1

An Introduction to the Method of Lines[1]

The chapters in this book pertain particularly to mathematical models expressed as *partial differential equations* (PDEs). The computer-based numerical solution of the PDE models is implemented primarily through the *method of lines* (MOL). We therefore start with this chapter, which is an introduction to the MOL. Although the reader may be familiar with the MOL, we suggest reading this chapter since it describes some aspects and details of our use of the MOL that appear in the subsequent chapters. We start with some basic features of PDEs.

SOME PDE BASICS

Our physical world is most generally described in scientific and engineering terms with respect to three-dimensional (3D) space and time, which we abbreviate as *spacetime*. PDEs provide a mathematical description of physical spacetime, and they are therefore among the most widely used forms of mathematics. As a consequence, methods for the solution of PDEs, such as the MOL, are of broad interest in science and engineering.

As a basic illustrative example of a PDE, we consider

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \tag{1.1}$$

where

- u dependent variable (dependent on x and t)
- t independent variable representing time
- x independent variable representing one dimension of 3D space
- D constant explained next

Note that Eq. (1.1) has two independent variables, x and t, which is the reason it is classified as a PDE (any differential equation with more than one independent

[1] This chapter is based on an article that originally appeared in the online encyclopedia *Scholarpedia* [1].

variable is a PDE). A differential equation with only one independent variable is generally termed an *ordinary differential equation* (ODE); we will consider ODEs later as part of the MOL.

Equation (1.1) is termed the *diffusion equation*. When applied to heat transfer, it is *Fourier's second law*; the dependent variable u is temperature and D is the *thermal diffusivity*. When Eq. (1.1) is applied to mass diffusion, it is *Fick's second law*; u is mass concentration and D is the *coefficient of diffusion* or the *diffusivity*.

$\partial u/\partial t$ is a partial derivative of u with respect to t (x is held constant when taking this partial derivative, which is why *partial* is used to describe this derivative). Equation (1.1) is *first order in t* since the highest-order partial derivative in t is first order; it is *second order in x* since the highest-order derivative in x is second order. Equation (1.1) is *linear* or *first degree* since all of the terms are to the first power (note that *order* and *degree* can easily be confused).

INITIAL AND BOUNDARY CONDITIONS

Before we consider a solution to Eq. (1.1), we must specify some *auxiliary conditions* to complete the statement of the PDE problem. The number of required auxiliary conditions is determined by the *highest-order derivative in each independent variable*. Since Eq. (1.1) is first order in t and second order in x, it requires one auxiliary condition in t and two auxiliary conditions in x. To have a complete, *well-posed* problem, some additional conditions may have to be included, for example, that specify valid ranges for coefficients. However, this is a more advanced topic and will not be developed further here.

t is termed an *initial-value variable* and therefore requires *one initial condition* (IC). It is an initial-value variable since it starts at an initial value, t_0, and moves forward over a *finite interval* $t_0 \le t \le t_f$ or a *semi-infinite interval* $t_0 \le t \le \infty$ without any additional conditions being imposed. Typically in a PDE application, the initial-value variable is time, as in the case of Eq. (1.1).

x is termed a *boundary-value variable* and therefore requires *two boundary conditions* (BCs). It is a boundary-value variable since it varies over a *finite interval* $x_0 \le x \le x_f$, a *semi-infinite interval* $x_0 \le x \le \infty$, or a *fully infinite interval* $-\infty \le x \le \infty$, and at *two different values of x*, conditions are imposed on u in Eq. (1.1). Typically, the two values of x correspond to boundaries of a physical system, and hence the name *boundary conditions*.

As examples of auxiliary conditions for Eq. (1.1),

- An IC could be

$$u(x, t = 0) = u_0 \tag{1.2}$$

  where u_0 is a given function of x (typically a constant) for $x_0 \le x \le x_f$.
- Two BCs could be

$$u(x = x_0, t) = u_b \tag{1.3a}$$

$$\frac{\partial u(x = x_f, t)}{\partial x} = 0 \tag{1.3b}$$

  where u_b is a given boundary (constant) value of u for all t.

- Another common possibility is where the IC is given as earlier and the BCs are $u(x = x_0, t) = f_0(t)$ and $u_x(x = x_f, t) = f_b(t)$.

An important consideration is the possibility of *discontinuities at the boundaries*, produced, for example, by differences in ICs and BCs at the boundaries, which can cause computational difficulties, particularly for *hyperbolic PDEs* (such as the classic linear wave equation $\partial^2 u/\partial t^2 = \partial^2 u/\partial x^2$).

BCs can be of three types:

1. If the dependent variable is specified, as in BC (1.3a), the BC is termed *Dirichlet*.
2. If the derivative of the dependent variable is specified, as in BC (1.3b), the BC is termed *Neumann*.
3. If both the dependent variable and its derivative appear in the BC, it is termed a *BC of the third type* or a *Robin BC*.

TYPES OF PDE SOLUTIONS

Equations (1.1)–(1.3) constitute a complete (*well-posed*) PDE problem and we can now consider what we mean by a solution to this problem. Briefly, the solution of a PDE problem is a *function that defines the dependent variable as a function of the independent variables* – in this case, $u(x, t)$. In other words, we seek a function that when substituted in the PDE and all of its auxiliary conditions satisfies simultaneously all of these equations.

The solution can be of two types:

1. If the solution is an actual mathematical function, it is termed an *analytical solution*. While analytical solutions are the "gold standard" for PDE solutions in the sense that they are exact, they are also generally difficult to derive mathematically for all but the simplest PDE problems (in much the same way that solutions to nonlinear algebraic equations generally cannot be derived mathematically except for certain classes of nonlinear equations).
2. If the solution is in numerical form, for example, $u(x, t)$ tabulated numerically as a function of x and t, it is termed a *numerical solution*. Ideally, the numerical solution is simply a numerical evaluation of the analytical solution. But since an analytical solution is generally unavailable for realistic PDE problems in science and engineering, the *numerical solution is an approximation to the analytical solution*, and our expectation is that it represents the analytical solution with good accuracy. However, numerical solutions can be computed with modern-day computers for very complex problems, and they will generally have good accuracy (even though this cannot be established directly by comparison with the analytical solution since the latter is usually unknown).

The focus of the MOL is *the calculation of accurate numerical solutions*.

PDE SUBSCRIPT NOTATION

Before we go on to the general classes of PDEs that the MOL can handle, we briefly discuss an alternative notation for PDEs. Instead of writing the partial derivatives

as in Eq. (1.1), we adopt a *subscript notation* that is easier to state and bears a closer resemblance to the associated computer coding. For example, we can write Eq. (1.1) as

$$u_t = Du_{xx} \tag{1.4}$$

where, for example, u_t is subscript notation for $\partial u/\partial t$. In other words, a partial derivative is represented as the dependent variable with a subscript that defines the independent variable. For a derivative that is of order n, the independent variable is repeated n times; for example, for Eq. (1.1), u_{xx} represents $\partial^2 u/\partial x^2$.

A GENERAL PDE SYSTEM

Using the subscript notation, we can now consider some general PDEs. For example, a general PDE first order in t can be considered:

$$\overline{u}_t = \overline{f}(\overline{x}, t, \overline{u}, \overline{u}_{\overline{x}}, \overline{u}_{\overline{xx}}, \ldots) \tag{1.5}$$

where an overbar (overline) denotes a vector. For example, $\overline{u}$ denotes a vector of n dependent variables

$$\overline{u} = (u_1, u_2, \ldots, u_n)^T$$

that is, a system of n simultaneous PDEs. Similarly, $\overline{f}$ denotes an n vector of derivative functions

$$\overline{f} = (f_1, f_2, \ldots, f_n)^T$$

where T denotes a *transpose* (here a row vector is transposed to a column vector). Note also that $\overline{x}$ is a vector of spatial coordinates, so that, for example, in *Cartesian coordinates* $\overline{x} = (x, y, z)^T$, while in *cylindrical coordinates* $\overline{x} = (r, \theta, z)^T$. Thus, Eq. (1.5) can represent PDEs in one, two, and three spatial dimensions.

Since Eq. (1.5) is first order in t, it requires one IC

$$\overline{u}(\overline{x}, t = 0) = \overline{u}_0(\overline{x}, \overline{u}, \overline{u}_{\overline{x}}, \overline{u}_{\overline{xx}}, \ldots) \tag{1.6}$$

where $\overline{u}_0$ is an n vector of IC functions

$$\overline{u}_0 = (u_{10}, u_{20}, \ldots, u_{n0})^T$$

The derivative vector $\overline{f}$ of Eq. (1.5) includes functions of various spatial derivatives, $(\overline{u}, \overline{u}_{\overline{x}}, \overline{u}_{\overline{xx}}, \ldots)$, and therefore we cannot state a priori the required number of BCs. For example, if the highest-order derivative in $\overline{x}$ in all of the derivative functions is second order, then we require $2 \times d \times n$ BCs, where d is the number of spatial dimensions. Thus, for Eq. (1.4), the number of required BCs is 2 (second order in x) $\times 1$ (one dimensional) $\times 1$ (one PDE) $= 2$.

We state the general BC requirement of Eq. (1.5) as

$$\overline{f}_b(\overline{x}_b, \overline{u}, \overline{u}_{\overline{x}}, \overline{u}_{\overline{xx}}, \ldots) = 0 \tag{1.7}$$

where the subscript b denotes *boundary*. The vector of BC functions, $\overline{f}_b$, has a length (number of functions) determined by the highest-order derivative in $\overline{x}$ in each PDE (in Eq. (1.5)), as discussed previously.

PDE GEOMETRIC CLASSIFICATION

Equations (1.5)–(1.7) constitute a general PDE system to which the MOL can be applied. Before proceeding to the details of how this might be done, we need to discuss the three basic forms of the PDEs as classified geometrically. This *geometric classification* can be done rigorously if certain mathematical forms of the functions in Eqs. (1.5)–(1.7) are assumed. However, we will adopt a somewhat more descriptive (less rigorous but more general) form of these functions for the specification of the three geometric classes.

If the derivative functions in Eq. (1.5) contain *only first-order derivatives in* $\bar{x}$, the PDEs are classified as *first-order hyperbolic*. As an example, the equation

$$u_t + vu_x = 0 \tag{1.8}$$

is generally called the *linear advection equation*; in physical applications, v is a linear or flow velocity. Although Eq. (1.8) is possibly the simplest PDE, this simplicity is deceptive in the sense that it can be very difficult to integrate numerically since it *propagates discontinuities*, a distinctive feature of first-order hyperbolic PDEs.

Equation (1.8) is termed a *conservation law* since it typically expresses conservation of *mass, energy, or momentum* under the conditions for which it is derived, that is, the *assumptions on which the equation is based*. Conservation laws are a bedrock of PDE mathematical models in science and engineering, and an extensive literature pertaining to their solution, both analytical and numerical, has evolved over many years.

An example of a *first-order hyperbolic system* (using the notation $u_1 \Rightarrow u$, $u_2 \Rightarrow v$) is

$$u_t = v_x \tag{1.9a}$$

$$v_t = u_x \tag{1.9b}$$

Equations (1.9a) and (1.9b) constitute a system of *two linear, constant-coefficient, first-order hyperbolic PDEs*.

Differentiation and algebraic substitution can occasionally be used to eliminate some dependent variables in systems of PDEs. For example, if Eq. (1.9a) is differentiated with respect to t and Eq. (1.9b) is differentiated with respect to x

$$u_{tt} = v_{xt}$$

$$v_{tx} = u_{xx}$$

we can then eliminate the mixed partial derivative between these two equations (assuming v_{xt} in the first equation equals v_{tx} in the second equation) to obtain

$$u_{tt} = u_{xx} \tag{1.10}$$

Equation (1.10) is the *second-order hyperbolic wave equation*.

If the derivative functions in Eq. (1.5) contain *only second-order derivatives in* $\bar{x}$, the PDEs are classified as *parabolic*. Equation (1.1) is an example of a parabolic PDE.

Finally, if a PDE *contains no derivatives in t* (e.g., the LHS of Eq. (1.5) is zero), it is classified as *elliptic*. As an example,

$$u_{xx} + u_{yy} = 0 \tag{1.11}$$

is *Laplace's equation*, where x and y are spatial independent variables in Cartesian coordinates. Note that with no derivatives in t, *elliptic PDEs require no ICs*; that is, they are entirely boundary-value PDEs.

PDEs with *mixed geometric characteristics* are possible and, in fact, are quite common in applications. For example, the PDE

$$u_t = -u_x + u_{xx} \tag{1.12}$$

is *hyperbolic–parabolic*. Since it frequently models convection (hyperbolic) through the term u_x and diffusion (parabolic) through the term u_{xx}, it is generally termed a *convection–diffusion equation*. If additionally, it includes a function of the dependent variable u such as

$$u_t = -u_x + u_{xx} + f(u) \tag{1.13}$$

then it might be termed a *convection–diffusion–reaction equation* since $f(u)$ typically models the rate of a chemical reaction. If the function is only for the independent variables, that is,

$$u_t = -u_x + u_{xx} + g(x, t) \tag{1.14}$$

the equation could be labeled an *inhomogeneous PDE*.

This discussion clearly indicates that PDE problems come in a very wide variety, depending, for example, on linearity, types of coefficients (constant, variable), coordinate system, geometric classification (hyperbolic, elliptic, parabolic), number of dependent variables (number of simultaneous PDEs), number of independent variables (number of dimensions), types of BCs, smoothness of the IC, and so on, so it might seem impossible to formulate numerical procedures with any generality that can address a relatively broad spectrum of PDEs. But in fact, the MOL provides a surprising degree of generality, although the success in applying it to a new PDE problem depends to some extent on the experience and inventiveness of the analyst; that is, MOL is not a single, straightforward, clearly defined approach to PDE problems, but rather is a general concept (or philosophy) that requires specification of details for each new PDE problem. We now proceed to illustrate the formulation of a MOL numerical algorithm, with the caveat that this will not be a general discussion of MOL as it is applied to any conceivable PDE problem.

ELEMENTS OF THE MOL

The basic idea of the MOL is to *replace the spatial (boundary-value) derivatives in the PDE with algebraic approximations*. Once this is done, the spatial derivatives are no longer stated explicitly in terms of the spatial independent variables. Thus, in effect, *only the initial-value variable, typically time in a physical problem, remains*. In other words, with only one remaining independent variable, we have a *system of ODEs that approximate the original PDE*. The challenge, then, is to formulate the approximating system of ODEs. Once this is done, we can apply any integration

algorithm for initial-value ODEs to compute an approximate numerical solution to the PDE. Thus, one of the salient features of the MOL is the use of *existing, and generally well-established, numerical methods for ODEs*.

To illustrate this procedure, we consider the MOL solution of Eq. (1.8). First we need to replace the spatial derivative u_x with an algebraic approximation. In this case we will use a *finite difference* (FD), such as

$$u_x \approx \frac{u_i - u_{i-1}}{\Delta x} \tag{1.15}$$

where i is an *index designating a position along a grid in x and* Δx *is the spacing in x along the grid*. Thus, for the left-end value of x, $i = 1$, and for the right-end value of x, $i = M$; that is, the grid in x has M points. Then the MOL approximation of Eq. (1.8) is

$$\frac{du_i}{dt} = -v\frac{u_i - u_{i-1}}{\Delta x}, \quad 1 \le i \le M \tag{1.16}$$

Note that Eq. (1.16) is written as an ODE since there is now *only one independent variable, t*. Note also that Eq. (1.16) represents a system of M ODEs.

This transformation of a PDE, Eq. (1.8), to a system of ODEs, Eq. (1.16), illustrates the essence of the MOL, namely, *the replacement of the spatial derivatives, in this case* u_x*, so that a system of ODEs approximates the original PDE*. Then, *to compute the solution of the PDE, we compute a solution to the approximating system of ODEs*. But before considering this integration in t, we have to complete the specification of the PDE problem. Since Eq. (1.8) is first order in t and first order in x, it requires one IC and one BC. These will be taken as

$$u(x, t = 0) = f(x) \tag{1.17a}$$

$$u(x = 0, t) = g(t) \tag{1.17b}$$

Since Eq. (1.16) constitutes M first-order, initial-value ODEs, M initial conditions are required, and from Eq. (1.17a), these are

$$u(x_i, t = 0) = f(x_i), \quad 1 \le i \le M \tag{1.18a}$$

Also, application of BC (1.17b) gives for grid point $i = 1$

$$u(x_1, t) = g(t) \tag{1.18b}$$

Equations (1.16) and (1.18) now constitute the complete MOL approximation of Eq. (1.8) subject to Eqs. (1.17a) and (1.17b). The solution of this ODE system gives the M functions

$$u_1(t), u_2(t), \ldots, u_{M-1}(t), u_M(t) \tag{1.19}$$

that is, an approximation to $u(x, t)$ at the grid points $i = 1, 2, \ldots, M$.

Before we go on to consider the numerical integration of the approximating ODEs, in this case Eq. (1.16), we briefly consider further the FD approximation of Eq. (1.15), which can be written as

$$u_x \approx \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \tag{1.20}$$

where $O(\Delta x)$ denotes *of order* Δx; that is, the truncation error of the approximation of Eq. (1.16) is *proportional to* Δx *to the first power (varies linearly with* Δx*)*; thus, Eq. (1.20) is also termed a *first-order* FD (since Δx is to the first power in the order or truncation error term). The term *truncation error* reflects the fact that the FD of Eq. (1.15) comes from a *truncated Taylor series*.

Note that the numerator of Eq. (1.15), $u_i - u_{i-1}$, is a difference in two values of u. Also, the denominator Δx remains finite (nonzero). Hence the name *finite difference* (and it is an approximation because of the truncated Taylor series, so a more complete description is *first-order FD approximation*). In fact, in the limit $\Delta x \to 0$, the approximation of Eq. (1.15) becomes *exactly the derivative*. However, in a practical computer-based calculation, Δx remains finite, so Eq. (1.15) remains an approximation.

Also, Eq. (1.8) typically describes the flow of a physical quantity such as concentration of a chemical species or temperature, represented by u, from left to right with respect to x with velocity v. Then, using the FD approximation of Eq. (1.20) at i involves u_i and u_{i-1}. In a flowing system, u_{i-1} is to the left (in x) of u_i or is *upstream* or *upwind* of u_i (to use a nautical analogy). Thus, Eq. (1.20) is termed a *first-order upwind FD approximation*. Generally, for strongly convective systems such as that modeled by Eq. (1.8), *some form of upwinding is required in the numerical solution of the descriptive PDEs*; we will look at this requirement further in the subsequent discussion.

ODE INTEGRATION WITHIN THE MOL

We now consider briefly the numerical integration of the M ODEs of Eq. (1.16). If the derivative du_i/dt is approximated by a first-order FD

$$\frac{du_i}{dt} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t) \tag{1.21}$$

where n is an index for the variable t (t moves forward in steps denoted or indexed by n), then an FD approximation of Eq. (1.16) is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -v\frac{u_i^n - u_{i-1}^n}{\Delta x}$$

or solving for u_i^{n+1},

$$u_i^{n+1} = u_i^n - (v\Delta t/\Delta x)(u_i^n - u_{i-1}^n), \quad i = 1, 2, \ldots, M \tag{1.22}$$

Equation (1.22) has the important characteristic that it gives u_i^{n+1} *explicitly*; that is, we can solve for the solution at the advanced point in t, $n+1$, from the solution at the base point n. In other words, explicit numerical integration of Eq. (1.16) is by the *forward FD* of Eq. (1.21), and this procedure is generally termed the *forward Euler method*, which is the most basic form of ODE integration.

While the explicit form of Eq. (1.22) is computationally convenient, it has a possible limitation. If the time step Δt is *above a critical value, the calculation becomes unstable*, which is manifest by successive changes in the dependent variable, $\Delta u = u_i^{n+1} - u_i^n$, becoming larger and eventually unbounded as the calculation moves forward in t (for increasing n). In fact, for the solution of Eq. (1.8) by the

method of Eq. (1.22) to remain stable, the dimensionless group $(v\Delta t/\Delta x)$, which is called the *Courant-Friedricks-Lewy or CFL number*, must remain below a critical value – in this case, unity. Note that this *stability limit* places an upper limit on Δt for a given v and Δx; if one attempts to increase the accuracy of Eq. (1.22) by using a smaller Δx (larger number of grid points in x by increasing M), a smaller value of Δt is required to keep the CFL number below its critical value. Thus, there is a *conflicting requirement of improving accuracy while maintaining stability*.

The stability limit of the explicit Euler method as implemented via the forward FD of Eq. (1.21) can be circumvented by using a *backward FD* for the derivative in t

$$\frac{du_i}{dt} \approx \frac{u_i^n - u_i^{n-1}}{\Delta t} + O(\Delta t) \tag{1.23}$$

so that the FD approximation of Eq. (1.16) becomes

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} = -v\frac{u_i^n - u_{i-1}^n}{\Delta x}$$

or after rearrangement (with $(v\Delta t/\Delta x) = \alpha$),

$$(1+\alpha)u_i^n - \alpha u_{i-1}^n = u_i^{n-1}, \quad i = 1, 2, \ldots, M \tag{1.24}$$

Note that we cannot now solve Eq. (1.24) explicitly for the solution at the advanced point u_i^n in terms of the solution at the base point u_i^{n-1}. Rather, Eq. (1.24) is *implicit* in u_i^n because u_{i-1}^n is also unknown; that is, we must solve Eq. (1.24) written for each grid point $i = 1, 2, \ldots, M$ as a simultaneous system of *bidiagonal equations* (bidiagonal because each of Eq. (1.24) has two unknowns so that *simultaneous solution of the full set of approximating algebraic equations is required* to obtain the complete numerical solution $u_1^n, u_2^n, \ldots, u_M^n$). Thus, the solution of Eq. (1.24) is termed the *implicit Euler method*.

We could then naturally ask why use Eq. (1.24) when Eq. (1.22) is so much easier to use (explicit calculation of the solution at the next step in t of Eq. (1.22) vs. the implicit calculation of Eq. (1.24)). The answer is that the implicit calculation of Eq. (1.24) is often worthwhile because the *implicit Euler method has no stability limit* (is *unconditionally stable* in comparison with the explicit method, with the stability limit stated in terms of the CFL condition). However, there is a price to pay for the improved stability of the implicit Euler method; that is, we must solve a *system of simultaneous algebraic equations*; Eq. (1.24) is an example. Furthermore, if the original ODE system approximating the PDE is nonlinear, we have to solve a *system of nonlinear algebraic equations*. (Equation (1.24) is linear, so the solution is much easier.) The system of nonlinear equations is typically solved by a *variant of Newton's method* that can become very demanding computationally if the number of ODEs is large (due to the use of a large number of spatial grid points in the MOL approximation of the PDE, especially when we attempt the solution of 2D and 3D PDEs). If you have had some experience with Newton's method, you may appreciate that the *Jacobian matrix* of the nonlinear algebraic system can *become very large and sparse as the number of spatial grid points increases*.

Additionally, although there is no limit for Δt with regard to stability for the implicit method, there is a *limit with regard to accuracy*. In fact, the first-order upwind

approximation of u_x in Eq. (1.8), Eq. (1.20), and the first-order approximation of u_t in Eq. (1.8), Eq. (1.21) or (1.23), taken together limit the accuracy of the resulting FD approximation of Eq. (1.8). One way around this accuracy limitation is to use *higher-order FD approximations for the derivatives in Eq. (1.8)*.

For example, if we consider the second-order approximation of u_x at i

$$u_x \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{1.25}$$

substitution in Eq. (1.8) gives the MOL approximation of Eq. (1.8)

$$\frac{du_i}{dt} = -v\frac{u_{i+1} - u_{i-1}}{2\Delta x}, \quad 1 \le i \le M \tag{1.26}$$

We could then reason that if the integration in t is performed by the explicit Euler method, that is, we use the approximation of Eq. (1.21) for $u_t = du_i/dt$, the resulting numerical solution should be more accurate than the solution from Eq. (1.22). In fact, the MOL approximation based on this idea

$$u_i^{n+1} = u_i^n - \frac{v\Delta t}{2\Delta x}(u_{i+1}^n - u_{i-1}^n), \quad i = 1, 2, \ldots, M \tag{1.27}$$

is *unconditionally unstable*; this conclusion can be demonstrated by a *von Neumann stability analysis* that we will not cover here. This surprising result demonstrates that *replacing the derivatives in PDEs with higher-order approximations does not necessarily guarantee more accurate solutions, or even stable solutions*.

NUMERICAL DIFFUSION AND OSCILLATION

Even if the implicit Euler method is used for the integration in t of Eq. (1.26) to achieve stability (or a more sophisticated explicit integrator in t is used that automatically adjusts Δt to achieve a prescribed accuracy), we would find that the solution *oscillates* unrealistically. This numerical distortion is one of two generally observed forms of numerical error. The other numerical distortion is *diffusion* that would be manifest in the solution from Eq. (1.22). Briefly, the solution would exhibit excessive smoothing or rounding at points in x where the solution changes rapidly. This overall observation that a *first-order approximation of u_x produces numerical diffusion, while higher-order approximations of u_x produce numerical oscillation* is predicted by the *Godunov order barrier theorem for the Riemann problem* [2]. To explain briefly, the order barrier is first order and *any linear FD approximation above first order will be oscillatory*. Equation (1.8) is an example of the Riemann problem [2] if IC Eq. (1.17a) is discontinuous; for example, $u(x, t = 0) = h(t)$, where $h(t)$ is the *Heaviside unit step function*. The (exact) analytical solution is the IC function $f(x)$ of Eq. (1.17a) moving left to right with velocity v (from Eq. (1.8)) and without distortion, that is, $u(x, t) = f(x - vt)$; however, the numerical solution will oscillate if u_x in Eq. (1.8) is replaced with a linear approximation of second or higher order.

We should also mention one point of terminology for FD approximations. The RHS of Eq. (1.25) is an example of a *centered approximation* since the two points at $i + 1$ and $i - 1$ are centered around the point i. Equation (1.20) is an example of a *noncentered, one-sided*, or *upwind approximation* since the points i and $i - 1$ are not centered with respect to i.

Finally, to conclude the discussion of first-order hyperbolic PDEs such as Eq. (1.8), since the Godunov theorem indicates that FD approximations above first order will produce numerical oscillations in the solution, the question remains if there are approximations above first order that are nonoscillatory. To answer this question we note first that *the Godunov theorem applies to linear approximations*; for example, Eq. (1.25) is a linear approximation since u on the RHS is to the first power. If, however, we consider nonlinear approximations of u_x, we can in fact develop approximations that are nonoscillatory. The details of such nonlinear approximations are beyond the scope of this discussion, so we will merely mention that they are termed *high-resolution* methods that seek a *total variation diminishing solution*. Such methods, which include *flux limiter* and *weighted essentially nonoscillatory* methods, seek to avoid nonreal oscillations when shocks or discontinuities occur in the solution (such as in the Riemann problem) [3].

So far we have considered only the MOL solution of first-order PDEs, for example, Eq. (1.8). We conclude this discussion of the MOL by considering a second-order PDE, the parabolic Eq. (1.1). To begin, we need an approximation for the second derivative u_{xx}. A commonly used *second-order, central approximation* is (again, derived from the Taylor series, so the term $O(\Delta x^2)$ represents the truncation error)

$$u_{xx} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x^2) \tag{1.28}$$

Substitution of Eq. (1.28) in Eq. (1.8) gives a system of approximating ODEs

$$\frac{du_i}{dt} = D\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}, \quad i = 1, 2, \ldots, M \tag{1.29}$$

Equation (1.29) is then integrated subject to IC (1.2) and BCs (1.3a) and (1.3b). This integration in t can be by the explicit Euler method, the implicit Euler method, or any other higher-order integrator for initial-value ODEs. Generally stability is not as much of a concern as with the previous hyperbolic PDEs (a characteristic of parabolic PDEs that tend to smooth solutions rather than hyperbolic PDEs that tend to propagate nonsmooth conditions). However, stability constraints do exist for explicit methods. For example, for the explicit Euler method with a step Δt in t, the stability constraint is $D\Delta t/\Delta x^2 <$ constant (so that as Δx is reduced to achieve better spatial accuracy in x, Δt must also be reduced to maintain stability).

Before proceeding with the integration of Eq. (1.29), we must include BCs (1.3a) and (1.3b). The Dirichlet BC at $x = x_0$, Eq. (1.3a), is merely

$$u_1 = u_b \tag{1.30}$$

and therefore the ODE of Eq. (1.29) for $i = 1$ is not required and the ODE for $i = 2$ becomes

$$\frac{du_2}{dt} = D\frac{u_3 - 2u_2 + u_b}{\Delta x^2} \tag{1.31}$$

DIFFERENTIAL ALGEBRAIC EQUATIONS

Equation (1.30) is *algebraic*, and therefore in combination with the ODEs of Eq. (1.29), we have a *differential algebraic (DAE) system*.

At $i = M$, we have Eq. (1.29)

$$\frac{du_M}{dt} = D\frac{u_{M+1} - 2u_M + u_{M-1}}{\Delta x^2} \tag{1.32}$$

Note that u_{M+1} is outside the grid in x; that is, $M + 1$ is a *fictitious point*. But we must assign a value to u_{M+1} if Eq. (1.32) is to be integrated. Since this requirement occurs at the boundary point $i = M$, we obtain this value by approximating BC (1.3b) using the centered FD approximation of Eq. (1.25)

$$u_x \approx \frac{u_{M+1} - u_{M-1}}{2\Delta x} = 0$$

or

$$u_{M+1} = u_{M-1} \tag{1.33}$$

We can add Eq. (1.33) as an *algebraic equation to our system of equations*, that is, continue to use the DAE format, or we can substitute u_{M+1} from Eq. (1.33) into the Eq. (1.32)

$$\frac{du_M}{dt} = D\frac{u_{M-1} - 2u_M + u_{M-1}}{\Delta x^2} \tag{1.34}$$

and arrive at an ODE system (Eq. (1.29) for $i = 3, \ldots, M - 1$, Eq. (1.31) for $i = 2$, and Eq. (1.34) for $i = M$). Both approaches, either an ODE system or a DAE system, have been used in MOL studies. Either way, we now have a complete formulation of the MOL ODE or DAE system, including the BCs at $i = 1, M$ in Eq. (1.29). The integration of these equations then gives the numerical solution $u_1(t), u_2(t), \ldots, u_M(t)$. The preceding discussion is based on a relatively basic DAE system, but it indicates that integrators designed for DAE systems can play an important role in MOL analysis.

If the implicit Euler method is applied to Eq. (1.29), we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = D\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2}, \quad i = 1, 2, \ldots, M$$

or (with $\alpha = D\Delta t/\Delta x^2$),

$$u_{i+1}^{n+1} - (1/\alpha + 2)u_i^{n+1} + u_{i-1}^{n+1} = (1/\alpha)u_i^n, \quad i = 1, 2, \ldots, M$$

which is a *tridiagonal* system of algebraic equations (three unknowns in each equation). Since such *banded* systems (the nonzero elements are banded around the main diagonal) are common in the numerical solution of PDE systems, special algorithms have been developed to take advantage of the banded structure, typically by not storing and using the zero elements outside the band. These special algorithms that take advantage of the *structure of the problem equations* can result in major savings in computation time. In the case of tridiagonal equations, the special algorithm is generally called *Thomas' method*. If the coefficient matrix of the algebraic system does not have a well-defined structure, such as bidiagonal or tridiagonal, but consists of mostly zeros with a relatively small number of nonzero elements, which is often the case in the numerical solution of PDEs, the coefficient matrix is said to be *sparse*; special algorithms and associated software for sparse systems have been

developed that can result in very substantial savings in the storage and numerical manipulation of sparse matrices.

Generally when applying the MOL, the integration of the approximating ODE/DAEs (e.g., Eqs. (1.21) and (1.29)) is accomplished by using *library routines for initial-value* ODE/DAEs. In other words, the explicit programming of the ODE/DAE integration (such as the explicit or implicit Euler method) is avoided; rather, an established integrator is used. This has the advantage that (1) the detailed programming of the integration can be avoided, particularly the linear algebra (solution of simultaneous equations) required by an implicit integrator, so that the MOL analysis is substantially simplified, and (2) library routines (usually written by experts) include features that make these routines especially effective (robust) and efficient such as automatic integration step size adjustment and the use of higher-order integration methods (beyond the first-order accuracy of the Euler methods); also, generally, they have been thoroughly tested. Thus, the use of quality ODE/DAE library routines is usually an essential part of MOL analysis. We therefore list at the end of this chapter some public domain sources of such library routines.

HIGHER DIMENSIONS AND DIFFERENT COORDINATE SYSTEMS

To conclude this discussion of the MOL solution of PDEs, we cover two additional points. First, we have considered PDEs in only Cartesian coordinates, and in fact, just one Cartesian coordinate, x. But MOL analysis can in principle be carried out in any coordinate system. Thus, Eq. (1.1) can be generalized to

$$\frac{\partial u}{\partial t} = D\nabla^2 u \tag{1.35}$$

where ∇^2 is the *coordinate independent Laplacian operator* that can then be expressed in terms of a particular coordinate system. For example, in cylindrical coordinates, Eq. (1.35) is

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{1.36}$$

and in spherical coordinates, Eq. (1.35) is

$$\frac{\partial u}{\partial t} = D\left[\frac{\partial^2 u}{\partial r^2} + \frac{2}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\left(\frac{\partial^2 u}{\partial \theta^2} + \frac{\cos\theta}{\sin\theta}\frac{\partial u}{\partial \theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 u}{\partial \phi^2}\right] \tag{1.37}$$

The challenge then in applying the MOL to PDEs such as Eqs. (1.36) and (1.37) is the algebraic approximation of the RHS ($\nabla^2 u$) using, for example, FDs, *finite elements* or *finite volumes*; all of these approximations have been used in MOL analysis, as well as *Galerkin*, *least squares*, *spectral*, and other methods. A particularly demanding step is *regularization of singularities* such as at $r = 0$ (note the number of divisions by r in the RHS of Eqs. (1.36) and (1.37)) and at $\theta = 0, \pi/2$ (note the divisions by $\sin(\theta)$ in Eq. (1.37)). Thus the application of the MOL typically requires analysis based on the experience and creativity of the analyst (i.e., it is generally not a mechanical procedure from beginning to end).

The complexity of the numerical solution of higher-dimensional PDEs in various coordinate systems prompts the question of why a particular coordinate system

would be selected over others. The mathematical answer is that the judicious choice of a coordinate system *facilitates the implementation of the BCs in the numerical solution*. The answer based on physical considerations is that the *coordinate system is selected to reflect the geometry of the problem system*. For example, if the physical system has the shape of a cylinder, cylindrical coordinates would be used. This choice then facilitates the implementation of the BC at the exterior surface of the physical system (exterior surface of the cylinder). However, this can also lead to complications such as the $r = 0$ singularities in Eq. (1.36) (due to the variable $1/r$ and $1/r^2$ coefficients). The resolution of these complications is generally worth the effort rather than the use of a coordinate system that does not naturally conform to the geometry of the physical system. If the physical system is not shaped in accordance with a particular coordinate system, that is, has an *irregular geometry*, then an approximation to the physical geometry is used, generally termed *body-fitted coordinates*.

h- AND *p*-REFINEMENT

Increasing or decreasing the grid spacing over parts or all of the problem domain is termed *h-refinement*. The name comes from the common convention in the numerical analysis literature of using h as the symbol for the grid spacing.

Modifying the order of the derivative approximation is termed *p-refinement*. The name comes from the convention of using the symbol p for the order of the approximation (e.g., $O(\Delta x)^2$ for a second-order approximation with $p = 2$).

h-refinement seeks to refine the grid spacing using *local truncation error* estimates or other refinement parameters in order to improve the accuracy of the solution. Similarly, *p*-refinement seeks to refine the order of derivative approximations (using the same refinement parameters). Generally there is no unique, general, best combination of *h*- and *p*-refinement and the solution of large problems usually requires some trial and error for a trade-off between accuracy and computational effort; *the goal is to reach a prespecified bound on the global error with a minimal amount of work* [4]. We will not discuss this aspect further here, but refer to [4–6] for further discussion.

ORIGIN OF THE NAME "METHOD OF LINES"

As a concluding point, we might consider the origin of the name *method of lines*. If we consider Eq. (1.29), integration of this ODE system produces the solution $u_2(t), u_3(t), \ldots, u_M(t)$. (*Note:* $u_1(t) = u_b$, a constant, from BC (1.30).) We could then plot these functions in an x–$u(x, t)$ plane as a vertical line at each x ($i = 2, 3, \ldots, M$), with the height of each line equal to $u(x_i, t)$. In other words, the plot of the solution would be a set of vertical parallel lines suggesting the name *method of lines* [7].

To illustrate this interpretation, consider the diffusion equation problem of Eq. (1.1) with

- Diffusion coefficient $D = 1$
- Dirichlet BC at the left end, $u(x = -5, t) = 0$

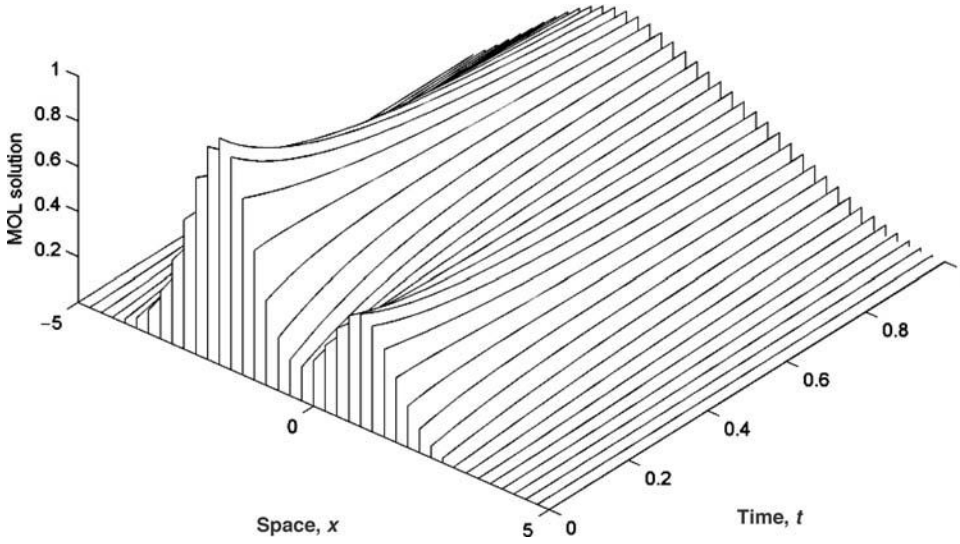

Figure 1.1. MOL solution of Eq. (1.1) illustrating the origin of the *method of lines*

- Neumann BC at the right end, $u_x(x = 5, t) = 0$ (spatial domain $-5 \leq x \leq 5$)
- Time domain $0 \leq t \leq 1$
- Initial condition $u(x, t = 0) = (1/2)e^{-(x-1)^2} + e^{-(x+2)^2}$

The MOL solution for the problem is shown in Figure (1.1). This numerical solution was obtained using Matlab and the MOL library routine `dss044` [7] with the number of grid points $M = 41$ (so that the grid spacing is $[5 - (-5)]/(41 - 1) = 0.25$).

The result of Figure 1.1 matches very well the *infinite-domain analytical solution*

$$u(x, t) = \frac{1}{2\sqrt{4Dt + 1}} \left(e^{\frac{3(2x+1)}{4Dt+1}} + 2\right) e^{-\frac{(x+2)^2}{4Dt+1}} \tag{1.38}$$

This agreement is illustrated in Figure 1.2 where the analytical result has been superimposed on the MOL solution. This comparison illustrates an important distinction between the analytical and numerical (MOL) solutions. The analytical solution is for an infinite domain, $-\infty \leq x \leq \infty$, while the MOL solution is computed on a finite domain (as required by a computer), $-5 \leq x \leq 5$ [1]. The agreement between the analytical and numerical solutions reflects the property that both solutions remain at essentially zero for $u(x = -5, t)$ and $u(x = 5, t)$ for $t \leq 1$ as indicated in Figure 1.2.[2]

[2] The exact analytical solution for the finite-domain problem is considerably more complicated than Eq. (1.38) but could be derived by a finite Fourier sine transform ([8], pp. 405–415) or a Green's function ([9], pp. 48, 58).

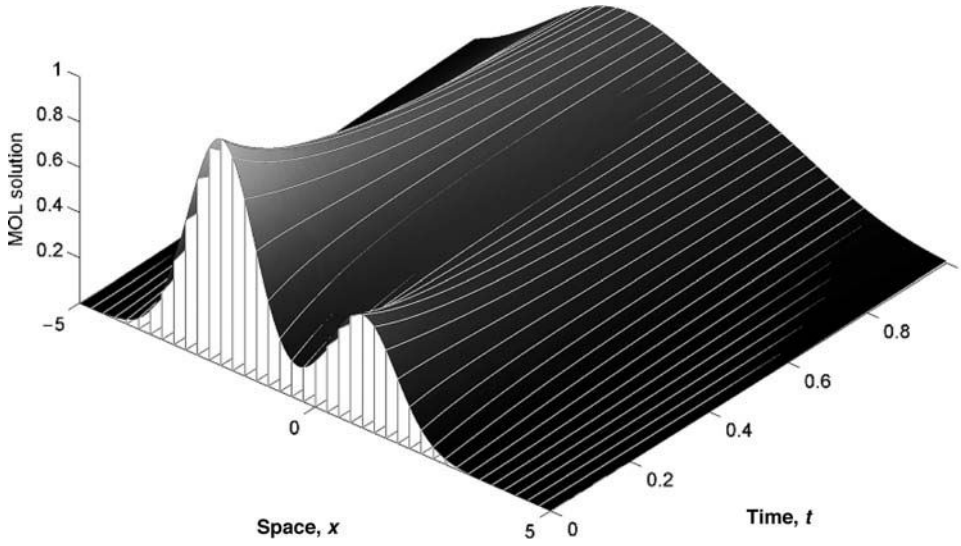

Figure 1.2. Superposition of the MOL solution of Eq. (1.1) and the analytical solution of Eq. (1.38)

SOURCES OF ODE/DAE INTEGRATORS

One of the very useful aspects of MOL is that it enables tried-and-tested ODE/DAE numerical routines to be used, many of which are in the public domain. The following sources are a good starting point for these routines. For example, the LSODE and VODE series of ODE/DAE integrators [2s], DASSL for DAEs [2s], and the SUNDIALS library [5s] are widely used in MOL analysis; test problems for ODE/DAE routines are also available [6s]. Additionally, routines that can be called from MOL codes are available to perform a variety of complementary computations (e.g., functional approximation by interpolation, evaluation of integrals, maximization and minimization in optimization associated with the solution of PDEs) [1s, 3s, 4s].

[1s] http://www.netlib.org/
[2s] http://www.netlib.org/ode/index.html (emphasis on ODE/DAE software that can be used in MOL analysis)
[3s] http://gams.nist.gov/
[4s] http://www.acm.org/toms/
[5s] http://www.llnl.gov/CASC/software.html
[6s] http://www.dm.uniba.it/~testset/

REFERENCES

[1] Hamdi, S., W. E. Schiesser, and G. W. Griffiths (2007), Method of Lines, *Scholarpedia*, **2**(7):2859; available online at http://www.scholarpedia.org/article/method_of_lines
[2] Wesseling, P. (2001), *Principles of Computational Fluid Dynamics*, Springer, Berlin
[3] Shu, C.-W. (1998), Essentially Non-Oscillatory and Weighted Essential Non-Oscillatory Schemes for Hyperbolic Conservation Laws, In: B. Cockburn, C. Johnson, C.-W. Shu,

and E. Tadmor (Eds.), *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, Lecture Notes in Mathematics, vol. 1697, Springer, Berlin, pp. 325–432

[4] de Sterck, H., T. A. Manteuffel, S. F. McCormick, J. Nolting, J. Ruge, and L. Tang (2008), Efficiency-Based *h*- and *hp*-Refinement Strategies for Finite Element Methods, *Num. Linear Algebr. Appl.*, **15**: 89–114.

[5] Aftosmis, M. J. and M. J. Berger (2002), Multilevel Error Estimation and Adaptive *h*-Refinement for Cartesian Meshes with Embedded Boundaries, In: *AIAA Paper 2002-0863, 40th AIAA Aerospace Sciences Meeting and Exhibit*, January 14–17, 2002, Reno, NV

[6] Dong, S. and G. E. Karniadakis (May 9, 2003), p-Refinement and p-Threads, *Comput. Methods Appl. Mech. Eng.*, **192**(19): 2191–2201

[7] Schiesser, W. E. (1991), *Numerical Method of Lines Integration of Partial Differential Equations*, Academic Press, San Diego, CA

[8] Schiesser, W. E. (1994), *Computational Mathematics in Engineering and Applied Science: ODEs, DAEs, and PDEs*, CRC Press, Boca Raton, FL

[9] Polyanin, A. (2002), *Handbook of Linear Partial Differential Equations for Engineers and Scientists*, Chapman & Hall/CRC, Boca Raton, FL