Second Edition

Numerical Methods for Physics

Alejandro L. Garcia
San Jose State University

Prentice Hall, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Garcia, Alejandro L.,
Numerical methods for physics / Alejandro L. Garcia. – 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-906744-2
1. Mathematical physics. 2. Differential equations,
Partial–Numerical solutions. 3. Physics–Data processing. I.
Title.
QC20 .G37 2000
530.15–dc21

99-28552
CIP

Executive Editor: Alison Reeves
Editor-in-Chief: Paul F. Corey
Assistant Vice President of Production and Manufacturing: David W. Riccardi
Executive Managing Editor: Kathleen Schiaparelli
Assistant Managing Editor: Lisa Kinne
Production Editor: Linda DeLorenzo
Marketing Manager: Steven Sartori
Editorial Assistant: Gillian Buonanno
Manufacturing Manager: Trudy Pisciotti
Art Director: Jayne Conte
Cover Designer: Bruce Kenselaar
Cover Art: John Christiana

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-906744-2

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall (*Singapore*) Pte Ltd
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

Preface

When I was an undergraduate, computers were just beginning to be introduced into the university curriculum. Physics majors were expected to take a single semester of Pascal taught by the computer science department. We wrote programs to sort lists, process a payroll, and so forth, but were expected to acquire the specialized tools of scientific computing on our own. Most of us wasted many human and computer hours learning them by trial and error.

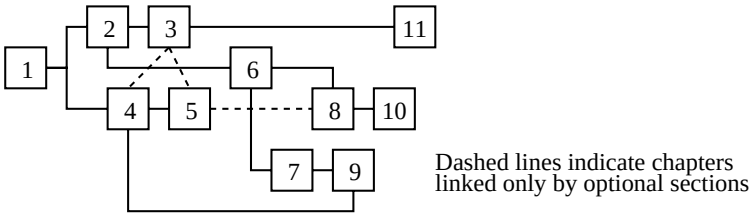
In recent years, many departments have added a computational physics course, taught by physicists, to their curricula. However, there is still considerable debate as to how this course should be organized. My philosophy is to use the upper division/graduate mathematical physics course as a model. Consider the following parallels between this text and a typical math physics book: A variety of numerical and analytical techniques used in physics are covered. Topics include ordinary and partial differential equations, linear algebra, Fourier transforms, integration, and probability. Because the text is written for physicists, these techniques are applied to solving realistic problems, many of which the students have encountered in other courses.

Numerical Methods for Physics is organized to cover what I believe are the most important, basic computational methods for physicists. The structure of the book differs considerably from the generic numerical analysis text. For example, about a third of the book is devoted to partial differential equations. This emphasis is natural considering the fundamental importance of Maxwell's equations, the Schrödinger equation, the Boltzmann equation, and so forth. Chapters 6 and 7 introduce some methods in computational fluid dynamics, an increasingly important topic in the fields of nonlinear physics, environmental physics, and astrophysics.

Numerical techniques may be classified as basic, advanced, and cutting edge. On the whole, this text covers only fundamental techniques; to work effectively with advanced numerical methods requires that the user first understand the basic algorithms. The discussion in the "Beyond This Chapter" section at the end of each chapter guides the reader to advanced algorithms and indicates when it is appropriate to use them. Unfortunately, the cutting edge moves so quickly that any attempt to summarize the latest algorithms would quickly be out of date.

The material in this text may be arranged in various ways to suit anything from a 10-week, upper-division class to a full-semester, graduate course. Most

chapters include optional sections that may be omitted without loss of continuity. Furthermore, entire chapters may be skipped; chapter interdependencies are indicated in the flow chart.



I have tried to present the algorithms in a clear, universal form that would allow the reader to easily implement them in *any* language. The programs are given in outline form in the main text with MATLAB and C++ listings in the appendices. In my classes, the students are allowed to use any language, yet I find that most end up using MATLAB. Its plotting utilities are particularly good—all the graphical results in the book were generated directly from the MATLAB programs. Advanced programmers (and students wishing to improve this skill) prefer using C++. FORTRAN versions of the programs, along with the MATLAB and C++ source code, are available online from Prentice Hall.

The over 250 exercises should be regarded as an essential part of the text. The time needed to do a problem ranges from 30 minutes to 2 days; in my classes, I assign about five exercises per week. Each exercise is labeled as:

[Pencil]	can be solved with pencil and paper.
[Computer]	requires using the computer.
[MATLAB]	best solved using MATLAB.
[C++]	best solved using C++.

While some texts emphasize month-long projects, I find that shorter exercises allow the class to move at a brisker pace, covering a wider variety of topics. Some instructors may wish to give one or two longer assignments, and many of the exercises may be expanded into such projects.

Readers familiar with the first edition will notice the following changes: C++ versions of the programs have been added, along with a new section (1.3) summarizing the language. The MATLAB programs have been updated to version 5. The discussion of derivatives has been moved from Chapter 1 to Chapter 2. A new section (6.3) has been added to Chapter 6. The discussion of hyperbolic partial differential equations has been collected into a new Chapter 7.

I wish to thank the people in my department, especially D. Strandburg, P. Hamill, A. Tucker, and J. Becker, for their strong support; my students and teaching assistants, J. Stroh, S. Moon, and D. Olson, who braved the rough waters of the early drafts; the National Science Foundation for its support of the computational physics program at San Jose State University; my editors at Prentice Hall and the technical staff of The MathWorks Inc. for their assistance; the National Oceanic and Atmospheric Administration Climate Monitoring and Diagnostics Laboratory for the CO_2 data used in Chapter 5. In addition, I

appreciate the comments of the following reviewers: David A. Boness, Seattle University; Wolfgang Christian, Davidson College; David M. Cook, Lawrence University; Harvey Gould, Clark University; Cleve Moler, The MathWorks, Inc; Cecile Penland, University of Colorado, Boulder; and Ross L. Spencer, Brigham Young University. Finally, I owe a special debt of gratitude to my entire family for their moral support as I wrote this book.

Alejandro L. Garcia

Dedicated to
Josefina Ovies García
and
Miriam González López

program to use this scheme and compare it with the others discussed in this section for the cases shown in Figures 7.3–7.7. For what values of τ is it stable? [Computer]

7.2 *PHYSICS OF TRAFFIC FLOW

Fluid Mechanics

In fluid mechanics the equations of motion are obtained by constructing equations of the form

$$\frac{\partial p}{\partial t} = -\nabla \cdot \mathbf{F}(p) \tag{7.29}$$

or in one dimension,

$$\frac{\partial p}{\partial t} = -\frac{\partial}{\partial x} F(p) \tag{7.30}$$

Here p is any one of the conserved quantities

$$p = \begin{cases} \text{mass density} \\ x, y \text{ or } z\text{−momentum density} \\ \text{energy density} \end{cases} \tag{7.31}$$

and F is

$$F(p) = \begin{cases} \text{mass flux} \\ x, y \text{ or } z\text{−momentum flux} \\ \text{energy flux} \end{cases} \tag{7.32}$$

that is, the corresponding flux.

While the equations for the momentum and energy are somewhat complicated, the equation for the mass density, ρ, is quite simple. The mass flux equals the mass density times the fluid velocity, v, so

$$\frac{\partial \rho(x,t)}{\partial t} = -\frac{\partial}{\partial x}\{\rho(x,t)v(x,t)\} \tag{7.33}$$

This equation is known as the *equation of continuity*. The equation for the momentum density may be rewritten as an equation for the velocity. This velocity equation involves the energy density (the coupling is in the pressure term), so we must solve the entire set of equations simultaneously.

The full set of hydrodynamics equations is called the *Navier-Stokes equations*. For a variety of reasons, these equations are usually not solved in their full form but rather with a number of approximations. Of course the approximations used depend on the problem at hand. For example, air is incompressible to a good approximation in many subsonic flows.

Traffic Flow

One of the simplest, nontrivial flows that may be studied involves fluids for which the velocity is only a function of density,

$$v(x,t) = v(\rho) \tag{7.34}$$

For example, suppose that the velocity of the fluid decreased linearly with increasing density as

$$v(\rho) = v_{\mathrm{m}}(1 - \rho/\rho_{\mathrm{m}}) \tag{7.35}$$

where $v_{\mathrm{m}} > 0$ is the maximum velocity and $\rho_{\mathrm{m}} > 0$ is the maximum density. What type of fluid behaves this way? One flow you are probably very familiar with is automobile traffic. The maximum velocity is the speed limit; if the density is near zero (few cars on the road), then the traffic moves at this speed. The maximum density, ρ_{m}, is achieved when the traffic is bumper-to-bumper. While on real highways the flow may not exactly obey Equation (7.35), it turns out to be a good first approximation.[65]

Our equation for the evolution of the density may be written as

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x}\left\{(\alpha + \frac{1}{2}\beta\rho)\rho\right\} \tag{7.36}$$

where $\alpha = v_{\mathrm{m}}$ and $\beta = -2v_{\mathrm{m}}/\rho_{\mathrm{m}}$. This equation is called the generalized inviscid Burger's equation. We obtain the standard inviscid Burger's equation when $\alpha = 0$ and $\beta = 1$. This equation has been studied extensively because it is the simplest nonlinear PDE with wave solutions.[16] Equations of this type appear frequently in nonlinear acoustics and shock wave theory.

Returning to our traffic model, we want to develop a method to solve the nonlinear PDE,

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x}\left\{\rho v(\rho)\right\} \tag{7.37}$$

Rewrite this equation as

$$\frac{\partial \rho}{\partial t} = -\left(\frac{d}{d\rho}\rho v(\rho)\right)\frac{\partial \rho}{\partial x} \tag{7.38}$$

or

$$\frac{\partial \rho}{\partial t} = -c(\rho)\frac{\partial \rho}{\partial x} \tag{7.39}$$

where $c(\rho) \equiv d(\rho v)/d\rho$. Using our linear function for $v(\rho)$ as given by Equation (7.35), we have

$$c(\rho) = v_{\mathrm{m}}(1 - 2\rho/\rho_{\mathrm{m}}) \tag{7.40}$$

Notice that $c(\rho)$ is also linear in ρ and takes the values $c(0) = v_{\mathrm{m}}$ and $c(\rho_{\mathrm{m}}) = -v_{\mathrm{m}}$. The function $c(\rho)$ is not the speed of the traffic, but rather is the speed at which disturbances (or waves) in the flow will travel. Since $c(\rho)$ may be both positive or negative, the waves may move in either direction. Note, however, that $c(\rho) \leq v(\rho)$, so the waves may never move faster than the cars.

Method of Characteristics

For $c(\rho)$ = constant, we have the advection equation for which we already know the solution. We may build an analytical solution to (7.39) from our knowledge of the solution of the advection equation by using the *method of*

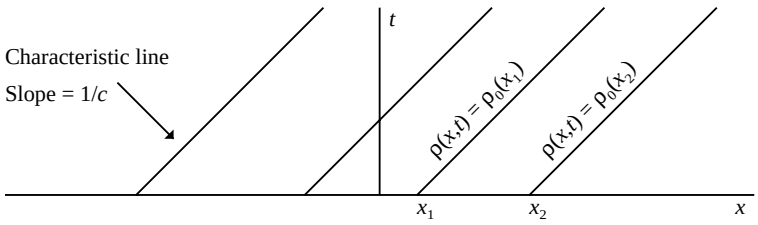


Figure 7.8: Sketch of the characteristic lines for the advection equation.

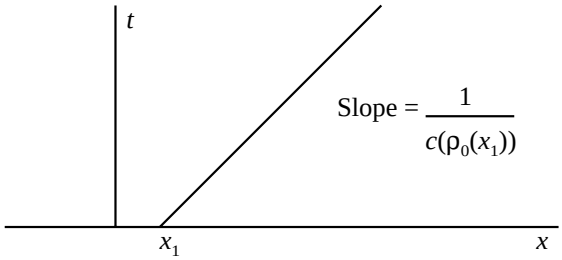


Figure 7.9: Sketch of a single characteristic line for the nonlinear traffic equation.

characteristics.[1] If you are not interested in learning this method, skim through the introduction to the stoplight problem (see Figures 7.11 and 7.12), and skip to its solution, Equation (7.46).

For a moment let's return to our solution of the linear advection PDE, Equation (7.11). We know that, with time, the initial condition, $\rho(x, t = 0) = \rho_0(x)$, is translated with speed c. Consider the sketch of the xt plane shown in Figure 7.8. Suppose that we draw a line with slope $dt/dx = 1/c$ from a point x_1 on the x-axis. This line will be a contour of constant ρ in the xt plane because the solution of the advection equation is just the initial condition displaced by a distance $\Delta x = c\Delta t$.

Now let's return to the nonlinear problem, Equation (7.39). In Figure 7.9 we draw the characteristic line from the point x_1; this line has slope $dt/dx = 1/c(\rho_0(x_1))$. Even in the nonlinear problem, the density is constant along this line. Here is the proof: For any function of two variables, the chain rule tells us that

$$\frac{d}{dt}f(x(t),t) = \frac{\partial}{\partial t}f(x(t),t) + \frac{dx}{dt}\frac{\partial}{\partial x}f(x(t),t) \tag{7.41}$$

Suppose that we vary x with t such that we move along the characteristic line. This means that $dt/dx = 1/c(\rho_0)$ or $dx/dt = c(\rho_0)$. Using the previous equation with $f(x,t) = \rho(x,t)$,

$$\frac{d}{dt}\rho(x(t),t) = \frac{\partial}{\partial t}\rho(x(t),t) + c(\rho_0(x))\frac{\partial}{\partial x}\rho(x(t),t) \tag{7.42}$$

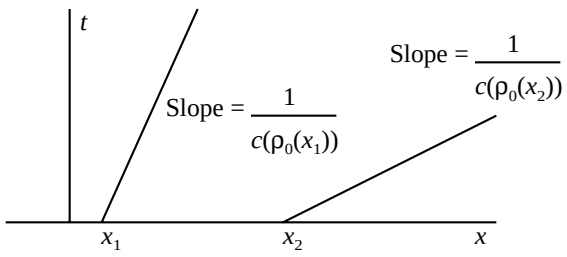


Figure 7.10: Sketch of various characteristic lines for the nonlinear traffic equation.

Yet from our original PDE, the right-hand side is zero, so

$$\frac{d}{dt}\rho(x(t),t) = 0 \qquad \text{(on the characteristic line)} \tag{7.43}$$

which completes the proof.

To use the method of characteristics to construct our solution, we draw a characteristic line from each point on the x-axis (Figure 7.10). You should think of these lines as forming a contour map of $\rho(x,t)$, since each line is a line of constant density.

Traffic at a Stoplight

Now to solve an actual traffic problem. The simplest problem we can solve is the initial distribution

$$\rho(x, t=0) = \rho_0(x) = \begin{cases} \rho_{\mathrm{m}} & x < 0 \\ 0 & x > 0 \end{cases} \tag{7.44}$$

that is, a step function (Figure 7.11). As a traffic problem, this could represent cars at a stoplight. Behind the light (which is at $x = 0$), the traffic is at its maximum density (bumper-to-bumper); there is no traffic on the other side of the light. At time $t = 0$ the light turns green and the cars are free to move. Intuitively, we know that not all cars start moving when the light turns green. The density decreases as the cars separate, but this effect propagates back into the stream of traffic with a finite wave speed (Figure 7.12). In fluid dynamics this is known as a rarefaction wave problem.

Let's start by drawing the characteristic lines on the positive x-axis. These lines will have slope $1/c(0) = 1/v_{\mathrm{m}}$. If we shade the region of constant density, we have the sketch shown in Figure 7.13. The first car through the light will move at the maximum velocity since there are no cars in front of it. The left border of the $\rho(x,t) = 0$ region is the location of the lead car.

Next, we add the characteristic lines for the points on the negative x-axis, as shown in Figure 7.14. Notice that most cars do not begin to move until long after the light has turned green. This is because the disturbance (or wave) can

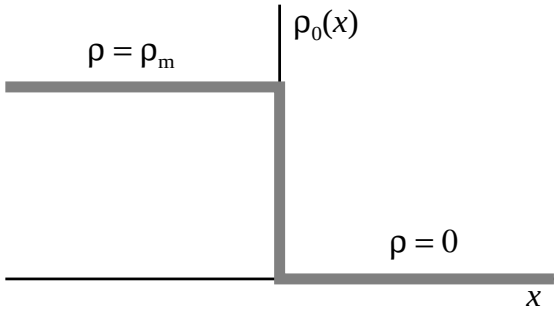


Figure 7.11: Initial density profile for traffic at a stoplight.

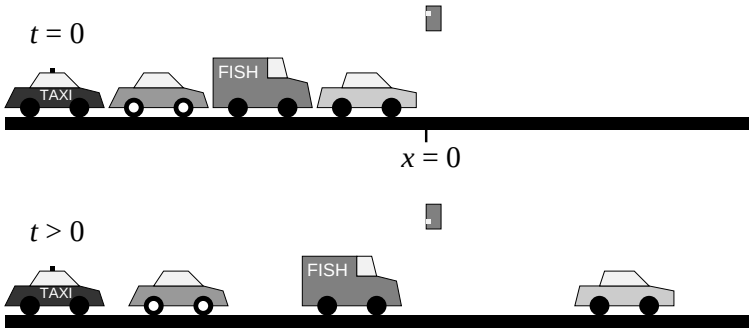


Figure 7.12: Traffic moving after a stoplight turns green. Notice that in the second frame the last car toward the rear has not moved.

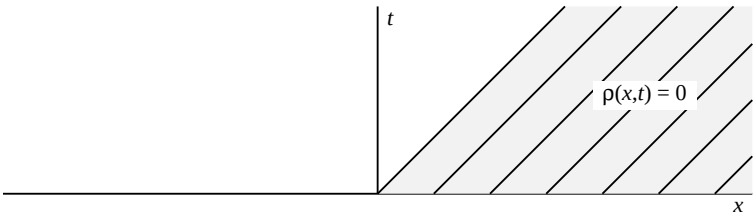


Figure 7.13: Partial construction of $\rho(x,t)$ in the xt plane using characteristic lines. In the shaded region the density $\rho(x,t)$ is zero. The left boundary of this region is given by the position of the lead car.

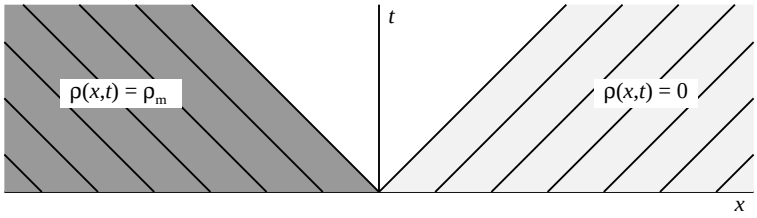


Figure 7.14: Partial construction of $\rho(x,t)$ in the xt plane using characteristic lines. In the shaded region on the left the density $\rho(x,t)$ is maximum (bumper-to-bumper).

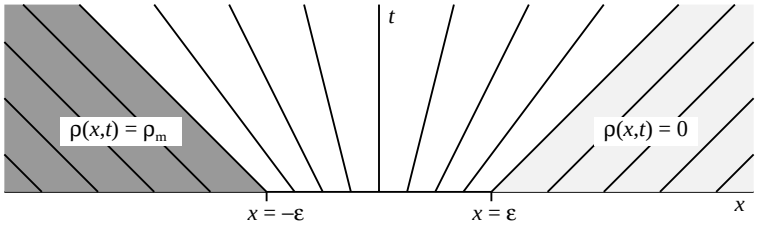


Figure 7.15: Characteristic lines for a continuous initial density profile. This density profile goes to a step function as $\epsilon \to 0$.

only move with velocity $c(\rho_{\mathrm{m}})$. For our linear relation between v and ρ, we have $c(\rho_{\mathrm{m}}) = -v_{\mathrm{m}}$.

To obtain all the characteristic lines we must remember that our initial condition is discontinuous. Suppose that we modified $\rho_0(x)$ so that it varied continuously from ρ_{m} to zero in a neighborhood of radius ϵ about $x = 0$. The slopes of the characteristic lines in this neighborhood would vary continuously from $1/v_{\mathrm{m}}$ to $-1/v_{\mathrm{m}}$ (Figure 7.15).

Taking the limit $\epsilon \to 0$, we have our final picture of the characteristic lines

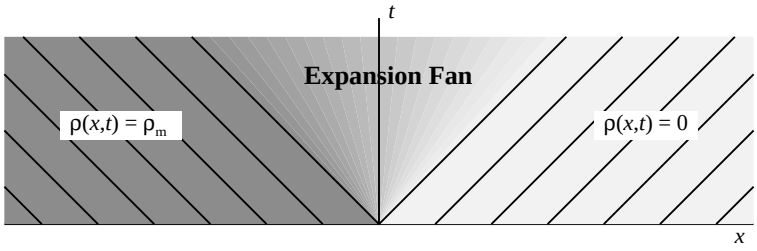


Figure 7.16: Construction of $\rho(x,t)$ in the xt plane using characteristic lines.

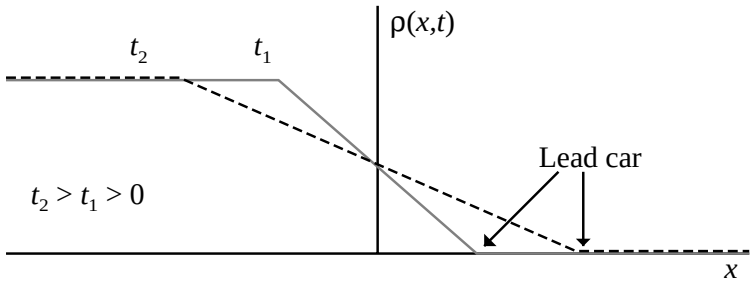


Figure 7.17: Traffic density, $\rho(x,t)$, as a function of position for various times.

as shown in Figure 7.16. The solution may be written as

$$\rho(x,t) = \begin{cases} \rho_{\rm m} & \text{for} \quad x \le -v_{\rm m}t \\ c^{-1}(x/t) & \text{for} \quad -v_{\rm m}t < x < v_{\rm m}t \\ 0 & \text{for} \quad x \ge v_{\rm m}t \end{cases} \tag{7.45}$$

where $c^{-1}(c(\rho)) = \rho$; that is, c^{-1} is the inverse function of $c(\rho)$. Using Equation (7.40) for $c(\rho)$ we have

$$\rho(x,t) = \begin{cases} \rho_{\rm m} & \text{for} \quad x \le -v_{\rm m}t \\ \frac{1}{2}\left(1 - \frac{x}{v_{\rm m}t}\right)\rho_{\rm m} & \text{for} \quad -v_{\rm m}t < x < v_{\rm m}t \\ 0 & \text{for} \quad x \ge v_{\rm m}t \end{cases} \tag{7.46}$$

Notice that in the region $-v_{\rm m}t < x < v_{\rm m}t$, the density varies linearly with position (Figure 7.17).

Traffic Program

Now that we have an analytical solution for a simple traffic problem, let's see how well our numerical methods can do. The equation of continuity is

$$\frac{\partial}{\partial t}\rho(x,t) = -\frac{\partial}{\partial x}F(\rho) \tag{7.47}$$

where the flow is $F(\rho) = \rho(x,t)v(\rho(x,t))$ and the velocity, $v(\rho)$, is given by Equation (7.35). The FTCS scheme for solving this equation is

$$\rho_i^{n+1} = \rho_i^n - \frac{\tau}{2h}(F_{i+1}^n - F_{i-1}^n) \tag{7.48}$$

where $F_i^n \equiv F(\rho_i^n)$. The Lax scheme uses the equation

$$\rho_i^{n+1} = \frac{1}{2}(\rho_{i+1}^n + \rho_{i-1}^n) - \frac{\tau}{2h}(F_{i+1}^n - F_{i-1}^n) \tag{7.49}$$

Table 7.2: Outline of program `traffic`, which computes the equation of continuity for traffic flow.

- Select numerical parameters (τ, h, etc.).
- Set initial condition (7.52) and periodic boundary conditions.
- Initialize plotting variables.
- Loop over desired number of steps.
  - Compute the flow, $F(\rho) = \rho(x,t)v(\rho(x,t))$.
  - Compute new values of density using:
    - FTCS scheme (7.48) **or**;
    - Lax scheme (7.49) **or**;
    - Lax-Wendroff scheme (7.50).
  - Record density for plotting.
  - Display snap-shot of density versus position. [MATLAB only]
- Graph density versus position and time as mesh plot.
- Graph contours of density versus position and time.

See pages 240 and 244 for program listings.

Finally, the Lax-Wendroff scheme uses

$$\rho_i^{n+1} = \rho_i^n - \frac{\tau}{2h}(F_{i+1}^n - F_{i-1}^n) + \frac{\tau^2}{2}\frac{1}{h}\left(c_{i+\frac{1}{2}}\frac{F_{i+1}^n - F_i^n}{h} - c_{i-\frac{1}{2}}\frac{F_i^n - F_{i-1}^n}{h}\right) \tag{7.50}$$

where

$$c_{i\pm\frac{1}{2}} \equiv c(\rho_{i\pm\frac{1}{2}}^n); \qquad \rho_{i\pm\frac{1}{2}}^n \equiv \frac{\rho_{i\pm1}^n + \rho_i^n}{2} \tag{7.51}$$

Notice how the last term of (7.50) is built: We would like to be able to evaluate the function $c(\rho)$ at values between grid points, that is, at $i+\frac{1}{2}$ and $i-\frac{1}{2}$. Since we know the value of ρ only at grid points, we estimate its value between grid points by using a simple average. We use this estimated value for $\rho_{i\pm\frac{1}{2}}$ to evaluate $c_{i\pm\frac{1}{2}}$.

The program called `traffic`, which implements these numerical schemes, is outlined in Table 7.2. As an initial condition we take a square pulse of the form

$$\rho(x, t=0) = \rho_0(x) = \begin{cases} \rho_{\rm m} & -L/4 < x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.52}$$

This initial value problem is similar to the stoplight problem considered above

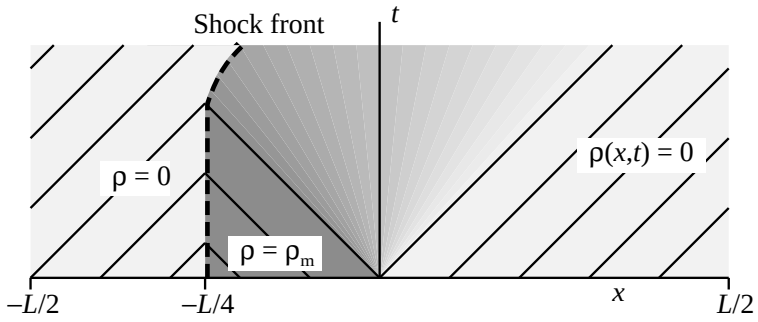


Figure 7.18: Characteristic lines for the finite pulse [see Equation (7.52)].

[see Equation (7.11)], except that the line of cars is of finite length. We take periodic boundary conditions so the problem resembles the start of a race on a circular track. From the solution to the stoplight problem, Equation (7.46), we expect the right side of the pulse to expand with the density varying linearly from $\rho_{\rm m}$ to zero.

The left edge of the pulse should not move until the density there drops below $\rho_{\rm max}$. The last car only begins moving when the traffic is no longer bumper-to-bumper. Figure 7.18 shows the characteristic lines for this problem; the discontinuity at $-L/4$ is a shock front. At the shock front, characteristic lines of high density and low density intersect, and the $\rho(x,t)$ is multivalued at the shock. The characteristic line solution is valid as long as we terminate the characteristic lines at the shock. Even if the initial condition is smoothed, this shock will develop since the slopes of low density and high density characteristics lines have opposite sign.

When the last car begins to move, the shock front also moves. Using the condition that the flux, $F(\rho)$, is a continuous function, we may compute the motion of the shock. Our formulation using characteristic lines may then be extended to complete the solution (see Exercise 7.13).

Running the `traffic` program using the FTCS method we obtain the results shown in Figure 7.19. Notice that while the FTCS method appears stable in this case, the solution is not at all satisfactory. The right edge of the pulse is curved, yet it should expand as a straight line (see Figure 7.17). Using the `traffic` program with the Lax method, we get the results shown in Figures 7.20 and 7.21. While the right edge is straighter, its slope is too large. Also the left edge is not maintained constant.

The Lax-Wendroff scheme does an excellent job, as shown in Figures 7.22 and 7.23. The latter is a contour plot of the density in the xt plane; compare this result with the characteristic lines shown in Figure 7.18. By running the program for more time steps, we can observe the evolution of the pulse after the last car starts to move (Figures 7.24 and 7.25). The shock at the left edge of the pulse moves, and its strength begins to decrease. Eventually the density

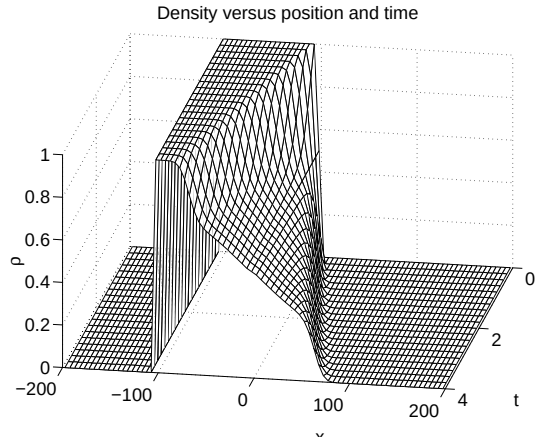


Figure 7.19: Mesh plot of density versus position and time from the `traffic` program using the FTCS method with 80 grid points and a time step of 0.2.

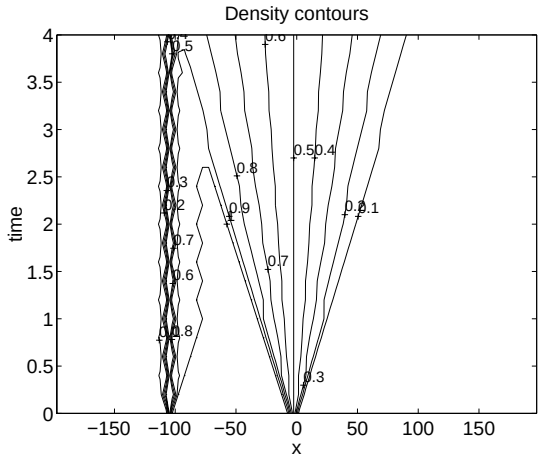


Figure 7.20: Contour plot of density versus position and time from the `traffic` program using the Lax method with 80 grid points and a time step of 0.2.

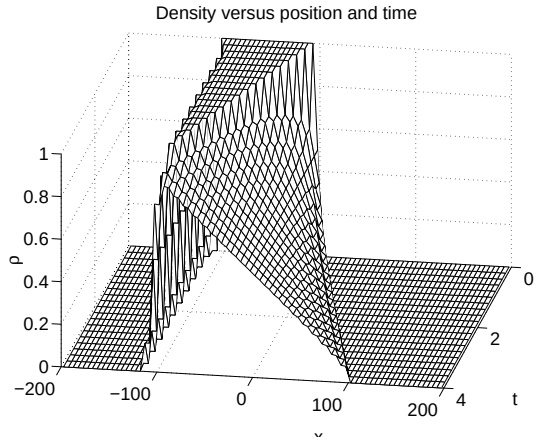


Figure 7.21: Mesh plot of density versus position and time from the `traffic` program using the Lax method with 80 grid points and a time step of 0.2.

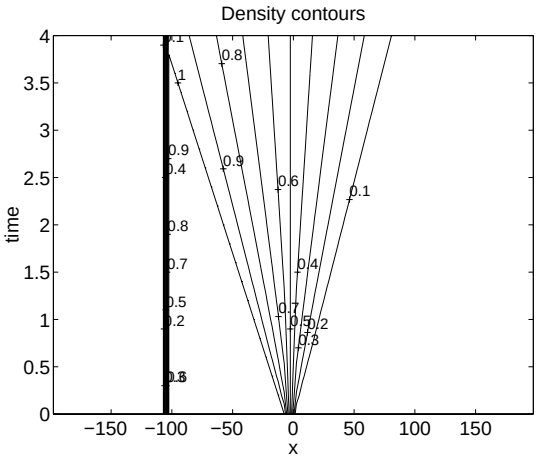


Figure 7.22: Contour plot of density versus position and time from the `traffic` program using the Lax-Wendroff method with 80 grid points and a time step of 0.2.

becomes uniform everywhere.

Shock waves in real traffic are very dangerous. Drivers have finite reaction times, so sudden changes in traffic density can cause accidents. In our traffic model, the local density determines the traffic velocity. Fortunately, under normal visibility conditions, drivers adjust their speed by judging the global traffic conditions (i.e., they look at more than just the car in front of them). This fact introduces a diffusion term into the model that smooths the discontinuous shock fronts.

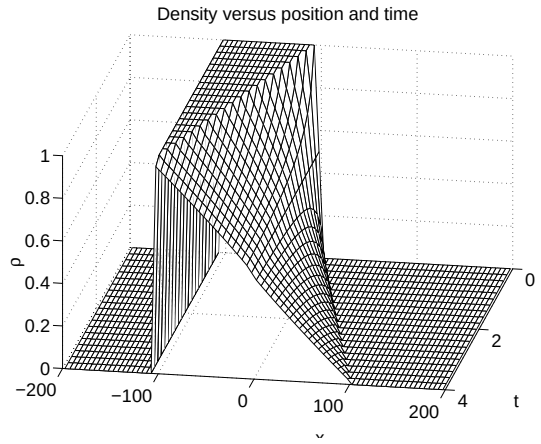


Figure 7.23: Mesh plot of density versus position and time from the `traffic` program using the Lax-Wendroff method with 80 grid points and a time step of 0.2.

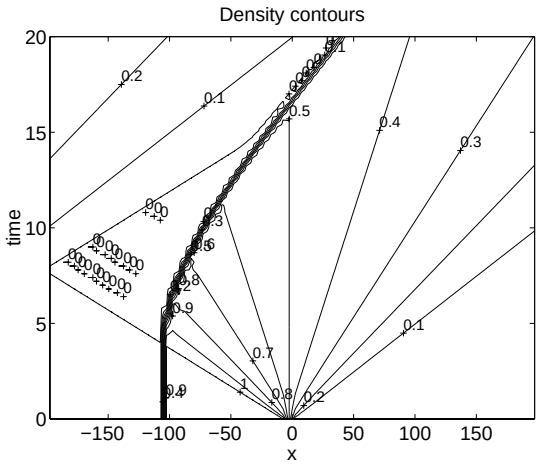


Figure 7.24: Contour plot of density versus position and time from the `traffic` program using the Lax-Wendroff method. Parameters are the same as in Figure 7.22 except the simulation is run five times longer (i.e., the number of time steps is five times larger).

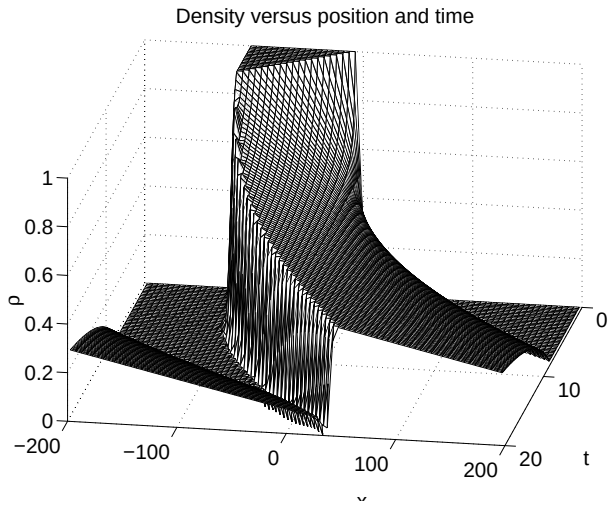


Figure 7.25: Mesh plot of density versus position and time from the `traffic` program using the Lax-Wendroff method. The contour plot for this run is shown in Figure 7.24.

EXERCISES

7. The flow of traffic is $F(x,t) = \rho(x,t)v(\rho)$. For the stoplight problem, obtain an expression for $F(x,t)$ using the solution (7.46) and $v(\rho) = v_{\rm m}(1-\rho/\rho_{\rm m})$. Sketch $F(x,t)$ versus x for $t > 0$, and show that it is maximum at $x = 0$ (i.e., at the light). [Pencil]

8. Call $x_{\rm c}(t)$ the position of a given car; then

$$\frac{dx_{\rm c}}{dt} = v(\rho(x_{\rm c}(t),t)) \tag{7.53}$$

(a) Show that

$$x_{\rm c}(t) = \begin{cases} x_{\rm c}(0) & t < -x_c(0)/v_{\rm m} \\ v_{\rm m}t - 2\sqrt{-x_{\rm c}(0)v_{\rm m}t} & t > -x_c(0)/v_{\rm m} \end{cases} \tag{7.54}$$

by using the solution to the stoplight problem, Equation (7.46). [Pencil] (b) Plot the trajectories $x_{\rm c}$ in the xt plane for various $x_c(0)$. [Computer] (c) Plot the time it takes for a car to reach the intersection as a function of $|x_c(0)|$. [Computer]

9. After a time t, the total amount of traffic that has passed through the light is $N(t) = \int_0^\infty \rho(x,t)\,dx$. Show that $N(t) = \int_0^t F(x=0,t)\,dt$, where $F(x,t) = \rho(x,t)v(x,t)$ is the flow. [Pencil]

10. Modify the `traffic` program so that it uses a Gaussian pulse of width $L/4$ as an initial distribution for the density. Center the pulse at $x = 0$ with $\rho(0,0) = \rho_{\rm m}$. Show how the density evolves with time. Explain why one side of the pulse expands while the other contracts [remember that the wave speed, $c(\rho)$, is not a constant]. Describe what a driver on this racetrack will experience. [Computer]

11. Modify the `traffic` program so that it uses the initial condition

$$\rho(x, t = 0) = \frac{\rho_{\rm m}}{2}[1 + \cos(4\pi x/L)]$$

(a) Plot the density versus position for a variety of times and show that the cosine wave turns into a sawtooth wave. In nonlinear acoustics this is referred to as an N-wave. If you have ever been to a very loud rock concert, you may have heard one of these. (b) Modify your program to compute the spatial power spectrum of the density (see Section 5.2). Remove the zero wave number component. Initially, the spectrum will contain a single peak, but in time other peaks appear. Use a mesh plot to graph the spectrum versus wave number and time. [Computer]

12. Suppose that we have a uniform density of traffic with a small congested area. Modify the `traffic` program so that it uses the initial condition

$$\rho(x, t = 0) = \rho_0[1 + \alpha \exp(-x^2/2\sigma^2)]$$

where $\alpha = 1/5$, $\sigma = L/8$, and ρ_0 are a constants. (a) Show that for light traffic (e.g., $\rho_0 = \rho_{\rm m}/4$) the perturbation moves forward. What is its speed? (b) Show that for heavy traffic (e.g., $\rho_0 = 3\rho_{\rm m}/4$) the perturbation moves backward. Interpret this result physically. (c) Show that for $\rho_0 = \rho_{\rm m}/2$ the perturbation is almost stationary; it drifts and distorts slightly. [Computer]

13. Call $x_{\rm s}(t)$ the position of the shock wave (see Figures 7.18 and 7.24). The velocity of the shock is given by

$$\frac{dx_{\rm s}}{dt} = \frac{F(\rho_+) - F(\rho_-)}{\rho_+ - \rho_-} \tag{7.55}$$

where $F(x, t) = \rho(x, t)v(x, t)$ is the flow and $\rho_\pm = \lim_{\epsilon \to 0} \rho(x_{\rm s} \pm \epsilon)$, that is, the density on each side of the shock front. (a) Show that

$$\frac{dx_{\rm s}}{dt} = \frac{1}{2}(c(\rho_+) + c(\rho_-)) \tag{7.56}$$

when v is linear in the density. [Pencil] (b) Use the density profile computed by the `traffic` program to compute $x_{\rm s}(t)$ given that $x_{\rm s}(0) = -L/4$. Compare your results with the locations of steep gradients in the contour plot produced by `traffic`. [Computer]

BEYOND THIS CHAPTER

In Section 7.2 the method of characteristics is used to obtain an analytical solution to the generalized Burger's equation. The method of characteristics may also be implemented as a numerical scheme for solving hyperbolic equations. For the wave equation we have two sets of characteristic lines (left- and right-moving waves). For more complicated problems (e.g., Euler equations in fluid mechanics) these characteristic lines are computed numerically as trajectories of a nonlinear ODE.[73]

One of the principal difficulties with numerically solving hyperbolic equations is the formation of shocks. At a shock the solution is discontinuous and our PDE description breaks down. One way to treat the problem is to use an

uneven grid and concentrate grid points at the location of the shock. Shock-capturing methods automatically adjust the grid spacing to accomplish this. See Anderson et al. [10] and Fletcher [47] for an extensive discussion of finite difference methods for solving hyperbolic equations. For a presentation of the hydrodynamic equations suitable for a physicist, see Tritton [128].

APPENDIX A: MATLAB LISTINGS

Listing 7A.1 Program `advect`. Solves the advection equation using various numerical schemes.

```
% advect - Program to solve the advection equation
% using the various hyperbolic PDE schemes
clear all;  help advect;  % Clear memory and print header

%* Select numerical parameters (time step, grid spacing, etc.).
method = menu('Choose a numerical method:', ...
       'FTCS','Lax','Lax-Wendroff');
N = input('Enter number of grid points: ');
L = 1.;     % System size
h = L/N;    % Grid spacing
c = 1;      % Wave speed
fprintf('Time for wave to move one grid spacing is %g\n',h/c);
tau = input('Enter time step: ');
coeff = -c*tau/(2.*h);  % Coefficient used by all schemes
coefflw = 2*coeff^2;    % Coefficient used by L-W scheme
fprintf('Wave circles system in %g steps\n',L/(c*tau));
nStep = input('Enter number of steps: ');

%* Set initial and boundary conditions.
sigma = 0.1;              % Width of the Gaussian pulse
k_wave = pi/sigma;        % Wave number of the cosine
x = ((1:N)-1/2)*h - L/2;  % Coordinates of grid points
% Initial condition is a Gaussian-cosine pulse
a = cos(k_wave*x) .* exp(-x.^2/(2*sigma^2));
% Use periodic boundary conditions
ip(1:(N-1)) = 2:N;  ip(N) = 1;   % ip = i+1 with periodic b.c.
im(2:N) = 1:(N-1);  im(1) = N;   % im = i-1 with periodic b.c.

%* Initialize plotting variables.
iplot = 1;          % Plot counter
aplot(:,1) = a(:);  % Record the initial state
tplot(1) = 0;       % Record the initial time (t=0)
nplots = 50;        % Desired number of plots
plotStep = nStep/nplots; % Number of steps between plots

%* Loop over desired number of steps.
for iStep=1:nStep  %% MAIN LOOP %%
```

```
  %* Compute new values of wave amplitude using FTCS,
  %  Lax or Lax-Wendroff method.
  if( method == 1 )      %%% FTCS method %%%
    a(1:N) = a(1:N) + coeff*(a(ip)-a(im));
  elseif( method == 2 )  %%% Lax method %%%
    a(1:N) = .5*(a(ip)+a(im)) + coeff*(a(ip)-a(im));
  else                   %%% Lax-Wendroff method %%%
    a(1:N) = a(1:N) + coeff*(a(ip)-a(im)) + ...
        coefflw*(a(ip)+a(im)-2*a(1:N));
  end

  %* Periodically record a(t) for plotting.
  if( rem(iStep,plotStep) < 1 )  % Every plot_iter steps record
    iplot = iplot+1;
    aplot(:,iplot) = a(:);       % Record a(i) for ploting
    tplot(iplot) = tau*iStep;
    fprintf('%g out of %g steps completed\n',iStep,nStep);
  end
end

%* Plot the initial and final states.
figure(1); clf;  % Clear figure 1 window and bring forward
plot(x,aplot(:,1),'-',x,a,'--'); legend('Initial','Final');
xlabel('x');  ylabel('a(x,t)');
pause(1);    % Pause 1 second between plots

%* Plot the wave amplitude versus position and time
figure(2); clf;  % Clear figure 2 window and bring forward
mesh(tplot,x,aplot); ylabel('Position');  xlabel('Time');
zlabel('Amplitude');
view([-70 50]);  % Better view from this angle
```

Listing 7A.2 Program `traffic`. Solves the equation of continuity for traffic flow.

```
% traffic - Program to solve the generalized Burger
% equation for the traffic at a stop light problem
clear all;  help traffic; % Clear memory and print header

%* Select numerical parameters (time step, grid spacing, etc.).
method = menu('Choose a numerical method:', ...
       'FTCS','Lax','Lax-Wendroff');
N = input('Enter the number of grid points: ');
L = 400;      % System size (meters)
h = L/N;      % Grid spacing for periodic boundary conditions
v_max = 25;   % Maximum car speed (m/s)
fprintf('Suggested timestep is %g\n',h/v_max);
tau = input('Enter time step (tau): ');
```

```
fprintf('Last car starts moving after %g steps\n', ...
                                  (L/4)/(v_max*tau));
nstep = input('Enter number of steps: ');
coeff = tau/(2*h);        % Coefficient used by all schemes
coefflw = tau^2/(2*h^2);  % Coefficient used by Lax-Wendroff

%* Set initial and boundary conditions
rho_max = 1.0;                  % Maximum density
Flow_max = 0.25*rho_max*v_max;  % Maximum Flow
% Initial condition is a square pulse from x = -L/4 to x = 0
rho = zeros(1,N); for i=round(N/4):round(N/2-1)
  rho(i) = rho_max;    % Max density in the square pulse
end
rho(round(N/2)) = rho_max/2;  % Try running without this line
% Use periodic boundary conditions
ip(1:N) = (1:N)+1;  ip(N) = 1;   % ip = i+1 with periodic b.c.
im(1:N) = (1:N)-1;  im(1) = N;   % im = i-1 with periodic b.c.

%* Initialize plotting variables.
iplot = 1;
xplot = ((1:N)-1/2)*h - L/2;  % Record x scale for plot
rplot(:,1) = rho(:);          % Record the initial state
tplot(1) = 0;
figure(1); clf;  % Clear figure 1 window and bring forward

%* Loop over desired number of steps.
for istep=1:nstep

  %* Compute the flow = (Density)*(Velocity)
  Flow = rho .* (v_max*(1 - rho/rho_max));

  %* Compute new values of density using FTCS,
  %  Lax or Lax-Wendroff method.
  if( method == 1 )      %%% FTCS method %%%
    rho(1:N) = rho(1:N) - coeff*(Flow(ip)-Flow(im));
  elseif( method == 2 )  %%% Lax method %%%
    rho(1:N) = .5*(rho(ip)+rho(im)) ...
                  - coeff*(Flow(ip)-Flow(im));
  else                   %%% Lax-Wendroff method %%%
    cp = v_max*(1 - (rho(ip)+rho(1:N))/rho_max);
    cm = v_max*(1 - (rho(1:N)+rho(im))/rho_max);
    rho(1:N) = rho(1:N) - coeff*(Flow(ip)-Flow(im)) ...
             + coefflw*(cp.*(Flow(ip)-Flow(1:N)) ...
                      - cm.*(Flow(1:N)-Flow(im)));
  end

  %* Record density for plotting.
  iplot = iplot+1;
  rplot(:,iplot) = rho(:);
  tplot(iplot) = tau*istep;
```

```
  %* Display snap-shot of density versus position
  plot(xplot,rho,'-',xplot,Flow/Flow_max,'--');
  xlabel('x'); ylabel('Density and Flow');
  legend('\rho(x,t)','F(x,t)');
  axis([-L/2, L/2, -0.1, 1.1]);
  drawnow;
end

%* Graph density versus position and time as wire-mesh plot
figure(1); clf;  % Clear figure 1 window and bring forward
mesh(tplot,xplot,rplot) xlabel('t'); ylabel('x'); zlabel('\rho');
title('Density versus position and time');
view([100 30]);  % Rotate the plot for better view point
pause(1);    % Pause 1 second between plots

%* Graph contours of density versus position and time.
figure(2); clf;   % Clear figure 2 window and bring forward
% Use rot90 function to graph t vs x since
% contour(rplot) graphs x vs t.
clevels = 0:(0.1):1;   % Contour levels
cs = contour(xplot,tplot,flipud(rot90(rplot)),clevels);
clabel(cs);            % Put labels on contour levels
xlabel('x');  ylabel('time');  title('Density contours');
```

APPENDIX B: C++ LISTINGS

Listing 7B.1 Program `advect`. Solves the advection equation using various numerical schemes.

```
// advect - Program to solve the advection equation
// using the various hyperbolic PDE schemes
#include "NumMeth.h"

void main() {

  //* Select numerical parameters (time step, grid spacing, etc.).
  cout << "Choose a numerical method: 1) FTCS, 2) Lax, 3) Lax-Wendroff : ";
  int method; cin >> method;
  cout << "Enter number of grid points: "; int N; cin >> N;
  double L = 1.;     // System size
  double h = L/N;    // Grid spacing
  double c = 1;      // Wave speed
  cout << "Time for wave to move one grid spacing is " << h/c << endl;
  cout << "Enter time step: "; double tau; cin >> tau;
  double coeff = -c*tau/(2.*h);    // Coefficient used by all schemes
```

```
double coefflw = 2*coeff*coeff;  // Coefficient used by L-W scheme
cout << "Wave circles system in " << L/(c*tau) << " steps" << endl;
cout << "Enter number of steps: "; int nStep; cin >> nStep;

//* Set initial and boundary conditions.
const double pi = 3.141592654;
double sigma = 0.1;              // Width of the Gaussian pulse
double k_wave = pi/sigma;        // Wave number of the cosine
Matrix x(N), a(N), a_new(N);
int i,j;
for( i=1; i<=N; i++ ) {
  x(i) = (i-0.5)*h - L/2;  // Coordinates of grid points
  // Initial condition is a Gaussian-cosine pulse
  a(i) = cos(k_wave*x(i)) * exp(-x(i)*x(i)/(2*sigma*sigma));
}
// Use periodic boundary conditions
int *ip, *im;  ip = new int [N+1];  im = new int [N+1];
for( i=2; i<N; i++ ) {
  ip[i] = i+1;    // ip[i] = i+1 with periodic b.c.
  im[i] = i-1;    // im[i] = i-1 with periodic b.c.
}
ip[1] = 2;  ip[N] = 1;
im[1] = N;  im[N] = N-1;

//* Initialize plotting variables.
int iplot = 1;          // Plot counter
int nplots = 50;        // Desired number of plots
double plotStep = ((double)nStep)/nplots;
Matrix aplot(N,nplots+1), tplot(nplots+1);
tplot(1) = 0;       // Record the initial time (t=0)
for( i=1; i<=N; i++ )
  aplot(i,1) = a(i);  // Record the initial state

//* Loop over desired number of steps.
int iStep;
for( iStep=1; iStep<=nStep; iStep++ ) {

  //* Compute new values of wave amplitude using FTCS,
  //  Lax or Lax-Wendroff method.
  if( method == 1 )      ////// FTCS method //////
    for( i=1; i<=N; i++ )
      a_new(i) = a(i) + coeff*( a(ip[i])-a(im[i]) );
  else if( method == 2 )  ////// Lax method //////
    for( i=1; i<=N; i++ )
      a_new(i) = 0.5*( a(ip[i])+a(im[i]) ) +
                 coeff*( a(ip[i])-a(im[i]) );
  else                    ////// Lax-Wendroff method //////
    for( i=1; i<=N; i++ )
      a_new(i) = a(i) + coeff*( a(ip[i])-a(im[i]) ) +
                 coefflw*( a(ip[i])+a(im[i])-2*a(i) );
```

```
    a = a_new;   // Reset with new amplitude values

    //* Periodically record a(t) for plotting.
    if( fmod((double)iStep,plotStep) < 1 ) {
      iplot++;
      tplot(iplot) = tau*iStep;
      for( i=1; i<=N; i++ )
        aplot(i,iplot) = a(i);       // Record a(i) for ploting
      cout << iStep << " out of " << nStep << " steps completed" << endl;
    }
  }
  nplots = iplot;   // Actual number of plots recorded

  //* Print out the plotting variables: x, a, tplot, aplot
  ofstream xOut("x.txt"), aOut("a.txt"),
          tplotOut("tplot.txt"), aplotOut("aplot.txt");
  for( i=1; i<=N; i++ ) {
    xOut << x(i) << endl;
    aOut << a(i) << endl;
    for( j=1; j<nplots; j++ )
      aplotOut << aplot(i,j) << ", ";
    aplotOut << aplot(i,nplots) << endl;
  }
  for( i=1; i<=nplots; i++ )
    tplotOut << tplot(i) << endl;

  delete [] ip, im;  // Release allocated memory
}
/***** To plot in MATLAB; use the script below ********************
%* Plot the initial and final states.
load x.txt; load a.txt; load tplot.txt; load aplot.txt;
figure(1); clf;  % Clear figure 1 window and bring forward
plot(x,aplot(:,1),'-',x,a,'--'); legend('Initial','Final');
xlabel('x');  ylabel('a(x,t)');
pause(1);    % Pause 1 second between plots
%* Plot the wave amplitude versus position and time
figure(2); clf;  % Clear figure 2 window and bring forward
mesh(tplot,x,aplot); ylabel('Position');  xlabel('Time');
zlabel('Amplitude');
view([-70 50]);  % Better view from this angle
*****************************************************************/
```

Listing 7B.2 Program `traffic`. Solves the equation of continuity for traffic flow.

```
// traffic - Program to solve the generalized Burger
// equation for the traffic at a stop light problem
#include "NumMeth.h"
```

```
void main() {

  //* Select numerical parameters (time step, grid spacing, etc.).
  cout << "Choose a numerical method: 1) FTCS, 2) Lax, 3) Lax-Wendroff : ";
  int method; cin >> method;
  cout << "Enter the number of grid points: "; int N; cin >> N;
  double L = 400;      // System size (meters)
  double h = L/N;      // Grid spacing for periodic boundary conditions
  double v_max = 25;   // Maximum car speed (m/s)
  cout << "Suggested timestep is " << h/v_max << endl;
  cout << "Enter time step (tau): "; double tau; cin >> tau;
  cout << "Last car starts moving after "
       << (L/4)/(v_max*tau) << " steps" << endl;
  cout << "Enter number of steps: "; int nStep; cin >> nStep;
  double coeff = tau/(2*h);          // Coefficient used by all schemes
  double coefflw = tau*tau/(2*h*h);  // Coefficient used by Lax-Wendroff
  double cp, cm;       // Variables used by Lax-Wendroff

  //* Set initial and boundary conditions
  double rho_max = 1.0;                  // Maximum density
  double Flow_max = 0.25*rho_max*v_max;  // Maximum Flow
  // Initial condition is a square pulse from x = -L/4 to x = 0
  Matrix rho(N), rho_new(N);
  int i,j, iBack = N/4, iFront = N/2 - 1;
  for( i=1; i<=N; i++ )
    if( iBack <= i && i <= iFront ) rho(i) = rho_max;
    else rho(i) = 0.0;
  rho(iFront+1) = rho_max/2;  // Try running without this line
  // Use periodic boundary conditions
  int *ip, *im;  ip = new int [N+1];  im = new int [N+1];
  for( i=2; i<N; i++ ) {
    ip[i] = i+1;    // ip[i] = i+1 with periodic b.c.
    im[i] = i-1;    // im[i] = i-1 with periodic b.c.
  }
  ip[1] = 2;  ip[N] = 1;
  im[1] = N;  im[N] = N-1;

  //* Initialize plotting variables.
  int iplot = 1;
  Matrix tplot(nStep+1), xplot(N), rplot(N,nStep+1);
  tplot(1) = 0.0;             // Record initial time
  for( i=1; i<=N; i++ ) {
    xplot(i) = (i - 0.5)*h - L/2;  // Record x scale for plot
    rplot(i,1) = rho(i);           // Record the initial state
  }

  //* Loop over desired number of steps.
  Matrix Flow(N);
  int iStep;
```

```
  for( iStep=1; iStep<=nStep; iStep++ ) {

    //* Compute the flow = (Density)*(Velocity)
    for( i=1; i<=N; i++ )
      Flow(i) = rho(i) * (v_max*(1.0 - rho(i)/rho_max));

    //* Compute new values of density using FTCS,
    //  Lax or Lax-Wendroff method.
    if( method == 1 )       ////// FTCS method //////
      for( i=1; i<=N; i++ )
        rho_new(i) = rho(i) - coeff*(Flow(ip[i])-Flow(im[i]));
    else if( method == 2 )  ////// Lax method //////
      for( i=1; i<=N; i++ )
        rho_new(i) = 0.5*(rho(ip[i])+rho(im[i]))
                   - coeff*(Flow(ip[i])-Flow(im[i]));
    else                    ////// Lax-Wendroff method //////
      for( i=1; i<=N; i++ ) {
        cp = v_max*(1 - (rho(ip[i])+rho(i))/rho_max);
        cm = v_max*(1 - (rho(i)+rho(im[i]))/rho_max);
        rho_new(i) = rho(i) - coeff*(Flow(ip[i])-Flow(im[i]))
             + coefflw*(cp*(Flow(ip[i])-Flow(i))
                      - cm*(Flow(i)-Flow(im[i])));
      }
    // Reset with new density values
    rho = rho_new;

    //* Record density for plotting.
    cout << "Finished " << iStep << " of " << nStep << " steps" << endl;
    iplot++;
    tplot(iplot) = tau*iStep;
    for( i=1; i<=N; i++ )
      rplot(i,iplot) = rho(i);

  }
  int nplots = iplot;     // Number of plots recorded

  //* Print out the plotting variables: tplot, xplot, rplot
  ofstream tplotOut("tplot.txt"), xplotOut("xplot.txt"),
           rplotOut("rplot.txt");
  for( i=1; i<=nplots; i++ )
    tplotOut << tplot(i) << endl;
  for( i=1; i<=N; i++ ) {
    xplotOut << xplot(i) << endl;
    for( j=1; j<nplots; j++ )
      rplotOut << rplot(i,j) << ", ";
    rplotOut << rplot(i,nplots) << endl;
  }

  delete [] ip, im;
}
```

```
/***** To plot in MATLAB; use the script below ********************
load tplot.txt; load xplot.txt; load rplot.txt;
%* Graph density versus position and time as wire-mesh plot
figure(1); clf;  % Clear figure 1 window and bring forward
mesh(tplot,xplot,rplot) xlabel('t'); ylabel('x'); zlabel('\rho');
title('Density versus position and time');
view([100 30]);  % Rotate the plot for better view point
pause(1);    % Pause 1 second between plots
%* Graph contours of density versus position and time.
figure(2); clf;   % Clear figure 2 window and bring forward
% Use rot90 function to graph t vs x since
% contour(rplot) graphs x vs t.
clevels = 0:(0.1):1;   % Contour levels
cs = contour(xplot,tplot,flipud(rot90(rplot)),clevels);
clabel(cs);            % Put labels on contour levels
xlabel('x');  ylabel('time');  title('Density contours');
*****************************************************************/
```