10	2779	5.14e-11	2780	6.21e-09
96	1.26e-12	38	2.34e-12	3978	8.00e-12	5000	1.92e-09

Table 4.1: Comparison of the error defined by (4.1.8) of Method 1, Method 2, **ODE15s** and the Crank-Nicolson method on the interval $[t_0/\Lambda, 1]$ for $\Lambda = 50$. M is the number of functions evaluations (linear system solves) executed by each method. Here $\ell = 8$.



(a) (b)

(c) (e)

Figure 4.2: Comparison of the error (4.1.8) of the Method 1; Method 2; **ODE15s** and the Crank-Nicolson method. M is the number of linear system solves to compute the solution of the model problem (3.2.13). The results were computed for $\Lambda = 2$ (a), $\Lambda = 3$ (b), $\Lambda = 5$ (c) and $\Lambda = 50$ (e), on the interval $[1/\Lambda, 1]$ and $\ell = 8$.

It is apparent both from Subsection 3.2.5.2 and Figure 4.2 that the uniform convergence estimate deteriorates exponentially in Λ. We are interested in knowing if for larger Λ, the convergence rate of Method 2 will eventually be surpassed by the rate of convergence of **ODE15s**. To this end, we first compute the optimal parameters for large Λ given in Table 4.2. In Figure 4.3, we plot on a logarithm scale the convergence rate of Method 2 and **ODE15s** for $\Lambda = 10^{15}$ on the interval $[1/\Lambda, 1]$.

The result shows reasonably equivalent accuracy for small M. But as M increases the Laplace method regain its superiority. In Figure 4.4 we did the same for $\Lambda = 10^{20}$ and conclude that even for huge values of Λ, the Laplace inversion method has higher accuracy than **ODE15s**.

Λ	α	$A(\alpha)$	$\mu t_1/M$	$B(\alpha)$
10^{15}	0.806	38.48	0.007	0.125
10^{20}	0.801	50.3	0.004	0.096

Table 4.2: Optimal parameters for $\Lambda = 10^{15}$ and 10^{20}.

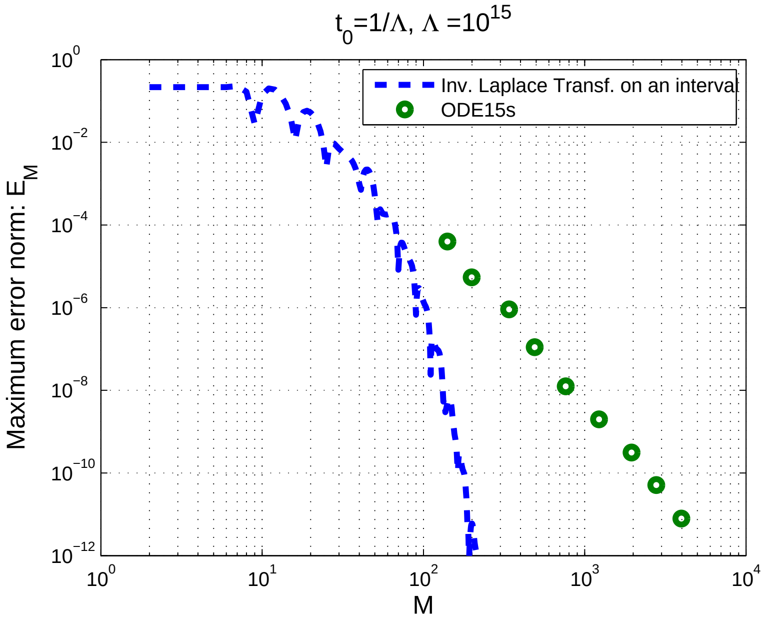


Figure 4.3: Convergence error of **ODE15s** and Method 2 for $\Lambda = 10^{15}$.

4.2 Criterion II: Accuracy per CPU time

Examining the accuracy per linear system solve is insufficient for a complete comparison of the numerical methods. This criterion does not reflect the improved performance caused by the LU and Hessenberg factorizations of Section 3.1 and subsection 3.2.5.2. To this end, we will also consider the cost in computational time, CPU time per accuracy, which is measured by MATLAB's command *CPU time*.

Even though the computational cost of each method is significantly affected by the way the algorithm is implemented, note that the main computational cost of the Laplace method comes from the Hessenberg decomposition (3.2.18)

$$D = QHQ^T,$$

and the evaluation (3.2.19)–(3.2.20), namely

$$(z_k I - H)\boldsymbol{V} = Q^T \boldsymbol{u}_0,$$

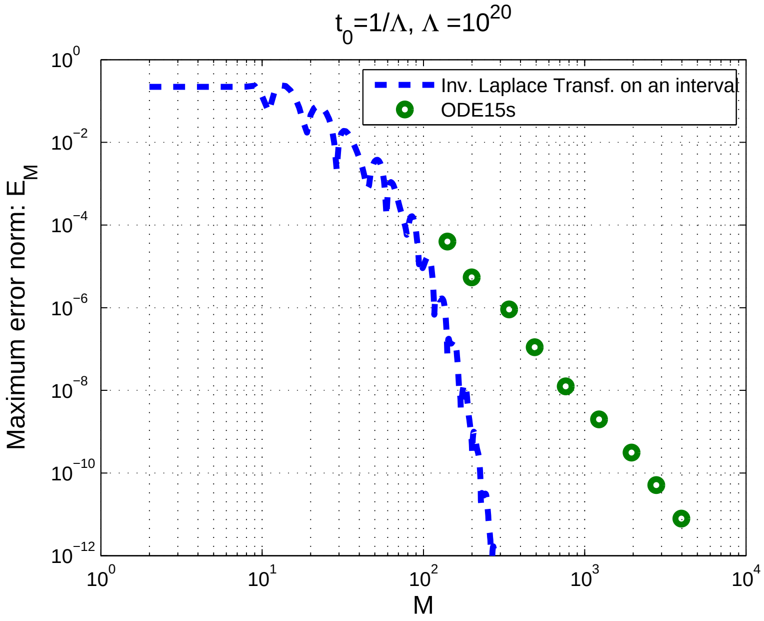


Figure 4.4: Convergence error of **ODE15s** and Method 2 for $\Lambda = 10^{20}$.

where $\boldsymbol{V} = Q^T\boldsymbol{U}$, and

$$\boldsymbol{U} = Q\boldsymbol{V}.$$

Similarly, the main computational effort for **ODE15s** comes from the n-step evaluation in (3.1.9)

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j \boldsymbol{u}^{n+1} = h \left(D\boldsymbol{u}^{n+1} + \boldsymbol{b} \right) + \kappa\gamma_k \left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^0 \right),$$

and the solution of a linear system involving the matrix D at each step [28, p. 70]. For the Crank-Nicolson method the major computational cost comes from the evaluation of the sequence (3.1.16)–(3.1.17)

$$L\boldsymbol{y} = P^T(B\boldsymbol{u}^n) + hP^Tb,$$

$$\boldsymbol{U}\boldsymbol{u}^{n+1} = \boldsymbol{y}.$$

However, for a fair comparison, we consider the CPU time of the entire algorithm in each case. In Figure 4.5, we show the run time results for the four methods as a function of the accuracy. In this figure, we have plotted the CPU time against the absolute error befined by (4.1.8). Note, however, that the CPU time of the Method 1 displayed in that figure is for a single value of t. Therefore, to obtain the total computational time for all 8 time levels, the values of the CPU time in Figure 4.5 should be multiplied by 8. As a consequence of this, the curve of Method 1 will be located above that of Method 2. The CPU time of **ODE15s** and Crank-Nicolson have been computed directely over the entire interval of integration, i.e., $[1/\Lambda, 1]$ for $\Lambda = 50$.

According to the figure, both Laplace Method 1 and 2 give better results compared to **ODE15s** and the Crank-Nicolson method. Again Method 2 is superior to Method 1 if ℓ increases.

We concluded the last paragraph of Section 4.1 by noting the superiority of Method 2 even for huge values of Λ over the intervals $[1/\Lambda, 1]$. Here we do a similar test but for the CPU time over the same intervals. Although, in Figure 4.6, we observe a slight increase of the CPU time of the Method 2 as Λ increases, it remains superior to **ODE15s**.

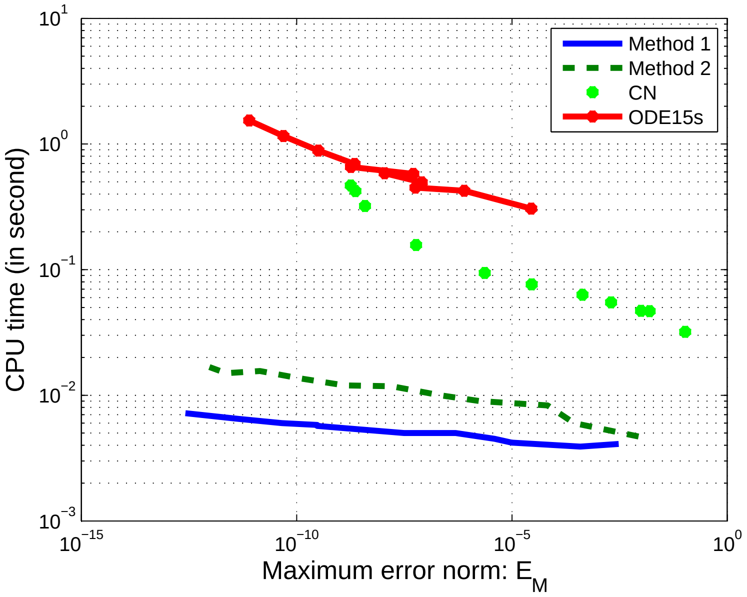


Figure 4.5: CPU time vs error of Method 1 (ILT1), Method 2 (ILT2) **ODE15s** and Crank Nicolson, on the interval interval $[0.02, 1]$, $\ell = 8$. Here the values of the CPU time of Method 1 should be multiplied by 8.

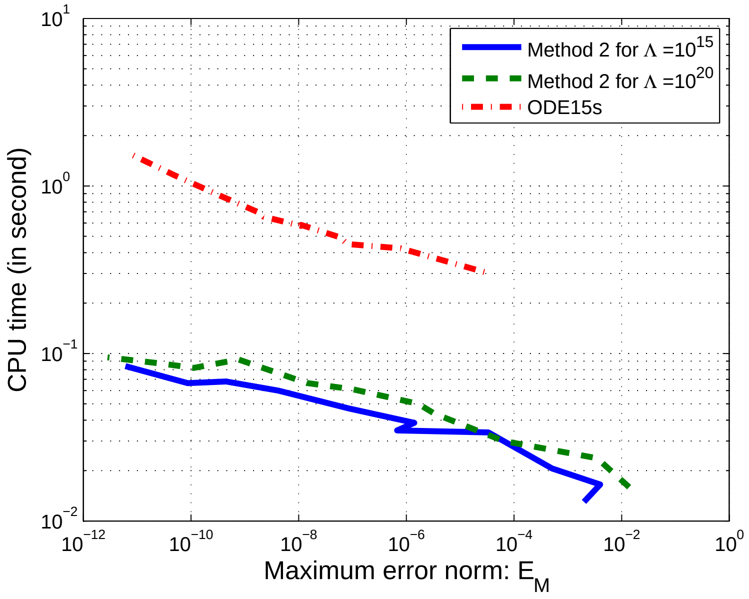


Figure 4.6: CPU time vs error for **ODE15s** and Method 2 on the interval $[1/\Lambda,\,1]$, for huge values of Λ.

Chapter 5

Applications

5.1 Axial dispersion model

In this chapter we will consider the application of the Laplace transform to solve the axial dispersion model (ADM). The dispersion equation is used to describe the process of fluid flow through tubes/pipes in many chemical engineering applications. The flow is normally in the laminar flow regime, when the diameter is very small or when the fluid is highly viscous.

Typically the ADM is characterized by parameters such as the Peclet number, P_e, to describe the variation from the plug flow reactor. The plug flow reactor (PFR) is an ideal model used to describe properties of continuous flowing fluid in pipe. It assumes no mixing in the axial direction (i.e., direction of the flow), complete mixing in the radial direction and a uniform velocity profile across the radius. The absence of longitudinal mixing is the special characteristic of this reactor. This assumption is at the opposite extreme of the complete mixing assumption of the ideal continuous tank reactor (CSTR).

In practice, most systems do not conform to either of these two ideal reactors, as deviations due to dispersion occur. The magnitude of the dispersion is determined by the Peclet number P_e, which describes the deviation from the CSTR or the PFR. As P_e increases from 0 to ∞ the reactor changes from CSTR to PFR.

The solution of the model requires an appropriate set of boundary conditions, describing the flux entering and leaving the reactor. Two type of boundary conditions exist, the open and closed [36]. The closed boundaries are the most used, as they assure continuity of the fluxes across the boundary where the stream enters and leaves the system only once. And it reduces to both the CSTR when $P_e = 0$ and to the PFR when $P_e \to \infty$. This is not the case for the open boundary condition.

5.1.1 Axial dispersion model

The axial dispersion model has its origins in chemical engineering to model the simultaneous convection and diffusion in a stream flowing through a vessel. In non-dimensional form, the model can be written as a one dimensional equation

$$\frac{\partial c}{\partial t} - \frac{1}{P_e}\frac{\partial^2 c}{\partial x^2} + \frac{\partial c}{\partial x} = 0, \qquad 0 \leq x \leq 1 \qquad \text{and} \qquad t > 0, \tag{5.1.1}$$

where c is the solute concentration (or sometimes the temperature) at the position x within the vessel, t is the time, and P_e the Peclet number.

The closed boundary conditions, also known as Danckwerts boundary conditions [37], have the form:

$$\text{At} \quad x = 0, \qquad c(0,t) - \frac{1}{P_e}c_x(0,t) = c_0(t). \tag{5.1.2}$$

$$\text{At} \quad x = 1, \qquad c_x(1,t) = 0. \tag{5.1.3}$$

Here $c_0(t)$ is the dimensionless inlet concentration, which can be a function of time. The equation (5.1.1) shows that the flux into the system comprises both the convective and dissipative flux, and (5.1.2)–(5.1.3) express the fact that the system is closed at the right exit, but is fed at the left exit.

The initial condition is

$$c(x,0) = f(x), \tag{5.1.4}$$

for some function $f(x)$. An analytical solution of (5.1.1)–(5.1.3) can be obtained using the Fourier method.

5.1.2 Analytical solution

The equations (5.1.1)–(5.1.4) is solved with Fourier's method, i.e., the method of separation of variables. The detailed description of this method can be found in [38, p. 92].

We are interested in the particular case where the input signal takes the form of a Dirac impulse $\delta(t)$, describing a stimulus response experiment when a tracer is injected in a pure liquid at the origin $x = 0$. The boundary and initial conditions become $c_0(t) = \delta(t)$ and $f(x) = 0$. At the end point $x = 1$, the exit system response derived from (5.1.1) is

$$c(x,t) = 2e^{xP_e/2}\sum_{n=1}^{\infty}\frac{P_n(x)Q_n(t)}{P_e(P_e^2 + 4P_e + 4w_n)}, \tag{5.1.5}$$

where

$$P_n(x) = w_n\cos(w_n x) + \frac{P_e}{2}\sin(w_n x), \tag{5.1.6}$$

$$Q_n(t) = e^{-(w_n^2 + \frac{P_e^2}{4})t/P_e}, \tag{5.1.7}$$

and the w_n are the positive roots of the equation

$$\tan(w_n) = \frac{4w_nP_e}{4w_n^2 - P_e^2}. \tag{5.1.8}$$

The roots w_n of equation (5.1.8) correspond to the abscissa of the intersection points of the curves

$$\tan(w) \qquad \text{and} \qquad \frac{4wP_e}{4w^2 - P_e^2}. \tag{5.1.9}$$

These two curves are drawn in Figure 4.1. We notice that each of the positive roots is simple and lie in an interval of the form $[k\pi, (k+1)\pi]$ for $k = 0, 1, ...$ We disregard the solution $w_0 = 0$. From (5.1.8) we deduce

$$\cos w_n = \frac{4w_n^2 - P_e^2}{4w_nP_e} \sin w_n. \tag{5.1.10}$$

Relation (5.1.5) at $x = 1$, in view of (5.1.10), can be simplified to

$$c(1,t) = e^{\frac{P_e}{2}} \sum_{n=1}^{\infty} \frac{2w_n \sin w_n[P_e^2 + 4w_n^2] \exp\{-\frac{(P_e^2+4w_n^2)t}{4P_e}\}}{P_e[P_e^2 + 4P_e + 4w_n^2]}. \tag{5.1.11}$$

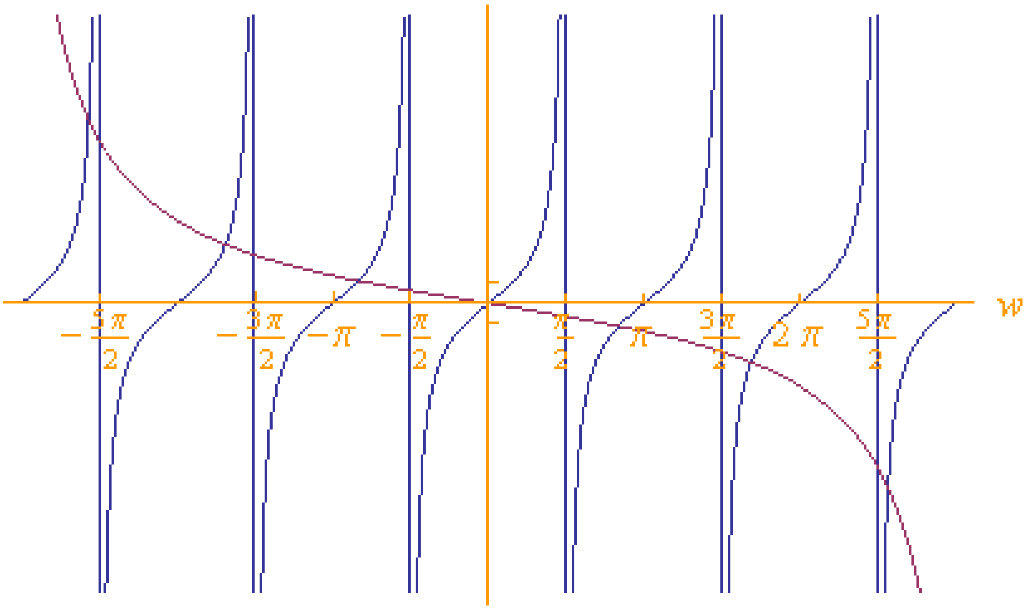


Figure 5.1: Axial distribution of the roots of the equation (5.1.8)

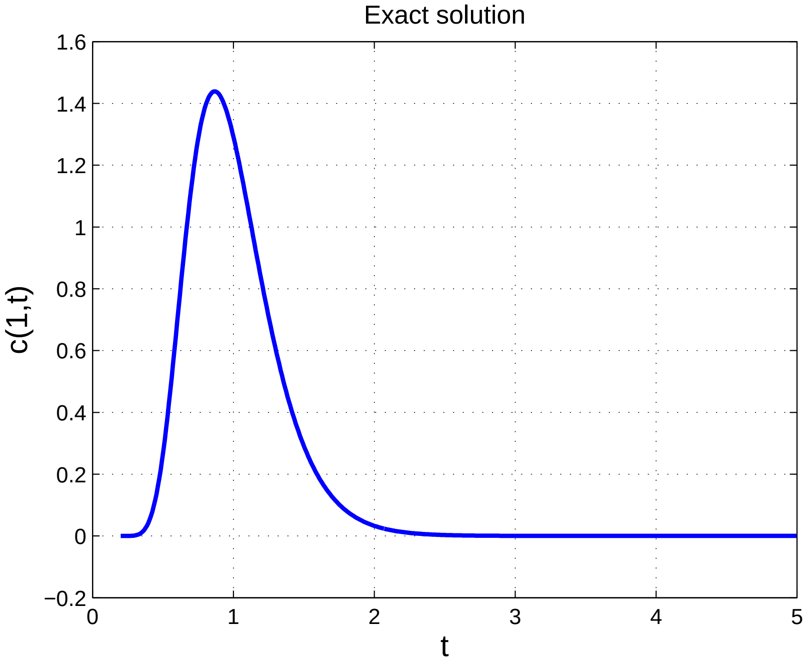


Figure 5.2: Exact solution (5.1.11)

Unfortunately, the series (5.1.11) is slowly convergent for small t and for large value of Peclet number P_e [39]. We will only consider large values of t in the numerical experiments reported here.

Since the PDE (5.1.1) is defined on the interval $[0, 1]$, we consider the linear transformation

$$x \longleftrightarrow \frac{1}{2}\,(x+1)\,,$$

to convert to the interval $[-1, 1]$ for numerical computation. Thus (5.1.1)–(5.1.3) becomes

$$\frac{\partial c}{\partial t} - \frac{4}{P_e}\frac{\partial c}{\partial x^2} + 2\frac{\partial c}{\partial x} = 0, \qquad -1 \leq x \leq 1, \tag{5.1.12}$$

with

$$x = -1, \qquad c(-1,t) - \frac{2}{P_e}c_x(-1,t) = c_0(t), \tag{5.1.13}$$

$$x = 1, \qquad c_x(1,t) = 0, \tag{5.1.14}$$

and

$$c(x,0) = 0, \qquad t = 0. \tag{5.1.15}$$

5.1.3 Laplace inversion method

From the theory of Laplace transforms presented in Chapter 1 and Section 3.2, the equation (5.1.12) can be rewritten as a second-order ODE

$$zC(x,z) - \frac{4}{P_e}\frac{\partial^2 C(x,z)}{\partial x^2} + 2\frac{\partial C(x,z)}{\partial x} = 0, \tag{5.1.16}$$

where $C(x,z)$ is the Laplace transform of $c(x,t)$. Analogously, the boundary conditions (5.1.13)–(5.1.14) become

$$x = -1, \qquad C(x,z) - \frac{2}{P_e}C_x(x,z) = 1, \tag{5.1.17}$$

$$x = 1, \quad C_x(1,t) = 0, \tag{5.1.18}$$

and

$$C(x,z) = 0, \qquad z = 0. \tag{5.1.19}$$

To solve (5.1.16)-(5.1.19) we consider the spectral discretization method of Chapter 2. Then we integrate the subsequent system of ODE with the Laplace transform of Section 3.2. We shall compute the first and second differentiation matrices to approximate the derivatives.

Unlike in Chapter 1 where we used the MATLAB code `chebdif`, owing to the nature of the boundary conditions, we consider here the code `cheb2bc` [23]. The code `cheb2bc` computes the sets $\{x_j\}_{j=0}^{N}$ of N Chebyshev extreme points and the differentiation matrices of the first and second order, $D_N^{(1)}$ and $D_N^{(2)}$, respectively. The code also return two $N \times 2$ matrices $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ which contains the boundary conditions at $x = \pm 1$.

The code is called from the command

```
»[x,D1,D2,phip,phim] = cheb2bc(N,bc);
```

The input N is the order of the derivative and `bc` an array containing the boundary conditions given by

$$\texttt{bc} = [0\ 1\ 0;\ 1\ \ -2/P_e\ 1]\,.$$

The first three numbers in the array define the boundary condition (5.1.18), whereas the second three numbers define the boundary condition (5.1.17).

The output `D1` and `D2` contain the first and second differentiation matrices respectively. The matrices `phip` and `phim` incorporate the boundary conditions.

Using `cheb2bc`, the equations (5.1.16)–(5.1.19) in matrix form yields

$$z\boldsymbol{C}(z) - \frac{4}{P_e}D_N^{(2)}\boldsymbol{C}(z) + 2D_N^{(1)}\boldsymbol{C}(z) + \frac{4}{\pi}\boldsymbol{\phi} + 2\boldsymbol{\psi} = 0, \tag{5.1.20}$$

where

$$\boldsymbol{C}(z) = \begin{pmatrix} C(x_1,z) \\ \vdots \\ C(x_N,z) \end{pmatrix}.$$

The solution $\boldsymbol{C}(z)$ can be found by solving the equation

$$(z_k I - D)\,\boldsymbol{C}(z) = \boldsymbol{c}_0, \tag{5.1.21}$$

for each node $z_k = z(\ell_k)$, $\quad k = 0, 1..., M-1$ of the hyperbolic contour (3.2.3). Here

$$D = \frac{4}{\pi} D_N^{(2)} - 2D_N^{(1)}, \qquad \boldsymbol{c}_0 = -2\left(\frac{2}{\pi}\boldsymbol{\phi} + \boldsymbol{\psi}\right). \tag{5.1.22}$$

Note that the eigenvalues of D are real and negative as was confirmed by numerical experiment. Therefore the theory of Chapter 3 is applicable. Using the inversion formula (3.2.16) and the midpoint nodes (3.2.7) the approximate solution may be written in the form

$$\boldsymbol{c}_M(t) = \frac{h}{\pi}\mathrm{Im}\left(\sum_{k=0}^{M-1} e^{z_k t}\left((z_k I - D)^{-1}\boldsymbol{c}_0\right) z_k'\right). \tag{5.1.23}$$

In Figure 5.3 we display the solution at the outflow boundary $x = 1$. In Figure 5.4, we have plotted, as a function of M, the quantity

$$E_M = \max_{1 \le t \le 4} |\boldsymbol{c}(t) - \boldsymbol{c}_M(t)|. \tag{5.1.24}$$

at the outflow boundary $x = 1$. Here $\boldsymbol{c}(t)$ is the exact solution (5.1.11) and $\boldsymbol{c}_M(t)$ the numerical solution computed by (5.1.23). The maximum was computed at 500 equally spaced values of t in [1, 4].

The solid curve in Figure 5.4 represents the error in (5.1.24), whereas the dashed curve represents the theoretical convergence rate obtained in (3.2.35). The good agreement between the theoretical error prediction and the numerical error can be observed in that figure. Figure 5.5 shows the solution in two dimensions.

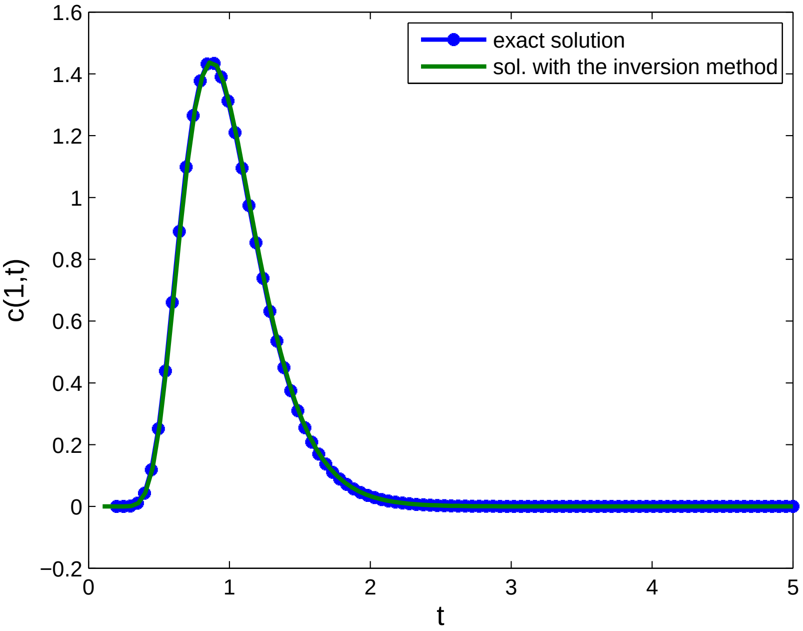


Figure 5.3: Solution at the exit point, i.e., $x = 1$ of equation (5.1.16) with Laplace inversion method. Here $P_e = 20$.

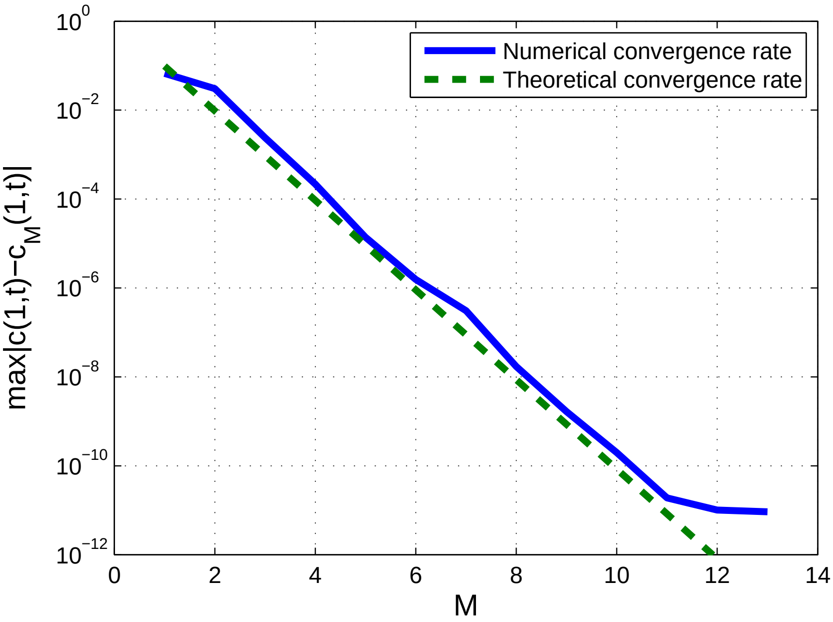


Figure 5.4: Convergence error rate of the Laplace inversion

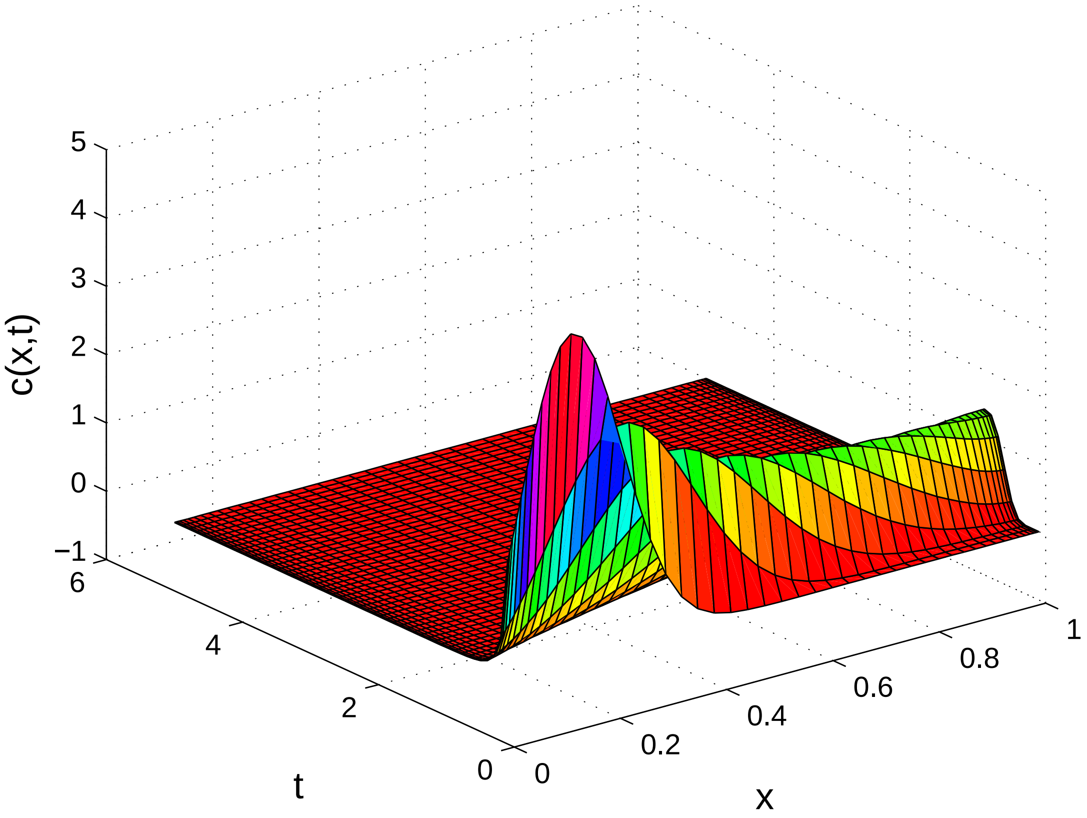


Figure 5.5: Solution of equation (5.1.16) in 2-D with Laplace transform method. Here $P_e = 20$.

5.2 Heat equation in two dimensions

As a second example of the application of the Laplace inversion method we consider the heat equation in two dimensions

$$u_t = \gamma \Delta u \tag{5.2.1}$$

defined on the square $[-1,1] \times [-1,1]$, where Δ is the two dimensional Laplacian

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{5.2.2}$$

As a model problem we consider initial and boundary conditions

$$\begin{aligned} u(-1,y,t) &= 0, \\ u(1,y,t) &= 0, \\ u(x,-1,t) &= 0, \\ u(x,1,t) &= 0, \\ u(x,y,0) &= e^x \left(1-x^2\right)\left(1-y^2\right). \end{aligned}$$

and $\gamma = 0.02$, which is the same problem solved in [40].

The solution function $u(x,y,t)$ of this differential equation describes the temperature, for example in a thin metal plate with zero temperature at the edges. We approximate the solution $u(x,y,t)$ at discrete time points t and discrete positions (x_i, y_i), $0 \le i,\, j \le n+1$. We semi-discretize the differential equation by the spectral collocation method.

To approximate the second derivative operator, we set up a grid based on Chebyshev points independently on both directions x and y, called a *tensor product grid*. On this grid, Δ is approximated by the tensor product, also known as *Kronecker products*. Let L be the discrete approximation of Δ.

Definition 5.2.1 *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$. Then the Kronecker product, also called tensor product, of A and B is the matrix defined by*

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn}B \end{pmatrix} \in \mathbb{R}^{mp\times nq}. \tag{5.2.3}$$

The i,j block entry is $a_{ij}B$ [41, Chapter 13]. In MATLAB the Kronecker product of A and B is computed by the command kron(A,B).

Definition 5.2.2 *Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$. The Kronecker sum, also called the tensor sum, of A and B, denoted $A \oplus B$, is the $mn \times mn$ matrix $(I_m \otimes A) + (B \otimes I_n)$ [41, Chapter 13].*

As an example, let

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix}.$$

Then the Kronecker product is the block matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix} = \left(\begin{array}{cc|cc} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{12} & a_{12}b_{22} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{12} & a_{22}b_{22} \end{array}\right).$$

From (5.2.2) we note that to approximate the Laplacian, we must differentiate in both directions x and y independently at the Chebyshev nodes.

The discrete approximation L_N of the Laplacian becomes the Kronecker sum of the second order differentiation matrix D, with boundary conditions incorporated, and I_N the identity matrix order N i.e.,

$$L_N = D \oplus I_N = I_N \otimes D + I_N \otimes D. \tag{5.2.4}$$

Let us consider the case $N = 6$. Since (5.2.1) has Dirichlet boundary conditions, we compute the differentiation matrix D with the code `chebdif`. The calling command is `»[x,D] = chebdif(N,2)`, where N is the order of the matrix . To impose the boundary conditions, we set $D = D(2:5, 2:5, 2)$, like we did in Chapter 2

$$D = \begin{pmatrix} -14 & 6 & -2 \\ 4 & -6 & 4 \\ -2 & 6 & -14 \end{pmatrix}.$$

The second derivative with respect to x is then given by

$$I_6 \otimes D = \left(\begin{array}{ccc|ccc|ccc} -14 & 6 & -2 & & & & & & \\ 4 & -6 & 4 & & & & & & \\ -2 & 6 & -14 & & & & & & \\ \hline & & & -14 & 6 & -2 & & & \\ & & & 4 & -6 & 4 & & & \\ & & & -2 & 6 & -14 & & & \\ \hline & & & & & & -14 & 6 & -2 \\ & & & & & & 4 & -6 & 4 \\ & & & & & & -2 & 6 & -14 \end{array}\right).$$

The second derivative with respect to y is computed by

$$D \otimes I_6 = \left(\begin{array}{ccc|ccc|ccc}
-14 & & & 6 & & & -2 & & \\
 & -14 & & & 6 & & & -2 & \\
 & & -14 & & & 6 & & & -2 \\
\hline
4 & & & -6 & & & 4 & & \\
 & 4 & & & -6 & & & 4 & \\
 & & 4 & & & -6 & & & 4 \\
\hline
-2 & & & 6 & & & -14 & & \\
 & -2 & & & 6 & & & -14 & \\
 & & -2 & & & 6 & & & -14
\end{array}\right).$$

The approximation L_6 of the Laplacian is then

$$L_6 = I_6 \otimes D + D \otimes I_6.$$

In matrix form (5.2.1) is

$$\boldsymbol{u}_t = \gamma L_N \boldsymbol{u} \tag{5.2.5}$$

We solve this equation by the inversion method of Section 3.2. The Laplace transform applied to (5.2.5) yields

$$(zI - \gamma L_N)\,\boldsymbol{U}(z) = \boldsymbol{u}_0,$$

where I is the $N^2 \times N^2$ identity matrix.

On the contour (3.2.3), the inversion formula gives

$$\boldsymbol{u}(t) = \frac{h}{2\pi i}\int_{-\infty}^{\infty} e^{z(\ell)t} z'(\ell) \left((zI - \gamma L_N)^{-1}\boldsymbol{u}_0\right) d\ell. \tag{5.2.6}$$

On the discrete points $z_k = z(\ell_k)$, with ℓ_k defined in (3.2.7) for $k = 0, 1, ..., M-1$, the application of the trapezoidal/midpoint rule yields

$$\boldsymbol{u}(t) = \frac{h}{\pi}\mathrm{Im}\left(\sum_{k=0}^{M-1} e^{z(\ell_k)t} z'(\ell_k) \left((zI - \gamma L_N)^{-1}\boldsymbol{u}_0\right)\right). \tag{5.2.7}$$

In Figures 5.6-5.10, we show the solutions in two dimensions at different times. Each figure was computed with a computational time less than 45 seconds on a 2 GB 2003 Pentium machine.

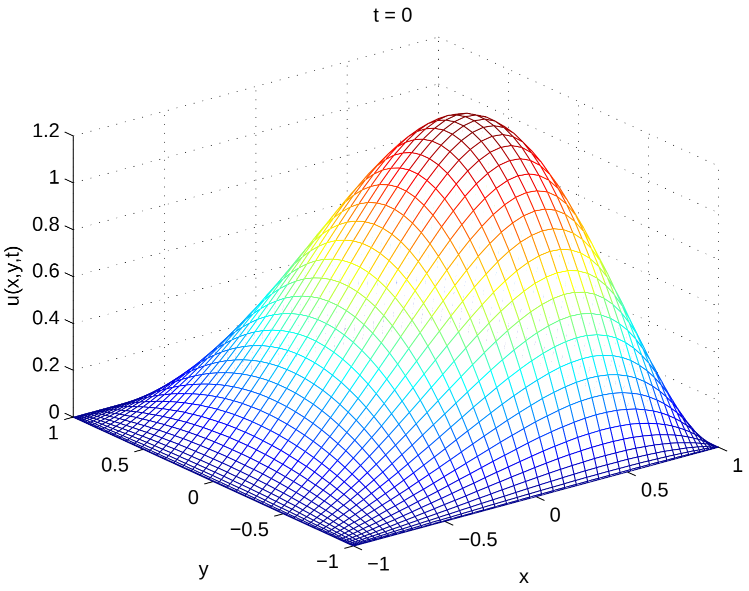


Figure 5.6: Solution of the 2D problem at $t = 0$

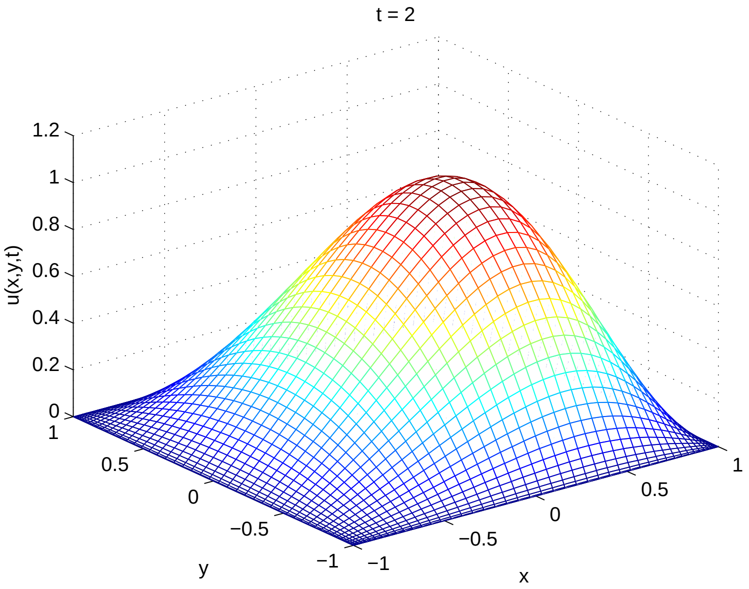


Figure 5.7: Solution of the 2D problem at $t = 2$

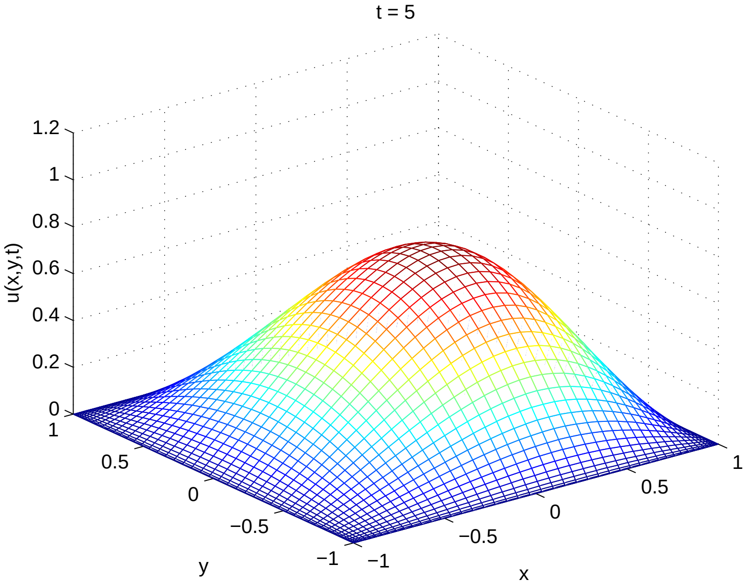


Figure 5.8: Solution of the 2D problem at $t = 5$

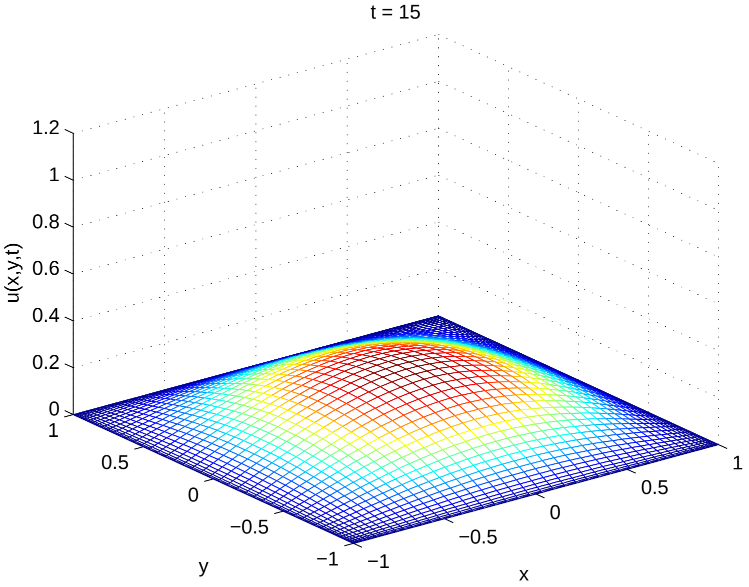


Figure 5.9: Solution of the 2D problem at $t = 15$

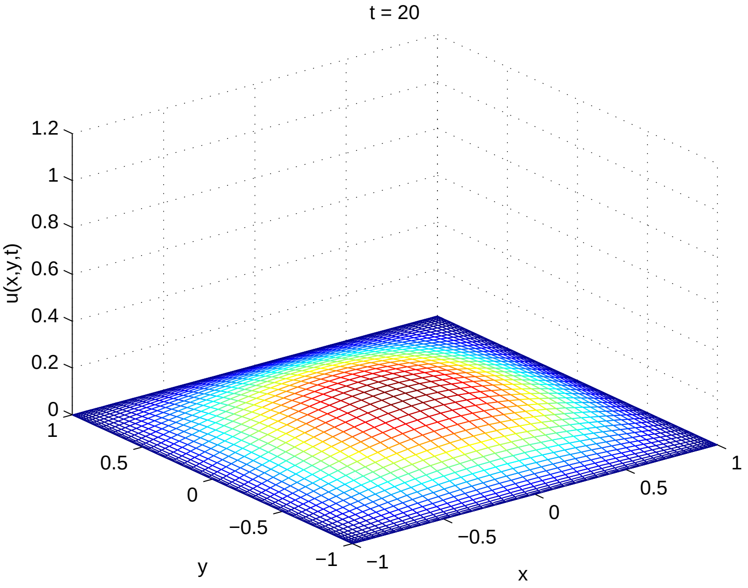


Figure 5.10: Solution of the 2D problem at $t = 20$

Chapter 6

Conclusion

Different methods exist for the time integration of semi-discrete linear parabolic PDEs. Our focus here was on the pure parabolic PDEs, with eigenvalues on the negative real axis. We have investigated three methods, namely the Laplace inversion method and the method-of-lines based on MATLAB **ODE15s** and the Crank-Nicolson method. The Laplace inversion method is relatively new, while the method-of-lines is classic. The purpose of this investigation was therefore to compare the new method with the classic methods.

Two criteria of comparison were considered: the accuracy per linear system solve, and the accuracy per CPU time. As model problem for these experiment we considered the standard heat equation.

Chapter 2 contains a discussion of the spectral methods used to semi-discretize the model problem. Spectral methods employ global interpolation polynomials that are differentiated to approximate derivatives. From this we derived the differentiation matrices. The semi-discretization of Chapter 2 leads to a system of ODEs which is integrated by the time integration methods discussed in Chapter 3.

In Chapter 3, we have analysed the different time integration methods. The first method was MATLAB's **ODE15s** algorithm, which is based on a family of the implicit linear multistep formulas, the numerical differentiation formulas (NDFs). The NDFs present some advantages on stability and accuracy compared to more conventional BDFs methods. The second method discussed was the Crank-Nicolson method, known to be an unconditionally stable method. A technique to enhance the computational cost was introduced, namely the LU decomposition with row pivoting.

The third method used was the Laplace inversion formula, which is relatively new in this context. We followed the idea of Talbot, in which the Bromwich integral is approximated by the trapezoidal/midpoint rules on a deformed contour. On such a contour the trapezoidal rule converges exponentially. In this thesis, we choose the hyperbolic contour as an alternative to Talbot's original contour.

The convergence rate of the trapezoidal rule on the hyperbolic contour depends critically on the contour's parameters. In [4] Weideman and Trefethen determined the contour's parameters for optimum accuracy. Theoretical error analysis were performed to estimate the optimal values of the parameters that define the contour. This was done by noting that the error associated with this method are the discretization error (when approximating the integral by the midpoint sum) and the truncation error (due to the truncation of the infinite midpoint sum). The investigation of the Laplace inversion method included the cases where the problem is solved at a single value of t as well as on an interval $[t_0, t_1]$.

Chapter 4 is about the numerical comparison of the three methods used in this thesis. Two criteria for comparison were considered. The accuracy per linear system solved and the accuracy per CPU time. As a measure of the error, we use the maximum error norm.

For the first criterion, the accuracy per linear system solved, we consider first the case where the solution is computed at a single value of t. The numerical results show the Laplace method to be more accurate compared to MATLAB's **ODE15s** and the Crank-Nicolson method. The Laplace method exhibits an exponential convergence rate while the other two methods have only a linear convergence. Then the solution is sought over an interval $[t_0, t_1]$. We defined Method 1 and Method 2. The first uses a different contour for each $\tau_j \in [t_0, t_1]$. The second uses a single contour for all $\tau_j \in [t_0, t_1]$. Despite a decreasing convergence rate as Λ increases, Method 2 presented the best result in terms of accuracy per linear system solved for various values of t over a certain interval.

To use the second criteria, the accuracy per CPU time, we consider the CPU time of each algorithm enhanced by the decomposition techniques of Chapter 3. Again as in the previous criterion, the Method 2 was superior.

In Chapter 5, we consider the application of the Laplace method to solve the axial dispersion model in Section 5.1 and the two dimensional heat equation in Section 5.2. The results obtained confirm the exponential accuracy of the Laplace inversion method.

6.1 Topics for further work

In this thesis we restricted our comparison of the Laplace inversion method and method-of-lines only to MATLAB's **ODE15s** and the Crank-Nicolson method. A direction for further work is to consider more advanced time integrators used in the method-of-lines. In that direction consideration of methods such implicit Runge-Kutta methods (Radau 5) and the modified extended backward differentiation formula (MEBDF) proposed in [42] can provide more insight into which of the Laplace transform and the method-of-lines is more efficient.

Appendices

Appendix A

Matrix Decomposition

Unlike finite differences or finite element methods where the approximation of boundary value problems lead to sparse matrices, spectral methods give a matrix representation of the derivative that is neither sparse nor symmetric. It can be computationally expensive to work with such matrices.

Matrix decomposition aims to transform a given problem into a canonical form that can be solved more readily. The matrix computational problem $A\boldsymbol{x} = \boldsymbol{b}$ for $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{x}$ and $\boldsymbol{b} \in \mathbb{R}^n$ is not always easy to solve in an optimal way. Its computational simplicity often depends on the structure of the matrix A; full unstructured matrices have a computational cost of $O(n^3)$. These matrices are decomposed into low-order simple form such as sparse, diagonal or triangular matrices that require only $O(n^2)$ operations.

Definition A.0.1 *Let $A \in \mathbb{R}^{n\times n}$. A is said to be*
1. symmetric if it is equal to it transpose, that is if $A = A^T$,
2. orthogonal if its inverse equals to its transpose, that is $A^{-1} = A^T$, so that AA^T=$A^TA = I$ where I is the identity matrix.

Definition A.0.2 *A permutation matrix P is any square matrix resulting from the rearranging the rows of the identity matrix of the same order. Permutation matrices are orthogonal.*

A.0.1 LU decomposition with partial pivoting

The equation $A\boldsymbol{x} = \boldsymbol{b}$ has a unique solution $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ if the matrix A is non-singular. This suggests the computation of the inverse A^{-1} of A as a means of finding x, however the computation of the inverse of a matrix is not recommended. One strategy is to transform A to some triangular system which is easier to solve by forward or backward substitution. This process is referred to as LU decomposition with partial pivoting.

Theorem A.0.3 *Let $A \in \mathbb{R}^{n\times n}$ a non-singular matrix i.e $det(A(1 : k, 1 : k)) \neq 0$ for $k = 1 : n - 1$. Then there exist a permutation matrix P, a* *unit-lower triangular matrix L and an upper-trinagular marix U such that*

$$A = PLU.$$

For a proof of this theorem we refer to [43, p. 121].

To solve the system

$$A\boldsymbol{x} = \boldsymbol{b},$$

it is reduced to

$$U\boldsymbol{x} = \boldsymbol{y},$$

where U is upper triangular and y is the vector solution of a lower triangular system

$$L\boldsymbol{y} = P^T\boldsymbol{b}.$$

Here the factorization of A as LU has no real importance other than being computationally convenient for obtaining a solution to the original problem [31, p. 92-102].

A.0.2 Hessenberg Decomposition

We present another approach for the reduction a matrix $A \in \mathbb{R}^{n\times n}$ to solve effectively the linear system $A\boldsymbol{x} = \boldsymbol{b}$. A matrix is $A = (a_{ij})$ is called an upper Hessenberg matrix if $a_{ij} = 0$ whenever $i > j + 1$. Thus an upper Hessenberg matrix has the form

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2(n-1)} & a_{2n} \\ 0 & a_{32} & a_{33} & \dots & a_{3(n-1)} & a_{3n} \\ 0 & 0 & a_{43} & \dots & a_{4(n-1)} & a_{4n} \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_{n(n-1)} & a_{nn} \end{pmatrix}.$$

The Hessenberg decomposition of $A \in \mathbb{R}^{n\times n}$ is computed as

$$A = QHQ^T,$$

where Q is an orthogonal matrix and $Q^TQ = QQ^T = I$, with I the identity matrix and H an upper Hessenberg matrix. The following theorem is proved in [31, p. 362]:

Theorem A.0.4 *Let $A \in \mathbb{R}^{n\times n}$, then there exist an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$, such that*

$$Q^TAQ = H,$$

where H is a Hessenberg matrix.

Consider the system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{A.0.1}$$

using the decomposition $A = QHQ^T$, the system is reduced to

$$H\boldsymbol{y} = \boldsymbol{c}, \quad \text{for} \quad \boldsymbol{y} = Q^T\boldsymbol{x}, \quad \text{and} \quad \boldsymbol{c} = Q^T\boldsymbol{b} \tag{A.0.2}$$

thus

$$\boldsymbol{x} = Q\boldsymbol{y}. \tag{A.0.3}$$

The solution $\boldsymbol{x}$ is obtained by combining the results in (A.0.3) of an almost triangular system in (A.0.2) (since H is an almost triangular matrix). This is done at only $O(N^2)$, compared to $O(N^3)$ in (A.0.1) where a full matrix is solved.

Consider the system, which we encounter in this thesis (see (3.2.13))

$$(zI - A)\,\boldsymbol{x} = \boldsymbol{b} \quad \text{for a given}\, z. \tag{A.0.4}$$

We argue that A.0.4 is easier to solve with the Hesenberg decomposition than with the LU. To see this, note that the LU decomposition of A gives

$$(zI - LU)\,\boldsymbol{x} = \boldsymbol{b}.$$

Because of the form of the factor on the left, the LU decomposition has no effect to improve the decomposition of (A.0.4). In contrast, the Hessenberg decomposition solves this problem more readily, as follows

$$\left(zI - QHQ^T\right)\boldsymbol{x} = \boldsymbol{b}$$

and since $I = QQ^T$, this results in the equation

$$(zI - H)\,\boldsymbol{y} = Q^T\boldsymbol{b}, \tag{A.0.5}$$

where $\boldsymbol{y} = Q^T\boldsymbol{x}$.

First we solve the almost triangular system (A.0.5) to obtain $\boldsymbol{y}$ and then combine the result as

$$\boldsymbol{x} = Q\boldsymbol{y},$$

to obtain $\boldsymbol{x}$.

List of References

[1] Weideman, J.A.C. and Trefethen, L.N.: The eigenvalues of second-order spectral differentiation matrices. *SIAM J. Numer. Anal.*, vol. 25, no. 6, pp. 1279–1298 (electronic), 1988.

[2] Talbot, A.: The accurate numerical inversion of Laplace transforms. *J. Inst. Math. Appl.*, vol. 23, no. 1, pp. 97–120, 1979.

[3] López-Fernández, M. and Palencia, C.: On the numerical inversion of the Laplace transform of certain holomorphic mappings. *Appl. Numer. Math.*, vol. 51, no. 2-3, pp. 289–303, 2004.

[4] Weideman, J.A.C. and Trefethen, L.N.: Parabolic and hyperbolic contours for computing the Bromwich integral. *Math. Comp.*, vol. 76, no. 259, pp. 1341–1356 (electronic), 2007.

[5] Duffy, D.G.: *Transform methods for solving partial differential equations.* 2nd edn. Chapman & Hall/CRC, Boca Raton, FL, 2004.

[6] Schiff, J.L.: *The Laplace transform.* Undergraduate Texts in Mathematics.

[7] Spiegel, M.R.: *Theory and problems of Laplace transforms.* Schaum Publishing Co., New York, 1965.

[8] Duffy, D.: On the numerical inversion of Laplace transform: comparison of the three new methods on characteristic problems from applications. *ACM Trans. Math. Soft.*, vol. 19, no. 3, pp. 333–359, 1993.

[9] Davies, B. and Martin, B.: Numerical inversion of the Laplace transform: a survey and comparison of methods. *J. Comput. Phys.*, vol. 33, no. 1, pp. 1–32, 1979.

[10] Narayanan, G.V. and Beskos, D.E.: Numerical operational methods for time-dependent linear problems. *Internat. J. Numer. Methods Engrg.*, vol. 18, no. 12, pp. 1829–1854, 1982.

[11] Abate, J. and Valko, P.P.: Multi-precision Laplace transform inversion. *Internat. J. Numer. Methods Engrg.*, vol. 60, no. 5, pp. 979–993, 2004.

[12] Weideman, J.A.C.: Numerical integration of periodic functions: a few examples. *Amer. Math. Monthly*, vol. 109, no. 1, pp. 21–36, 2002.

[13] Atkinson, K.E.: *An introduction to numerical analysis.* 2nd edn. John Wiley & Sons Inc., New York, 1989.

[14] Kress, R.: *Numerical analysis*, vol. 181 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1998.

[15] Martensen, V.E.: Zur numerischen auswertung uneigentlicher integrale. *Z. Angew. Math. Mech.*, vol. 48, 1968.

[16] Shampine, L.F. and Reichelt, M.W.: The MATLAB ODE suite. *SIAM J. Sci. Comput.*, vol. 18, no. 1, pp. 1–22, 1997.

[17] Antimirov, M.Y., Kolyshkin, A.A. and Vaillancourt, R.: *Applied integral transforms*, vol. 2 of *CRM Monograph Series.* American Mathematical Society, Providence, RI, 1993.

[18] Fornberg, B.: *A practical guide to pseudospectral methods*, vol. 1 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, Cambridge, 1996.

[19] Gottlieb, D. and Orszag, S.A.: *Numerical analysis of spectral methods: theory and applications.* Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1977. CBMS-NSF Regional Conference Series in Applied Mathematics, No. 26.

[20] Trefethen, L.N.: *Spectral methods in MATLAB*, vol. 10. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[21] Boyd, J.P.: *Chebyshev and Fourier spectral methods.* 2nd edn. Dover Publications Inc., Mineola, NY, 2001.

[22] Huang, W.Z. and Sloan, D.M.: The pseudospectral method for solving differential eigenvalue problems. *J. Comput. Phys.*, vol. 111, no. 2, pp. 399–409, 1994.

[23] Weideman, J.A.C. and Reddy, S.C.: A MATLAB differentiation matrix suite. *ACM Trans. Math. Software*, vol. 26, no. 4, pp. 465–519, 2000.

[24] Welfert, B.D.: Generation of pseudospectral differentiation matrices. I. *SIAM J. Numer. Anal.*, vol. 34, no. 4, pp. 1640–1657, 1997.

[25] Moler, C. and Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, vol. 45, no. 1, pp. 3–49 (electronic), 2003.

[26] Schiesser, W.E.: *The numerical method of lines.* Academic Press Inc., San Diego, CA, 1991.

[27] Ashino, R., Nagase, M. and Vaillancourt, R.: Behind and beyond the MATLAB ODE suite. *Comput. Math. Appl.*, vol. 40, no. 4-5, pp. 491–512, 2000.

[28] Shampine, L.F., Gladwell, I. and Thompson, S.: *Solving ODEs with MATLAB.* Cambridge University Press, Cambridge, 2003.

[29] Gottlieb, D. and Lustman, L.: The spectrum of the Chebyshev collocation operator for the heat equation. *SIAM J. Numer. Anal.*, vol. 20, no. 5, pp. 909–921, 1983.

[30] Klopfenstein, R.W.: Numerical differentiation formulas for stiff systems of ordinary differential equations. *RCA Rev.*, vol. 32, pp. 447–462, 1971.

[31] Golub, G.H. and Van Loan, C.F.: *Matrix computations.* Johns Hopkins Studies in the Mathematical Sciences, 3rd edn. Johns Hopkins University Press, Baltimore, MD, 1996.

[32] López-Fernández, M., Palencia, C. and Schädle, A.: A spectral order method for inverting sectorial Laplace transforms. *SIAM J. Numer. Anal.*, vol. 44, no. 3, pp. 1332–1350 (electronic), 2006.

[33] Sheen, D., Sloan, I.H. and Thomée, V.: A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature. *IMA J. Numer. Anal.*, vol. 23, no. 2, pp. 269–299, 2003.

[34] Weideman, J.A.C.: Optimizing Talbot's contours for the inversion of the Laplace transform. *SIAM J. Numer. Anal.*, vol. 44, no. 6, pp. 2342–2362 (electronic), 2006.

[35] in't Hout, K.: ADI finite difference schemes for option pricing in the Heston model with correlation. *Submitted for publication on 20/11/2008.*

[36] Kudrna, V., Siyakatshana, N., Cermakova, J. and Machon, V.: General solution of the dispersion model for a one-dimensional stirred flow system using Danckwerts' boundary conditions. *Chem. Engng. Sci.*, vol. 59, no. 12, pp. 3013–3020, 2004.

[37] Danckwerts, P.: Continuous flow systems distribution of residence times. *Chem. Eng. Sci.*, vol. 2, no. 2, pp. 1–13, 1954.

[38] Carslaw, H.S. and Jaeger, J.C.: *Conduction of heat in solids.* Oxford Science Publications, 2nd edn. The Clarendon Press Oxford University Press, New York, 1988.

[39] Martin, A.: Interpretation of residence time distribution data. *Chem. Eng. Sci.*, vol. 55, no. 7, pp. 1987–1994, 2000.

[40] Trefethen, L.N., Weideman, J.A.C. and Schmelzer, T.: Talbot quadratures and rational approximations. *BIT*, vol. 46, no. 3, pp. 653–670, 2006.

[41] Laub, A.J.: *Matrix analysis for scientists & engineers.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

[42] Cash, J.R.: Efficient time integrators in the numerical method of lines. *J. Comput. Appl. Math.*, vol. 183, no. 2, pp. 259–274, 2005.

[43] Noble, B. and Daniel, J.W.: *Applied linear algebra.* 3rd edn. Prentice-Hall Inc., Englewood Cliffs, N.J., 1977.