Chapter 1
Introduction to Partial Differential Equation Analysis Chemotaxis

(1.1) Introduction

This chapter serves as an introduction to the analysis of biomedical science and engineering (BMSE) systems based on partial differential equations (PDEs) programmed in R. The general format of this chapter and the chapters that follow consists of the following steps:

- Presentation of a PDE model as a system of partial differential equations, possibly with the inclusion of some additional ordinary differential equations (ODEs).
- Review of algorithms for the numerical solution of the PDE model.
- Discussion of a set of R routines that implement the numerical algorithms as applied to the model.
- Review of the computed output.
- Conclusions concerning the model, computer implementation, output and possible extensions of the analysis.

This format is introductory and application oriented with a minimum of mathematical formality. The intention is to help the reader start with PDE analysis of BMSE systems without becoming deeply involved in the details of PDE numerical methods and their computer implementation (e.g., coding). Also, the presentation is self contained so that the reader will not have to go to other sources such as a software download to find the routines that are discussed and used in a particular application. Our final objective then is for the detailed discussions of the various applications to facilitate a start in the PDE analysis of BMSE systems.

In this chapter we consider the following topics.

- A brief introduction to PDEs.
- Application of PDE analysis to chemotaxis.
- Algorithms for the numerical solution of a simultaneous 2-PDE, nonlinear chemotaxis model.
- Computer routines for implementation of the numerical algorithms.
- Traveling wave features of the 2-PDE chemotaxis model numerical solution.

(1.2) Linear diffusion model

Inanimate systems have the general feature that chemical species move from regions of high concentration to regions of low concentration by mechanisms that are often modeled as diffusion, that is, according to Fick's first and second laws. In 1D, this diffusion is described (according to Fick's second law) as

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left[D\frac{\partial c}{\partial x}\right] \tag{1.1a}$$

where

c volume concentration

x spatial coordinate

t time

D diffusivity

In accordance with the usual convention for PDE notation, the dependent variable will subsequently be denoted as u rather than c. Thus, eq. (1.1a) will be

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left[D\frac{\partial u}{\partial x}\right] \tag{1.1a}$$

We can note the following features of eq. (1.1a).

- Eq. (1.1a) is a PDE since it has two independent variables, x and t. A differential equation with only one independent variable is termed an ordinary differential equation (ODE). Note also that ∂ is used to denote a partial derivative.
- The solution of eq. (1.1a) is the dependent variable u as a function of the independent variables x and t, that is, $u(x,t)$ in numerical form (rather than analytical form).
- Eq. (1.1a) is linear since the dependent variable u and its partial derivatives are to the first degree (not to be confused with order since eq. (1.1a) is first order in t because of the first order derivative in t and second order in x because of the second order derivative in x). Classifying eq. (1.1a) as linear presupposes the diffusivity D is not a function of u. Eq. (1.1a) is nonlinear if $D = D(u)$ because of the product $D(u)\partial u/\partial x$.
- The diffusivity D is inside the first (left most) differentiation to handle the case when D is a function of x and/or u. If D is a constant, it can be moved outside the first differentiation.

Eq. (1.1a) models ordinary diffusion due to, for example, random motion of molecules. A distinguishing feature of this type of diffusion is net movement in the direction of

decreasing concentration as reflected in Fick's first law (see the derivation in Appendix 1)

$$\mathbf{q}_x = -D\frac{\partial u}{\partial x} \tag{1.1b}$$

$\mathbf{q}_x$ is a component of the diffusion flux vector (with additional components $\mathbf{q}_y, \mathbf{q}_z$ in Cartesian coordinates) and is therefore denoted with boldface. The minus sign signifies diffusion in the direction of decreasing concentration (since for $\partial u/\partial x < 0,\ \mathbf{q}_x > 0$).

Since eq. (1.1a) is first order in t and second order in x, it requires one initial condition (IC) and two boundary conditions (BCs). For example,

$$u(x, t = 0) = f(x),\ 0 \le x \le x_L \tag{1.1c}$$

$$u(x = 0, t) = g_1(t),\ u(x = x_L, t) = g_2(t),\ t > 0 \tag{1.1d,e}$$

where x_L is a constant (length) to be specified and f_1, g_1, g_2 are functions to be specified. Since BCs (1.1d,e) specify the dependent variable $u(x, t)$ at two particular (boundary) values of x, that is $x = 0, x_L$, they are termed Dirichlet BCs. The derivatives in x can be specified as BCs, e.g.,

$$\frac{\partial u(x = 0, t)}{\partial x} = g_3(t),\ \frac{\partial u(x = x_L, t)}{\partial x} = g_4(t) \tag{1.1f,g}$$

Eqs. (1.1f,g) are termed Neumann BCs. Also, the dependent variable and its derivative can be specified at a boundary, e.g.,

$$-\frac{u(x = 0, t)}{\partial x} + u(x = 0, t) = g_5(t),\ \frac{u(x = x_L, t)}{\partial x} + u(x = x_L, t) = g_6(t) \tag{1.1h,i}$$

Eqs. (1.1h,i) are termed third-type, Robin or natural BCs. All of these various forms of BCs (eq. (1.1d) to eq. (1.1i)) are useful in applications.

Finally, note that t is defined over an open-ended interval or domain, $t > 0$, and is termed an initial value variable (typically time in an application). x is defined between two different (boundary) values in x, denoted here as $0, x_L$ or more generally x_0, x_L (typically physical boundaries in an application). However the interval in x can be seminfinite, e.g., $-\infty < x \le 0,\ 0 \le x \le \infty$ or fully infinite, $-\infty < x < \infty$.

(1.3) Nonlinear chemotaxis model

Eqs. (1.1a) and (1.1b) can be extended to a nonlinear form of Fick's first and second laws. Also, we can consider more than one dependent variable, and we now consider two dependent variables, $u1(x, t)$ and $u2(x, t)$ in place of just $u(x, t)$. The 2-PDE model for chemotaxis is ([2], p68)

$$u_{1t} = -ku_2 \tag{1.2a}$$

$$u_{2t} = D\frac{\partial}{\partial x}\left[u_{2x} - 2\frac{u_2}{u_1}u_{1x}\right] \tag{1.2b}$$

Here we have employed subscript notation for some of the partial derivatives. For example,

$$\frac{\partial u_1}{\partial t} \Rightarrow u_{1t},\ \frac{\partial u_1}{\partial x} \Rightarrow u_{1x},\ \frac{\partial u_2}{\partial t} \Rightarrow u_{2t}$$

Note that the PDE dependent variables can have two subscripts. The first is a number denoting a particular dependent variable. The second is a letter denoting a partial derivative with respect to a particular independent variable. For example, u_{1t} denotes the first dependent variable u_1 differentiated with respect to t. Also, the second (letter) subscript can be repeated to denote a higher order derivative. For example, subscript notation can be used in eq. (1.2b),

$$\frac{\partial}{\partial x}\left[u_{2x}\right] = u_{2xx}$$

$$\frac{\partial}{\partial x}\left[-2\frac{u_2}{u_1}u_{1x}\right] = \left[-2\frac{u_2}{u_1}u_{1x}\right]_x$$

This compact subscript notation for partial derivatives can be useful in conveying a correspondence between the mathematics and the associated computer coding. This will be illustrated in the subsequent programming of eqs. (1.2).

The variables and parameters in eqs. (1.2) are

u_1	attractant concentration
u_2	bacteria concentration
x	spatial coordinate
t	time
k	rate constant
D	diffusivity

Eqs. (1.2) define the volume concentration of a microorganism (such as bacteria), $u_2(x,t)$, when responding to an attractant (such as a nutrient or food supply), $u_1(x,t)$. Eq. (1.2a) reflects the rate of consumption of u_1, that is u_{1t}, due to u_2; the rate constant is taken as $k > 0$ so that the minus is required for consumption ($u_{1t} \le 0$ with $u_2 \ge 0$).

Eq. (1.2b) is an extension of eq. (1.1b), and implies a diffusion flux $\mathbf{q}$ in the x direction.

$$\mathbf{q} = -D\left[u_{2x} - 2\frac{u_2}{u_1}u_{1x}\right] \tag{1.2c}$$

(the subscript x in $\mathbf{q}_x$ has been dropped). Eq. (1.2c) can be considered an extension of eq. (1.1b). It is nonlinear because of the term $(u_2/u_1)u_{1x}$. We can note the following details about eq. (1.2c).

- The first RHS term, $-Du_{2x}$ is just Fick's first law, eq. (1.1b). In other words, the flux of eq. (1.2c), $\mathbf{q}$, is composed partly of the usual flux in the direction of decreasing gradient ($Du_{2x} < 0$ which tends to make $\mathbf{q} > 0$ because of the minus sign in eq. (1.2c)).

- The second RHS term is opposite in sign to the first and therefore gives the opposite effect for the flux $\mathbf{q}$. Note that the gradient in this term is u_{1x}, not u_{2x}. Thus, this term causes the bacteria flux $\mathbf{q}$ to increase with increasing attractant concentration u_1.

  Also, the ratio u_2/u_1 is a factor in determining the flux $\mathbf{q}$. This ratio causes the rate of transfer (flux) of the bacteria, $\mathbf{q}$, to increase with increasing bacteria concentration, u_2, and also to increase with a decrease in the attractant concentration, u_1; for the latter, the bacteria apparently move faster when facing decreasing availability of the attractant (e.g., nutrient or food).

  Clearly, this nonlinear RHS term is a significant departure from the diffusion term of Fick's first law, eq. (1.1b). This is a unique feature of chemotaxis in which the bacteria seek higher concentrations of the attractant; this seems plausible if, for example, the attractant is a nutrient such as food. This effect is clearly a feature of an animate (living) system, such as bacteria, rather than an inanimate system.

- Since the second RHS term has a rather unconventional form, we would expect that it will introduce unusual features in the solution when compared with the usual diffusion modeled by Fick's first and second laws. These features will be considered when the routines for the solution of eqs. (1.2a,b) are discussed subsequently.

- Since the nonlinear diffusion term $-2D(u_2/u_1)u_{1x}$ requires $u_1(x,t)$, it is necessary to integrate eq. (1.2a) along with (1.2b); in other words, u_1 and u_{1x} come from eq. (1.2a). But the solution of eq. (1.2a) requires u_2 from eq. (1.2b) (in the RHS term of eq. (1.2a)). Thus, eqs. (1.2a) and (1.2b) must be integrated together to give $u_1(x,t), u_2(x,t)$ simultaneously. This requirement might be designated as a 2×2 system (two PDEs, eqs. (1.2a,b), in two unknowns, u_1, u_2).

Further discussion of eqs. (1.2) is given in Appendix 1.

Eqs. (1.2a) and (1.2b) are first order in t and therefore each requires an initial condition

$$u_1(x, t = 0) = f_1(x),\ u_2(x, t = 0) = f_2(x) \tag{1.3a,b}$$

where f_1, f_2 are functions to be specified.

Eq. (1.2b) is second order in x and therefore requires two BCs. Since $u_1(x,t)$ is a function of x, and it appears in a second derivative term in eq. (1.2b), we will also assign it two BCs.

$$\frac{\partial u_1(x \to -\infty, t)}{\partial x} = \frac{\partial u_1(x \to \infty, t)}{\partial x} = 0 \tag{1.4a,b}$$

$$\frac{\partial u_2(x \to -\infty, t)}{\partial x} = \frac{\partial u_2(x \to \infty, t)}{\partial x} = 0 \tag{1.4c,d}$$

The analytical solution to eqs. (1.2), (1.3) and (1.4) is ([2], p68)

$$u_1(z) = \left[1 + e^{-cz/D}\right]^{-1} \tag{1.5a}$$

$$u_2(z) = \frac{c^2}{kD} e^{-cz/D} \left[1 + e^{-cz/D}\right]^{-2} \tag{1.5b}$$

where $z = x - ct$; c is a constant to be specified (a velocity). Note that $u_1(z), u_2(z)$ are a function of the single Lagrangian variable z. In other words, these solutions are invariant for a constant value of z, regardless of how x and t may vary. A solution with this property is termed a traveling wave [1]. We will discuss this property further when the numerical solution to eqs. (1.2) is discussed subsequently.

We will use eqs. (1.5) for ICs (1.3) with $t = 0, z = x$.

$$f_1(x) = u_1(x, t = 0) = \left[1 + e^{-cx/D}\right]^{-1} \tag{1.3c}$$

$$f_2(x) = u_2(x, t = 0) = \frac{c^2}{kD} e^{-cx/D} \left[1 + e^{-cx/D}\right]^{-2} \tag{1.3d}$$

Eqs. (1.2) to (1.4) constitute the 2-PDE model to be studied numerically. Also, the analytical solutions, eqs. (1.5), will be used to evaluate the numerical solution.

(1.4) Method of lines solution of 2-PDE chemotaxis model

The method of lines (MOL) is a general procedure for the numerical integration of PDEs in which the derivatives in the boundary value (spatial) independent variables are approximated algebraically, in the present case, by finite differences (FDs). Then only one independent variable remains, in this case, the initial value t. Since we have now only one independent variable, the original PDEs are replaced with an approximating set of ODEs. These ODEs can then be integrated (solved) numerically by any established initial value ODE integrator. In the discussion that follows, we will use the R ODE integrator `lsodes` ([3]). This is the essence of the numerical method of lines.

The R routines for PDEs (1.2), ICs (1.3) and BCs (1.4) follow. The numerical solution from these routines will then be compared with the analytical solution, eqs. (1.5). A main program for the numerical MOL solution is in Listing 1.1.

(1.4.1) Main program

```
#
# Access ODE integrator
  library("deSolve");
#
# Access functions for numerical, analytical solutions
  setwd("c:/R/bme_pde/chap1");
  source("chemo_1.R");
  source("u1_anal.R");
  source("u2_anal.R");
```

```
  source("dss004.R");
#
# Level of output
#
#   ip = 1 - graphical (plotted) solutions
#            (u1(x,t), u2(x,t)) only
#
#   ip = 2 - numerical and graphical solutions
#
  ip=3;
#
# Grid (in x)
  nx=101;xl=-10;xu=15
  xg=seq(from=xl,to=xu,by=0.25);
#
# Parameters
  k=1;D=1;c=1;
  cat(sprintf("\n\n k = %5.2f   D = %5.2f   c = %5.2f\n",k,D,c));
#
# Independent variable for ODE integration
  nout=6;
  tout=seq(from=0,to=5,by=1);
#
# Initial condition (from analytical solutions,t=0)
  u0=rep(0,2*nx);
  for(i in 1:nx){
    u0[i]  =u1_anal(xg[i],tout[1],k,D,c);
    u0[i+n]=u2_anal(xg[i],tout[1],k,D,c);
  }
  ncall=0;
#
# ODE integration
  out=lsodes(y=u0,times=tout,func=chemo_1,parms=NULL)
  nrow(out)
  ncol(out)
#
# Arrays for plotting numerical, analytical solutions
  u1_plot=matrix(0,nrow=nx,ncol=nout);
  u2_plot=matrix(0,nrow=nx,ncol=nout);
 u1a_plot=matrix(0,nrow=nx,ncol=nout);
 u2a_plot=matrix(0,nrow=nx,ncol=nout);
  for(it in 1:nout){
    for(ix in 1:nx){
       u1_plot[ix,it]=out[it,ix+1];
```

```
       u2_plot[ix,it]=out[it,ix+1+nx];
      u1a_plot[ix,it]=u1_anal(xg[ix],tout[it],k,D,c);
      u2a_plot[ix,it]=u2_anal(xg[ix],tout[it],k,D,c);
    }
  }
#
# Display numerical solution
  if(ip==2){
    for(it in 1:nout){
      cat(sprintf("\n    t       x   u1(x,t)  u1_ex(x,t)  u1_err(x,t)"));
      cat(sprintf("\n                u2(x,t)  u2_ex(x,t)  u2_err(x,t)\n"));
      for(ix in 1:nx){
        cat(sprintf("%5.1f%8.2f%10.5f%12.5f%13.6f\n",tout[it],xg[ix],
        u1_plot[ix,it],u1a_plot[ix,it],u1_plot[ix,it]-u1a_plot[ix,it]));
        cat(sprintf("             %10.5f%12.5f%13.6f\n",
        u2_plot[ix,it],u2a_plot[ix,it],u2_plot[ix,it]-u2a_plot[ix,it]));
      }
    }
  }
#
# Calls to ODE routine
  cat(sprintf("\n\n ncall = %5d\n\n",ncall));
#
# Plot u1 numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u1_plot,type="l",xlab="x",ylab="u1(x,t),t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="u1(x,t); solid - num, points - anal;
          t=0,1,2,3,4,5;",lwd=2);
  matpoints(x=xg,y=u1a_plot,xlim=c(xl,xu),col="black",lwd=2)
#
# Plot u2 numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u2_plot,type="l",xlab="x",ylab="u2(x,t),t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="u2(x,t), solid - num, points - anal;
          t=0,1,2,3,4,5;",lwd=2);
  matpoints(x=xg,y=u2a_plot,xlim=c(xl,xu),col="black",lwd=2)
```

Listing 1.1: Main program for the solution of eqs. (1.2)

We can note the following details about Listing 1.1.

- The library of ODE solvers, `deSolve` (with `lsodes` for the subsequent integration of the MOL ODEs) is accessed. Also, four files with the routines to calculate the numerical and analytical solutions of eqs. (1.2) are accessed. Note the use of the forward slash `/` in `setwd` (set working directory).

```
#
# Access ODE integrator
  library("deSolve");
#
# Access functions for numerical, analytical solutions
  setwd("c:/R/bme_pde/chap1");
  source("chemo_1.R");
  source("u1_anal.R");
  source("u2_anal.R");
  source("dss004.R");
```

- Two cases are programmed: `ip=1` for graphical output only and `ip=2` for graphical and numerical output.

```
#
# Level of output
#
#   ip = 1 - graphical (plotted) solutions
#            (u1(x,t), u2(x,t)) only
#
#   ip = 2 - numerical and graphical solutions
#
  ip=2;
```

- A grid in x is defined over the interval $-10 \leq x \leq 15$ with `nx=101` points. This x domain was selected to be essentially infinite as required by BCs (1.4). This characteristic will be explained later. 101 points were determined to be adequate to achieve acceptable accuracy (spatial resolution), and this accuracy is confirmed by comparing the numerical solution to the analytical solution of eqs. (1.5).

```
#
# Grid (in x)
  nx=101;xl=-10;xu=15
  xg=seq(from=xl,to=xu,by=0.25);
```

- The model parameters (constants) are defined numerically and displayed. These parameters appear in eqs. (1.2) and eqs. (1.5).

```
#
# Parameters
  k=1;D=1;c=1;
  cat(sprintf("\n\n k = %5.2f   D = %5.2f   c = %5.2f\n",k,D,c));
```

- Six output values of t are placed in vector `tout`, that is, `tout=0,1,2,3,4,5`.

```
#
# Independent variable for ODE integration
  nout=6;
  tout=seq(from=0,to=5,by=1);
```

A total of six outputs (counting the IC) provides parametric plots in t (for $t = 0, 1, ...5$) as explained subsequently.

- The ICs of eqs. (1.3) are computed from the analytical solutions of eqs. (1.5) with $t =$ `tout[1] = 0` (in routines `u1_anal,u2_anal`) and placed in a single vector `u0` for subsequent use as an input to ODE integrator `lsodes` to start the numerical solution.

```
#
# Initial condition (from analytical solutions,t=0)
  u0=rep(0,2*nx);
  for(i in 1:nx){
    u0[i]   =u1_anal(xg[i],tout[1],k,D,c);
    u0[i+nx]=u2_anal(xg[i],tout[1],k,D,c);
  }
  ncall=0;
```

Note that there are `2*nx = 2*101 = 202` ICs and MOL ODEs that approximate the PDEs, eqs. (1.2). The length of `u0` informs the ODE integrator, `lsodes`, of the number of ODEs to be integrated (`202`). The counter for the calls to the ODE routine (discussed subsequently) is also initialzed.

- The integration of the 202 ODEs is accomplished by a call to `lsodes`.

```
#
# ODE integration
  out=lsodes(y=u0,times=tout,func=chemo_1,parms=NULL)
  nrow(out)
  ncol(out)
```

We can note the following details about the call to `lsodes`.

  - The IC vector `u0` is used to start the ODE integration (the parameter `y` is a reserved name for `lsodes`).
  - The output values of t are in vector `tout` defined previously. These values of t are also returned through the 2D solution array `out` (`times` is a reserved name for `lsodes`).
  - The ODE routine `chemo_1` discussed subsequently is used for the calculation of the 202 ODE derivatives (`func` is a reserved name for `lsodes`).

- The parameter list `parms` is unused. It could pass the parameters `k,D,c` to `chemo_1` but these parameters are already global (a feature of R) and therefore are available in `chemo_1` (`parms` is a reserved name for `lsodes`).
- The ODE solution is returned in a 2D array `out`. In other words, all $n = 202$ ODE solutions are returned in `out` at each of the `nout=6` values of t. However, the dimensions of `out` are `out(6,203)` with an additional value in the second dimension (`203` rather than `202`) for t. This dimensioning of `out` is confirmed by the R utilities `nrow,ncol`; note that if a concluding `;` is not used in calling `nrow,ncol`, the numerical values of the number of rows and columns (`6,203`) are displayed.

- The ODE solution returned in `out` is placed in two 2D arrays, `u1_plot,u2_plot`.

```
#
# Arrays for plotting numerical, analytical solutions
  u1_plot=matrix(0,nrow=nx,ncol=nout);
  u2_plot=matrix(0,nrow=nx,ncol=nout);
 u1a_plot=matrix(0,nrow=nx,ncol=nout);
 u2a_plot=matrix(0,nrow=nx,ncol=nout);
  for(it in 1:nout){
    for(ix in 1:nx){
       u1_plot[ix,it]=out[it,ix+1];
       u2_plot[ix,it]=out[it,ix+1+nx];
      u1a_plot[ix,it]=u1_anal(xg[ix],tout[it],k,D,c);
      u2a_plot[ix,it]=u2_anal(xg[ix],tout[it],k,D,c);
    }
  }
```

  Also, the analytical solutions of eqs. (1.5) are placed in two 2D arrays, `u1a_plot,u2a_plot` so that the numerical and analytical solutions can be compared. The routines for the analytical solutions, `u1_anal,u2_anal`, are discussed subsequently. Note the offset of `1` in the second subscript of `out`, e.g., `ix+1`. This offset is required since the first value of this subscript is used for the values of t, that is, `out[it,1]` has the values $t = 0, 1, 2, 3, 5$ corresponding to `it=1,2,...,6` (this is a property of the numerical solution in `out` from `lsodes`).

- For `ip=2`, the numerical solution is displayed in a tabulated format (in Table 1.1).

```
#
# Display numerical solution
  if(ip==2){
    for(it in 1:nout){
      cat(sprintf("\n    t       x   u1(x,t)  u1_ex(x,t)  u1_err(x,t)"));
      cat(sprintf("\n                u2(x,t)  u2_ex(x,t)  u2_err(x,t)\n"));
      for(ix in 1:nx){
```

```
        cat(sprintf("%5.1f%8.2f%10.5f%12.5f%13.6f\n",tout[it],xg[ix],
        u1_plot[ix,it],u1a_plot[ix,it],u1_plot[ix,it]-u1a_plot[ix,it]));
        cat(sprintf("             %10.5f%12.5f%13.6f\n",
        u2_plot[ix,it],u2a_plot[ix,it],u2_plot[ix,it]-u2a_plot[ix,it]));
      }
    }
  }
```

- The number of calls to `chemo_1` is displayed at the end of the solution.

```
#
# Calls to ODE routine
  cat(sprintf("\n\n ncall = %5d\n\n",ncall));
```

- $u_1(x,t), u_2(x,t)$ are plotted against x with t as a parameter in Figs. (1.1),(1.2).

```
#
# Plot u1 numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u1_plot,type="l",xlab="x",
    ylab="u1(x,t), t=0,1,...,5",xlim=c(xl,xu),lty=1,
    main="u1(x,t); t=0,1,...,5; lines - num, points - anal",lwd=2);
  matpoints(x=xg,y=u1a_plot,xlim=c(xl,xu),col="black",lwd=2)
#
# Plot u2 numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u2_plot,type="l",xlab="x",
    ylab="u2(x,t), t=0,1,...,5",xlim=c(xl,xu),lty=1,
    main="u2(x,t), t=0,1,...,5; lines - num, points - anal",lwd=2);
  matpoints(x=xg,y=u2a_plot,xlim=c(xl,xu),col="black",lwd=2)
```

`par(mfrow=c(1,1))` specifies a 1×1 array of plots, that is, a single plot. The R utility `matplot` plots the 2D array with the numerical solutions, `u1_plot,u2_plot`, and the utility `matpoints` superimposes the analytical solutions in `u1a_plot, u2a_plot` as points. These details plus the others specified with the various arguments (e.g., axis labels, x axis limits, main heading) are clear when considering Figs. (1.1) and (1.2).

We now consider the routines called by the main program of Listing 1.1, starting with `chemo_1`. The numerical and graphical output from Listing 1.1 is then considered.

(1.4.2) ODE routine

```
  chemo_1=function(t,u,parms){
#
```

```
# Function chemo_1 computes the t derivative vectors of u1(x,t), u2(x,t)
#
# One vector to two vectors
  u1=rep(0,nx);u2=rep(0,nx);
  for(i in 1:nx){
    u1[i]=u[i];
    u2[i]=u[i+nx];
  }
#
# u1x, u2x
  u1x=rep(0,nx);u2x=rep(0,nx);
  u1x=dss004(xl,xu,nx,u1);
  u2x=dss004(xl,xu,nx,u2);
#
# BCs
  u1x[1]=0; u1x[nx]=0;
  u2x[1]=0; u2x[nx]=0;
#
# Nonlinear term
  u1u2x=rep(0,nx);
  for(i in 1:nx){
    u1u2x[i]=2*u2[i]/u1[i]*u1x[i];
  }
#
# u1u2xx, u2xx
  u2xx=rep(0,nx);u1u2xx=rep(0,nx);
  u2xx  =dss004(xl,xu,nx,  u2x);
  u1u2xx=dss004(xl,xu,nx,u1u2x);
#
# PDEs
  u1t=rep(0,nx);u2t=rep(0,nx);
  for(i in 1:nx){
    u1t[i]=-k*u2[i];
    u2t[i]=D*(u2xx[i]-u1u2xx[i]);
  }
#
# Two vectors to one vector
  ut=rep(0,2*nx);
  for(i in 1:nx){
    ut[i]   =u1t[i];
    ut[i+nx]=u2t[i];
  }
#
# Increment calls to chemo_1
```

```
  ncall <<- ncall+1;
#
# Return derivative vector
  return(list(c(ut)));
  }
```

Listing 1.2: ODE routine `chemo_1` called by the main program of Listing 1.1

We can note the following details about Listing 1.2.

- The function is defined.

  ```
    chemo_1=function(t,u,parms){
  #
  # Function chemo_1 computes the t derivative vectors of u1(x,t), u2(x,t)
  ```

  The three input arguments are required by `lsodes` which calls `chemo_1` (even though `parms` is not used). `u` is the input vector of the 202 ODE dependent variables. `t` is the current value of the independent variable t.

- The single solution vector `u` is placed in two vectors `u1,u2` to facilitate the programming in terms of the dependent variables of eqs. (1.2).

  ```
  #
  # One vector to two vectors
    u1=rep(0,nx);u2=rep(0,nx);
    for(i in 1:nx){
      u1[i]=u[i];
      u2[i]=u[i+nx];
    }
  ```

- The first derivatives of `u1,u2` with respect to `x` are computed by calls to the library differentiator `dss004`.

  ```
  #
  # u1x, u2x
    u1x=rep(0,nx);u2x=rep(0,nx);
    u1x=dss004(xl,xu,nx,u1);
    u2x=dss004(xl,xu,nx,u2);
  ```

  The boundary values of x, `xl=-10,xu=15`, are set in Listing 1.1 and are effectively for the interval $-\infty \leq x \leq \infty$.

- BCs (1.4) are programmed at `xl=-10,xu=15`. Note that this interval is not symmetric with respect to $x = 0$ because of the movement of the traveling wave solutions of eqs. (1.5) in the positive x direction so that more distance is required for $x > 0$ than for $x < 0$. This feature of the interval in x will be clear when the numerical solution is examined graphically (plotted).

```
#
# BCs
  u1x[1]=0; u1x[nx]=0;
  u2x[1]=0; u2x[nx]=0;
```

The first derivatives of $u1, u2$ are set to zero at the boundaries according to BCs (1.4).

- The nonlinear term in eqs. (1.2b,c), $2(u_2/u_1)u_{1x}$, is computed and placed in array `u1u2x`

```
#
# Nonlinear term
  u1u2x=rep(0,nx);
  for(i in 1:nx){
    u1u2x[i]=2*u2[i]/u1[i]*u1x[i];
  }
```

- The second derivatives in eqs. (1.2b) are computed by `dss004`. The calculation of the second derivative as the derivative of the first derivative is termed stage-wise differentiation. This procedure is put to good use to calculate the derivative of the nonlinear term $2(u_2/u_1)u_{1x}$. This demonstrates a major advantage of the numerical solution of PDEs, that is, nonlinear terms of virtually any form can be accommodated.

```
#
# u1u2xx, u2xx
  u2xx=rep(0,nx);u1u2xx=rep(0,nx);
  u2xx  =dss004(xl,xu,nx,  u2x);
  u1u2xx=dss004(xl,xu,nx,u1u2x);
```

- Eqs. (1.2) are programmed

```
#
# PDEs
  u1t=rep(0,nx);u2t=rep(0,nx);
  for(i in 1:nx){
    u1t[i]=-k*u2[i];
    u2t[i]=D*(u2xx[i]-u1u2xx[i]);
  }
```

Note the resemblance of this programming to the PDEs (eqs. (1.2a,b)) which is an important advantage of the MOL approach to the numerical solution of PDEs. If the term `-u1u2xx[i]` is not included, the resulting PDE `u2t[i]=D*(u2xx[i])` is just Fick's second law, eq. (1.1a) (with constant `D`).

- The two derivative vectors `u1t,u2t` are placed in a single derivative vector `ut` to be returned from `chemo_1`.

```
#
# Two vectors to one vector
  ut=rep(0,2*nx);
  for(i in 1:nx){
    ut[i]  =u1t[i];
    ut[i+nx]=u2t[i];
  }
```

- The counter for the calls to `chemo_1` is incremented (and returned to the main program with `<<-`).

```
#
# Increment calls to chemo_1
  ncall <<- ncall+1;
```

- The derivative vector `ut` is returned to `lsodes` as a list (a requirement of `lsodes`).

```
#
# Return derivative vector
  return(list(c(ut)));
  }
```

  The final } concludes `chemo_1`.

In summary, `chemo_1` receives the independent variable `t` and the dependent variable vector `u` (of length $2(101) = 202$) as RHS (input) arguments and returns a derivative vector `ut`. Note that `u` and `ut` must be of the same length and the derivative of a particular dependent variable `u[i]` must be in position `ut[i]` in vector `ut`.

The straightforward coding in `chemo_1` demonstrates the versatility of the MOL solution of eqs. (1.2), a 2×2 system of simultaneous, nonlinear PDEs. Extensions to more complex PDE systems follows directly from the ideas expressed in `chemo_1`.

To complete the discussion of the R routines, we consider `u1_anal` and `u2_anal` for the analytical solutions of eqs. (1.5) (`u1_anal` and `u2_anal` are called in the main program of Listing 1.1 for numerical definition of the ICs of eqs. (1.3) and plotting of the analytical solutions (1.5)).

```
  u1_anal=function(x,t,k,D,c){
#
# Function u1_anal computes the analytical solution for u1(x,t)
#
  z=x-c*t;
  u1a=1/(1+exp(-c*z/D));
```

```
#
# Return solution
  return(c(u1a));
  }
```

Listing 1.3a: Routine `u1_anal` for the analytical solution of eq. (1.5a)

The programming of eq. (1.5a) in `u1_anal` is straightforward. We can note two details.

- The use of the Lagrangian variable `z=x-c*t` so that the solution is invariant for a particular value of z even though x and t vary. This produces a traveling wave solution as demonstrated in the graphical output described next.
- The value of the solution, `u1a`, is returned to the calling program, in this case the main program of Listing 1.1, as a numerical vector (with one element, that is, as a scalar). This contrasts with `chemo_1` in Listing 1.2 for which the derivative vector is returned as a list (as required by `lsodes`).

Function `u2_anal` is similar to `u1_anal` and follows directly from eq. (1.5b).

```
  u2_anal=function(x,t,k,D,c){
#
# Function u2_anal computes the analytical solution for u2(x,t)
#
  z=x-c*t;
  u2a=(c^2/(k*D))*exp(-c*z/D)/(1+exp(-c*z/D))^2;
#
# Return solution
  return(c(u2a));
  }
```

Listing 1.3b: Routine `u2_anal` for the analytical solution of eq. (1.5b)

This completes the programming of eqs. (1.2) with the associated ICs of eq. (1.3) and the BCs of eqs. (1.4). We now consider the output from the routines in Listings 1.1, 1.2 and 1.3.

(1.5) Model output

Execution of the main program in Listing 1.1 with `ip=2` gives the following abbreviated numerical output.

```
 k =  1.00   D =  1.00   c =  1.00

 > nrow(out)
 [1] 6
 > ncol(out)
```

```
[1] 203

  Output for t=0,1,2,3,4 removed

   t       x   u1(x,t)  u1_ex(x,t)  u1_err(x,t)
               u2(x,t)  u2_ex(x,t)  u2_err(x,t)
 5.0  -10.00   0.00000     0.00000     0.000001
               0.00000     0.00000     0.000003
 5.0   -9.75   0.00000     0.00000     0.000001
               0.00000     0.00000     0.000003
 5.0   -9.50   0.00000     0.00000     0.000001
               0.00000     0.00000     0.000003
 5.0   -9.25   0.00000     0.00000     0.000001
               0.00000     0.00000     0.000003
 5.0   -9.00   0.00000     0.00000     0.000001
               0.00000     0.00000     0.000003
                 .           .
                 .           .
                 .           .
     Output for x = -8.75 to 3.75 removed
                 .           .
                 .           .
                 .           .
 5.0    4.00   0.26894     0.26894    -0.000003
               0.19657     0.19661    -0.000044
 5.0    4.25   0.32081     0.32082    -0.000007
               0.21787     0.21789    -0.000029
 5.0    4.50   0.37754     0.37754    -0.000006
               0.23500     0.23500    -0.000009
 5.0    4.75   0.43782     0.43782     0.000001
               0.24614     0.24613     0.000009
 5.0    5.00   0.50001     0.50000     0.000010
               0.25002     0.25000     0.000018
 5.0    5.25   0.56220     0.56218     0.000019
               0.24615     0.24613     0.000013
 5.0    5.50   0.62248     0.62246     0.000024
               0.23500     0.23500     0.000000
 5.0    5.75   0.67920     0.67918     0.000024
               0.21788     0.21789    -0.000015
 5.0    6.00   0.73108     0.73106     0.000019
               0.19659     0.19661    -0.000025
                 .           .
                 .           .
                 .           .
```

```
    Output for x = 6.25 to 13.75 removed
                .           .
                .           .
                .           .
5.0   14.00   0.99986     0.99988    -0.000016
               0.00014     0.00012     0.000017
5.0   14.25   0.99988     0.99990    -0.000021
               0.00012     0.00010     0.000021
5.0   14.50   0.99990     0.99993    -0.000027
               0.00010     0.00007     0.000027
5.0   14.75   0.99991     0.99994    -0.000035
               0.00009     0.00006     0.000035
5.0   15.00   0.99991     0.99995    -0.000044
               0.00009     0.00005     0.000045

ncall =   295
```

Table 1.1: Selected numerical output from the routines of Listings 1.1, 1.2, 1.3

We can note the following details of the numerical output in Table 1.1.

- The model parameters are displayed.

  ```
  k =  1.00   D =  1.00   c =  1.00
  ```

- The dimensions of the output array from `lsodes` are `out[6,203]` as explained previously.

  ```
  > nrow(out)
  [1] 6
  > ncol(out)
  [1] 203
  ```

  In particular, the second dimension of `out` is `203` rather than `2*101 = 202` in order to include the value of t.

- The output for $t = 0, 1, 2, 3, 4$ is deleted to conserve space. Abbreviated output for $t = 5$ is displayed. In particular, the output for $x = 4$ to $x = 6$ demonstrates the following properties of the numerical and analytical solutions.

  - For $x = 5, t = 5$, with $c = 1$, the Lagrangian variable is $z = x - ct = 5 - (1)(5) = 0$. For this value of z, the numerical and analytical `u1(x,t),u2(x,t)` have the following values.

    ```
    5.0    5.00   0.50001     0.50000     0.000010
                  0.25002     0.25000     0.000018
    ```

The numerical `u1(x=5,t=5) = 0.50001` while the analytical `u1(x=5,t=5) = 0.50000`. Similarly, the numerical `u2(x=5,t=5) = 0.25002` while the analytical `u2(x=5,t=5) = 0.25000`. This agreement between the numerical and analytical solutions is particularly noteworthy since at `x=5,t=5`, `u1(x,t),u2(x,t)` are changing most rapidly (i.e., $z = 0$). In particular, `u2(x,t)` goes through a maximum with rapid change as indicated in Fig. 1.2.

- The computational effort to produce the numerical solution is modest.

```
ncall =  295
```

The plotted output from Listing 1.1 follows as Figs. 1.1, 1.2. We can note the following details about Figs. 1.1, 1.2.

- The agreement between the numerical (solid line) and the analytical solution of eq. (1.5a) (numbers) is quite satisfactory (as confirmed by Table 1.1). Thus, the grid with `nx=101` points appears to give acceptable spatial resolution in x. Of course, the number of grid points could be changed and the effect on the numerical solution of Table 1.1 could then be observed, a form of h refinement (since h is often used to denote the grid spacing in the numerical analysis literature). An alternative for evaluating the solution would be to compare the solution from `dss004` in `chemo_1` (based on five point, fourth order FDs) with, for example, the solution from `dss006` (based on seven point, sixth order FDs). This is a form of p refinement since p is often used to denote the order of an approximation; in the case of `dss004` and `dss006`, $p = 4, 6$, respectively.

- The boundary values $x = -10, 15$ appear to be effectively at $-\infty, \infty$ in the sense that the solution does not change near these boundaries (note the zero derivative conditions of BCs (1.4)).

- The traveling wave characteristic of eqs. (1.2) and (1.5) is clear. That is, the solution moves left to right with a velocity $c = 1$ (in $z = x - ct$); the curves are for the six values $t = 0, 1, ..., 5$. The previous use of *characteristic* is more mathematical than might be appreciated. The Lagrangian variable $z = x - ct$ is generally termed a characteristic of the solution, and for a constant value of z, the solution is invariant; for example, the RHSs of eqs. (1.5) are invariant for a given value of z even though x and t may change.

In conclusion, the MOL solutions from the R routines in Listings 1.1 to 1.3 are in good agreement with the analytical solutions of eqs. (1.5). We now consider some additional programming to elucidate the origin of the solution properties in Table 1.1 and Figs, 1.1, 1.2.

(1.6) Computation of PDE terms

One approach to understanding the solutions in Figs. 1.1 and 1.2 would be to derive an analytical solution such as eqs. (1.5), then study these solutions mathematically.

While this approach was possible in the case of eqs. (1.2) to (1.4), generally the PDE problems that are studied numerically are so complex (e.g., too many nonlinear PDEs) as to preclude an analytical approach. We therefore consider a numerical approach to PDE analysis that can generally be used without an analytical solution, but rather, requires only the numerical solution.

Specifically, we consider PDEs such as eqs. (1.2) which have LHS terms that are functions of the derivatives in the initial value variable t, and RHS terms that are functions of the derivatives in the spatial (boundary value) variable x (there could be more than one spatial variable). Then a detailed examination of the RHS terms gives an indication of the mathematical details that define the variation of the PDE dependent variables (the PDE solutions) with x. Also, a summation of the RHS terms according to the PDE gives an indication of the details of the LHS that define the variation of the numerical solutions with t. The only requirment to do this analysis in x and t is the availability of the numerical solution.

We now consider how this approach can be applied to eqs. (1.2) to (1.4) as an example of the detailed analysis of PDE numerical solutions. The programming for this analysis is the following code that is added to the end of Listing 1.1; nothing else changes in Listing 1.1 except the addition of a third value of the output index `ip`, that is `ip=3`, to initiate execution of the following code.

```
#
# Plotting of PDE RHS terms
  if(ip==3){
#
# 1D arrays of various PDE RHS terms (denoted 1d)
  u1_1d   =rep(0,nx); u2_1d =rep(0,nx);u1u2_1d=rep(0,nx);
  u1x_1d  =rep(0,nx);u2x_1d =rep(0,nx);
  u1u2x_1d=rep(0,nx);u2xx_1d=rep(0,nx);
#
# 2D arrays for plotting (denoted 2d)
  u1x_2d  =matrix(0,nrow=nx,ncol=nout);
  u2x_2d  =matrix(0,nrow=nx,ncol=nout);
  u1u2_2d =matrix(0,nrow=nx,ncol=nout);
  u2xx_2d =matrix(0,nrow=nx,ncol=nout);
  u1u2x_2d=matrix(0,nrow=nx,ncol=nout);
  u1t_2d  =matrix(0,nrow=nx,ncol=nout);
  u2t_2d  =matrix(0,nrow=nx,ncol=nout);
  u1x_2d_anal=matrix(0,nrow=nx,ncol=nout);
  u2x_2d_anal=matrix(0,nrow=nx,ncol=nout);
  u1t_2d_anal=matrix(0,nrow=nx,ncol=nout);
  u2t_2d_anal=matrix(0,nrow=nx,ncol=nout);
#
# PDE RHS terms
  for(it in 1:nout){
```

```
    for(ix in 1:nx){
      u1_1d[ix]=u1_plot[ix,it];
      u2_1d[ix]=u2_plot[ix,it];
    }
    u1x_1d=dss004(xl,xu,nx,u1_1d);
    u2x_1d=dss004(xl,xu,nx,u2_1d);
    u1x_1d[1]=0;u1x_1d[nx]=0;
    u2x_1d[1]=0;u2x_1d[nx]=0;
    for(ix in 1:nx){
      u1u2_1d[ix]=u2_1d[ix]/u1_1d[ix]*u1x_1d[ix];
    }
    u1u2x_1d=dss004(xl,xu,nx,u1u2_1d);
     u2xx_1d=dss004(xl,xu,nx, u2x_1d);
#
#   2D arrays for plotting
    for(ix in 1:n){
#
#     Derivatives of solutions in x
      u1x_2d[ix,it]=u1x_1d[ix];
      u2x_2d[ix,it]=u2x_1d[ix];
      u1u2_2d[ix,it]=u1u2_1d[ix];
      u2xx_2d[ix,it]=u2xx_1d[ix];
      u1u2x_2d[ix,it]=u1u2x_1d[ix];
#
#     Derivatives of solutions in t
      u1t_2d[ix,it]=-k*u2_1d[ix];
      u2t_2d[ix,it]=D*(u2xx_1d[ix]-2*u1u2x_1d[ix]);
#
#     Analytical derivatives of solutions in t
      expz=exp(-c*(xg[ix]-c*tout[it])/D);
      u1x_2d_anal[ix,it] =(1/c)*(1/(1+expz)^2)*expz*(c^2/D);
      u2x_2d_anal[ix,it]=-(1/c)* (c^4/(k*D^2))*expz*(1-expz)/(1+expz)^3;
      u1t_2d_anal[ix,it]=-c*u1x_2d_anal[ix,it];
      u2t_2d_anal[ix,it]=-c*u2x_2d_anal[ix,it];
    }
#
# Next t
  }
#
# Plot Du2_{xx}
  par(mfrow=c(1,1));
  matplot(x=xg,y=D*u2xx_2d,type="l",xlab="x",ylab="Du2_{xx},t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="Du2_{xx}; t=0,1,2,3,4,5;",lwd=2);
#
```

```
# Plot -2D((u2/u1)u1_x)_x
  par(mfrow=c(1,1));
  matplot(x=xg,y=-2*D*u1u2x_2d,type="l",xlab="x",ylab="-2D((u2/u1)u1_x)_x,
          t=0,1,2,3,4,5",xlim=c(xl,xu),lty=1    ,main="-2D((u2/u1)u1_x)_x,
          t=0,1,2,3,4,5;",lwd=2);
#
# Plot u1 derivative in x numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u1x_2d,type="l",xlab="x",ylab="u1(x,t)_x,t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="u1(x,t)_x; solid - num, points - anal;
          t=0,1,2,3,4,5;",lwd=2);
  matpoints(x=xg,y=u1x_2d_anal,xlim=c(xl,xu),col="black",lwd=2)
#
# Plot u2 derivative in x numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u2x_2d,type="l",xlab="x",ylab="u2(x,t)_t,t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="u2(x,t)_x, solid - num, points - anal;
          t=0,1,2,3,4,5;",lwd=2);
  matpoints(x=xg,y=u2x_2d_anal,xlim=c(xl,xu),col="black",lwd=2)
#
# Plot u1 derivative in t numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u1t_2d,type="l",xlab="x",ylab="u1(x,t)_t,t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="u1(x,t)_t; solid - num, points - anal;
          t=0,1,2,3,4,5;",lwd=2);
  matpoints(x=xg,y=u1t_2d_anal,xlim=c(xl,xu),col="black",lwd=2)
#
# Plot u2 derivative in t numerical, analytical
  par(mfrow=c(1,1));
  matplot(x=xg,y=u2t_2d,type="l",xlab="x",ylab="u2(x,t)_t,t=0,1,2,3,4,5",
          xlim=c(xl,xu),lty=1,main="u2(x,t)_t, solid - num, points - anal;
          t=0,1,2,3,4,5;",lwd=2);
  matpoints(x=xg,y=u2t_2d_anal,xlim=c(xl,xu),col="black",lwd=2)
#
# End ip = 3
  }
```

Listing 1.4: Additonal programming to study the individual terms of eqs. (1.2)

We can note the following details of Listing 1.4

- The execution of the code for `ip=3` is initiated with an `if`. Then seven 1D arrays (vectors) are defined via the `rep` utility (these arrays are identified with `1d` in the names). The length of each vector equals the number of points in x, that is `nx`, which is the length of vector `xg` in Listing 1.1.

```
#
# Plotting of PDE RHS terms
  if(ip==3){
#
# 1D arrays of various PDE RHS terms (denoted 1d)
  u1_1d   =rep(0,nx); u2_1d =rep(0,nx);u1u2_1d=rep(0,nx);
  u1x_1d  =rep(0,nx);u2x_1d =rep(0,nx);
  u1u2x_1d=rep(0,nx);u2xx_1d=rep(0,nx);
```

Initially the elements in all of the 1D arrays are zero, and they are then reset (computed) in the subsequent code.

- Similarly, 11 2D arrays are defined via the `matrix` utility (these arrays are identified with `2d` in the names).

```
#
# 2D arrays for plotting (denoted 2d)
  u1x_2d  =matrix(0,nrow=nx,ncol=nout);
  u2x_2d  =matrix(0,nrow=nx,ncol=nout);
  u1u2_2d =matrix(0,nrow=nx,ncol=nout);
  u2xx_2d =matrix(0,nrow=nx,ncol=nout);
  u1u2x_2d=matrix(0,nrow=nx,ncol=nout);
  u1t_2d  =matrix(0,nrow=nx,ncol=nout);
  u2t_2d  =matrix(0,nrow=nx,ncol=nout);
  u1x_2d_anal=matrix(0,nrow=nx,ncol=nout);
  u2x_2d_anal=matrix(0,nrow=nx,ncol=nout);
  u1t_2d_anal=matrix(0,nrow=nx,ncol=nout);
  u2t_2d_anal=matrix(0,nrow=nx,ncol=nout);
```

The column dimension is the number of output points in t, that is `nout`, which is the length of vector `tout` in Listing 1.1. In this way, the variation of the PDE terms in x and t can be plotted. Initially, all $nx \times nout = 101 \times 6 = 606$ elements in these 2D arrays are zero, and then are reset through the subsequent computations.

- The calculations are performed for the six values of t, `t=0,1,2,3,4,5`, with a `for` in `it`. Then the `n=101` values in x, `x=-10,-9.75,...,15` are included with a `for` in `ix`.

```
#
# PDE RHS terms
  for(it in 1:nout){
    for(ix in 1:nx){
      u1_1d[ix]=u1_plot[ix,it];
      u2_1d[ix]=u2_plot[ix,it];
    }
```

The numerical solutions computed previously (in Listing 1.1)) are placed in two 1D arrays `u1_1d,u2_1d` for the variation of the numerical solutions $u_1(x,t), u_2(x,t)$ with x (index `ix`).

- First order derivatives in x are computed by the library differentiator `dss004`.

```
u1x_1d=dss004(xl,xu,nx,u1_1d);
u2x_1d=dss004(xl,xu,nx,u2_1d);
u1x_1d[1]=0;u1x_1d[nx]=0;
u2x_1d[1]=0;u2x_1d[nx]=0;
```

where

Derivative	R variable
$\partial u_1/\partial x$	`u1x_1d`
$\partial u_2/\partial x$	`u2x_1d`

BCs (1.4) are then imposed (to correct the boundary derivatives from `dss004`).

- The nonlinear term and the second derivative in eq. (1.2b) are computed.

```
for(ix in 1:nx){
  u1u2_1d[ix]=u2_1d[ix]/u1_1d[ix]*u1x_1d[ix];
}
u1u2x_1d=dss004(xl,xu,nx,u1u2_1d);
 u2xx_1d=dss004(xl,xu,nx, u2x_1d);
```

where

Derivative	R variable
$(u_2/u_1)\partial u_1/\partial x$	`u1u2_1d`
$\partial\left[(u_2/u_1)\partial u_1/\partial x\right]/\partial x$	`u1u2x_1d`
$\partial^2 u_2/\partial x^2$	`u2xx_1d`

- The 1D terms in x are placed in 2D arrays for subsequent plotting.

```
#
#   2D arrays for plotting
    for(ix in 1:nx){
#
#     Derivatives of solutions in x
```

```
#
#     Derivatives of solutions in x
      u1x_2d[ix,it]=u1x_1d[ix];
      u2x_2d[ix,it]=u2x_1d[ix];
      u1u2_2d[ix,it]=u1u2_1d[ix];
      u2xx_2d[ix,it]=u2xx_1d[ix];
      u1u2x_2d[ix,it]=u1u2x_1d[ix];
```

- The derivatives in t according to eqs. (1.2a,b) are computed and placed in 2D arrays.

```
#
#     Derivatives of solutions in t
      u1t_2d[ix,it]=-k*u2_1d[ix];
      u2t_2d[ix,it]=D*(u2xx_1d[ix]-2*u1u2x_1d[ix]);
```

where

Derivative	R variable
$\partial u_1/\partial t$	`u1t_2d`
$\partial u_2/\partial t$	`u2t_2d`

- The analytical derivatives in t are computed by differentiation of eqs. (1.5) with respect to t and placed in 2D arrays.

```
#
#     Analytical derivatives of solutions in t
      expz=exp(-c*(xg[ix]-c*tout[it])/D);
      u1x_2d_anal[ix,it] =(1/c)*(1/(1+expz)^2)*expz*(c^2/D);
      u2x_2d_anal[ix,it]=-(1/c)* (c^4/(k*D^2))*expz*(1-expz)/(1+expz)^3;
      u1t_2d_anal[ix,it]=-c*u1x_2d_anal[ix,it];
      u2t_2d_anal[ix,it]=-c*u2x_2d_anal[ix,it];
    }
#
# Next t
  }
```

Note the use of the Lagrangian variable $z = x - ct$. The first } concludes the `for` in `ix`. The second } concludes the `for` in `it`.

- Six terms with 2D arrays are plotted against x with t as a parameter.

$$D\partial^2 u_2/\partial x^2 \qquad \text{Fig. 1.3}$$

$$-2D\partial\left[(u_2/u_1)\partial u_1/\partial x\right]/\partial x \qquad \text{Fig. 1.4}$$

$$\partial u_1/\partial x \qquad \text{Fig. 1.5}$$

$$\partial u_2/\partial x \qquad \text{Fig. 1.6}$$

$$\partial u_1/\partial t \qquad \text{Fig. 1.7}$$

$$\partial u_2/\partial t \qquad \text{Fig. 1.8}$$

```
#
# Plot Du2_{xx}
  par(mfrow=c(1,1));
  matplot(x=xg,y=D*u2xx_2d,type="l",xlab="x",
    ylab="Du2_{xx}, t=0,1,...,5",xlim=c(xl,xu),lty=1,
    main="Du2_{xx}; t=0,1,...,5;",lwd=2);
                    .             .
                    .             .
                    .             .
    Coding for Figs. (1.4) to (1.8) removed to conserve space
                    .             .
                    .             .
                    .             .
#
# End ip = 3
  }
```

The `for` with `ip=3` is then concluded.

We can note the following details about the graphical output.
For Fig. 1.3,

- $D\partial^2 u_2/\partial x^2$ moves left to right as a traveling wave (a function of only $z = x - vt$).
- The form of $D\partial^2 u_2/\partial x^2$ is relatively complex with positive and negative values.

For Fig. 1.4,

- $-2D\partial\left[(u_2/u1_)\partial u_1/\partial x\right]/\partial x$ moves left to right as a traveling wave (a function of only $z = x - vt$).
- $-2D\partial\left[(u_2/u_1)\partial u_1/\partial x\right]/\partial x$ also changes sign as $D\partial^2 u_2/\partial x^2$ of Fig. 1.3. The sum of the terms in Figs. 1.3 and 1.4 produces $\partial u_2/\partial t$ of eq. (1.2b) in a rather complicated way as reflected in Fig. 1.6.

For Fig. 1.5,

- $\partial u_1/\partial x$ moves left to right as a traveling wave (a function of only $z = x - vt$).
- The numerical $\partial u_1/\partial x \geq 0$ which follows from the calculation by `dss004`. The analytical $\partial u_1/\partial x \geq 0$ which follows from the analytical solution of eq. (1.5a).
- The solution in Fig. 1.1, $u_1(x,t)$, has only a positive derivative in x as reflected in Fig. 1.5. Also, the location of the largest and smallest values of the derivatives in Fig. 1.5 are reflected in Fig. 1.1.

For Fig. 1.6,

- $\partial u_2/\partial x$ moves left to right as a traveling wave (a function of only $z = x - vt$) and it changes sign.
- The numerical $\partial u_2/\partial x$, which follows from the calculation by `dss004`. The analytical $\partial u2/\partial x$, which follows from the analytical solution of eq. (1.5b).
- The solution in Fig. 1.2, $u_2(x,t)$, has positive and negative derivatives in x as reflected in Fig. 1.6. In other words, the change in the sign of the derivative in Fig. 1.6 produces a pulse in Fig. 1.2.

For Fig. 1.7,

- $\partial u_1/\partial t$ moves left to right as a traveling wave (a function of only $z = x - vt$).
- The numerical $\partial u_1/\partial t \leq 0$ which follows from the RHS of eq. (1.2a) with $k > 0$, The analytical $\partial u_1/\partial t \leq 0$ which follows from the analytical solution of eq. (1.5a).

For Fig. 1.8,

- $\partial u_2/\partial t$ moves left to right as a traveling wave (a function of only $z = x - vt$).
- The numerical $\partial u_2/\partial t$ which follows from the RHS of eq. (1.2b). The analytical $\partial u_2/\partial t$ which follows from the analytical solution of eq. (1.5b).

Figs. (1.3) to (1.8) give a detailed explanation of Figs. (1.1) and (1.2). In particular, all of the terms in eqs. (1.2a) and (1.2b) are functions of only $z = x - vt$, that is, $u_1(z), u_2(z)$ which follows from eqs. (1.5). And $u_1(x,t), u_2(x,t) \geq 0$ as required physically since these two dependent variables represent concentrations.

Since the solutions of eqs. (1.5) are functions of z only, we have the following relationships.

$$\frac{\partial u_1}{\partial x} = \frac{du_1}{dz}\frac{\partial z}{\partial x} = \frac{du_1}{dz}(1)$$
$$\frac{\partial u_1}{\partial t} = \frac{du_1}{dz}\frac{\partial z}{\partial t} = \frac{du_1}{dz}(-c)$$

from which it follows that

$$\frac{\partial u_1}{\partial t} = (-c)\frac{\partial u_1}{\partial x} \tag{1.6a}$$

Also, for u_2,

$$\frac{\partial u_2}{\partial t} = (-c)\frac{\partial u_2}{\partial x} \tag{1.6b}$$

Eqs. (1.6) were used previously to compute the analytical $\partial u_1/\partial t, \partial u_2/\partial t$ (in Figs. 1.7,1.8) from the analytical $\partial u_1/\partial x, \partial u_2/\partial x$ (in Figs. 1.5,1.6).

```
u1t_2d_anal[ix,it]=-c*u1x_2d_anal[ix,it];
u2t_2d_anal[ix,it]=-c*u2x_2d_anal[ix,it];
```

(from Listing 1.4). However, the numerical $\partial u_1/\partial x, \partial u_2/\partial x$ were computed by numerical differentiation of the solutions, $u_1(x,t), u_2(x,t)$ (using `dss004`) while the numerical $\partial u_1/\partial t, \partial u_2/\partial t$ were calculated as the LHS derivatives of eqs. (1.2a), (1.2b). The agreement of the various derivatives in Figs. 1.5, 1.6, 1.7, and 1.8 illustrates the property that the solutions of eqs. (1.2) are a function of only the Lagrangian variable $z = x - ct$ (a traveling wave solution); this property is stated with eqs. (1.5).

In other words, $\partial u_1/\partial t, \partial u_2/\partial t$ could have been calculated directly from the analytical solutions of eqs. (1.5) (rather than from eqs. (1.6)), but this variation indicates an important property of traveling wave solutions, that is, the partial derivatives in x and t are multiples of $(-c)^{\pm p}$) where p is the order of the derivatives, as illustrated by eqs. (1.6) with $p = 1$. The change in the sign of the derivatives in x and t according to eqs. (1.6) is evident by comparing Figs. 1.5 and 1.7, and Figs. 1.6 and 1.8.

As a point of terminology, $u_1(x,t), u_2(x,t)$ is termed an Eulerian solution (fixed frame in x) while the equivalent $u_1(z), u_2(z)$ is termed a Lagrangian solution (moving frame in x).

(1.7) Conclusions

The preceding example of eqs. (1.2) indicates that the calculation of a numerical solution for a nonlinear system of PDEs is straightforward. Further, experimentation with the PDEs can be easily accomplished, e.g., variation of the parameters such as k, D, c and even the form of the PDEs is straightforward, e.g., variation in the RHS of eq. (1.2b). For example, we could drop the nonlinear term in eq. (1.2b), $((u_2/u_1)u_{1x})_x$, and compute a solution to the simplified eq. (1.2b) (now just Fick's second law, eq. (1.1a)). Comparison of the two solutions (with and without $((u_2/u_1)u_{1x})_x$) would give another indication of the effect of this term. For example, eq. (1.1a) would not have a traveling wave solution; rather just conventional diffusion would cause a smoothing of $u_2(x,t)$ to a constant value in x, a fundamentally different type of solution.

In the present example, we had an analytical solution available that could be used to evaluate the accuracy of the numerical solution, but this usually is not the case. Rather, we use numerical methods because analytical solutions are generally not available.

The use of library routines such as `lsodes` ([3]) and `dss004` ([1]) substantially facilitates the calculation of a numerical PDE solution.

References

[1] Griffiths, G.W., and W.E. Schiesser (2012), *Traveling Wave Analysis of Partial Differential Equations*, Elsevier, Burlington, MA, USA

[2] Murray, J.D. (2003), *Mathematical Biology, II: Spatial Models and Biomedical Applications*, Third Edition

[3] Soetaert, K., J. Cash and F. Mazzia (2012), *Solving Differential Equations in R*, Springer-Verlag, Heidelberg, Germany

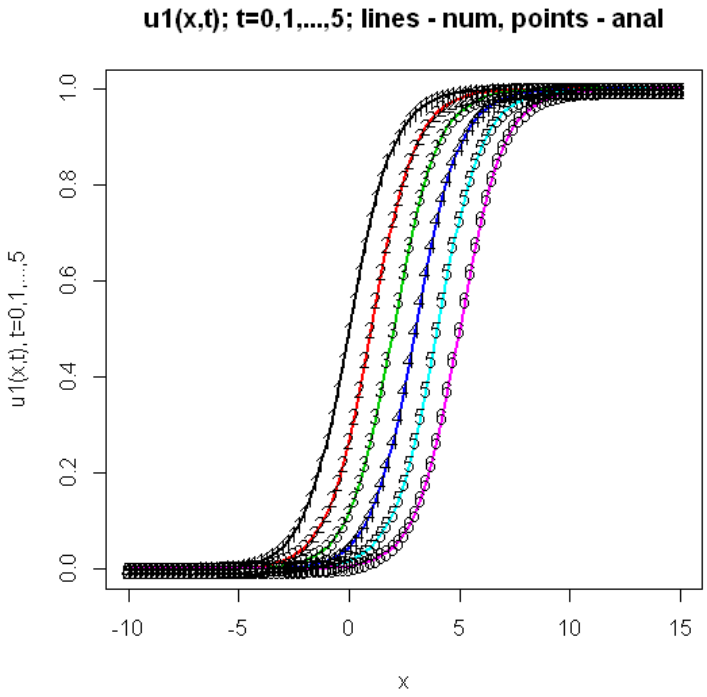

Figure 1.1: $u_1(x,t)$ vs x with t as a parameter

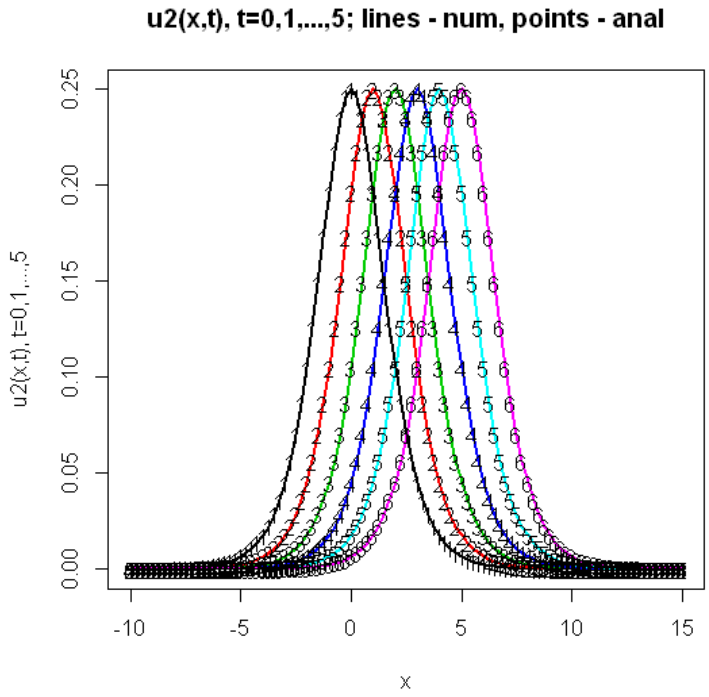

Figure 1.2: $u_2(x,t)$ vs x with t as a parameter

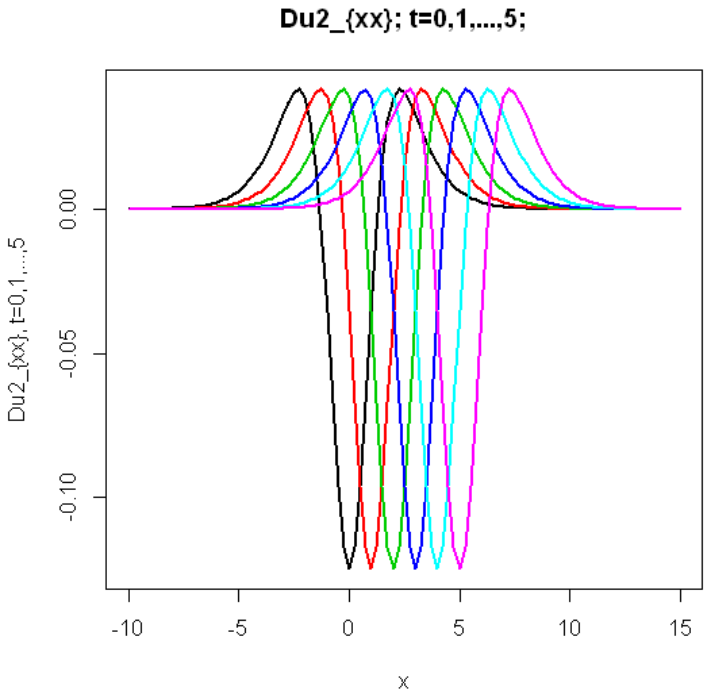

Figure 1.3: $D\partial^2 u_2/\partial x^2$ vs x with t as a parameter

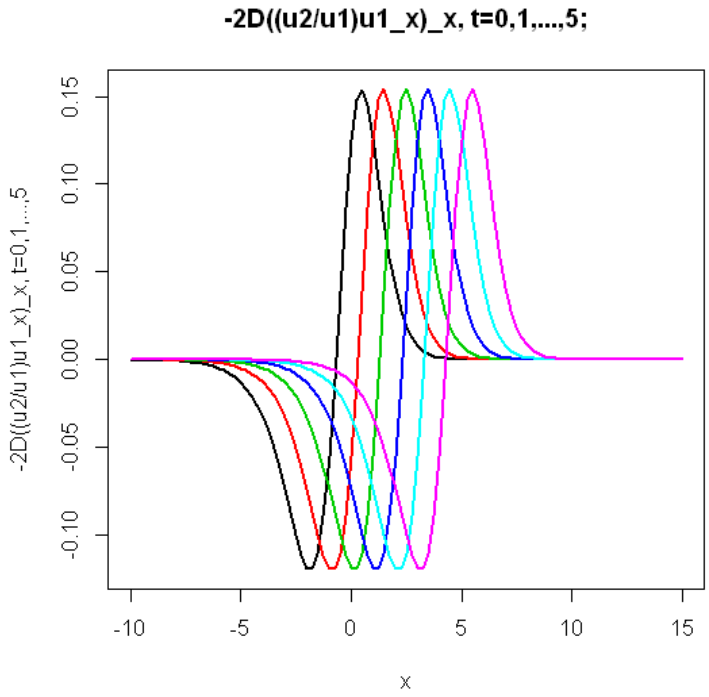

Figure 1.4: $-2D\partial\left[(u_2/u_1)\partial u_1/\partial x\right]/\partial x$ vs x with t as a parameter

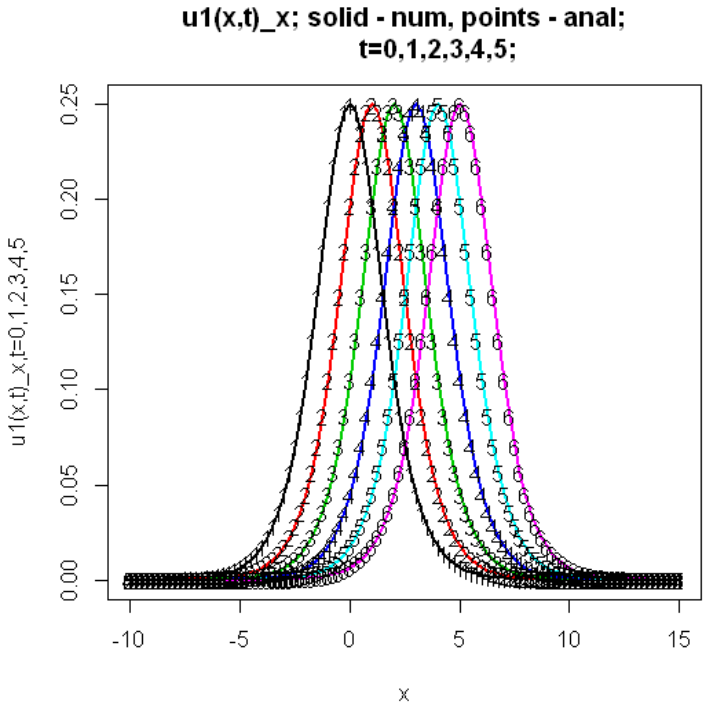

Figure 1.5: $\partial u_1/\partial x$ vs x with t as a parameter

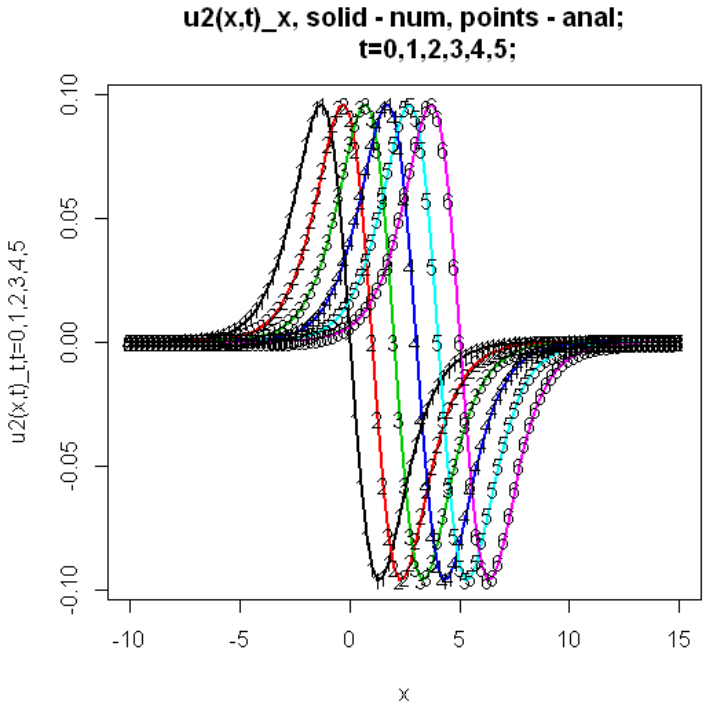

Figure 1.6: $\partial u_2/\partial x$ vs x with t as a parameter

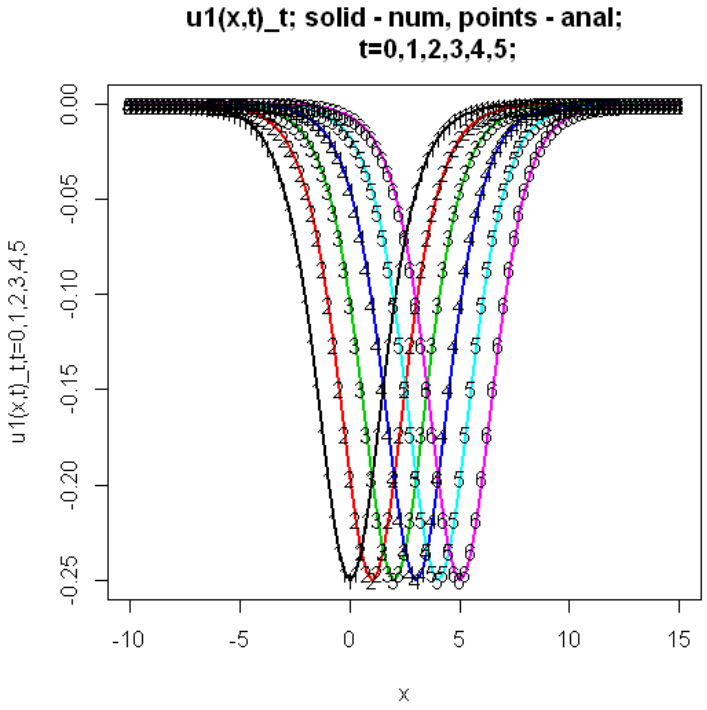

Figure 1.7: $\partial u_1/\partial t$ vs x with t as a parameter

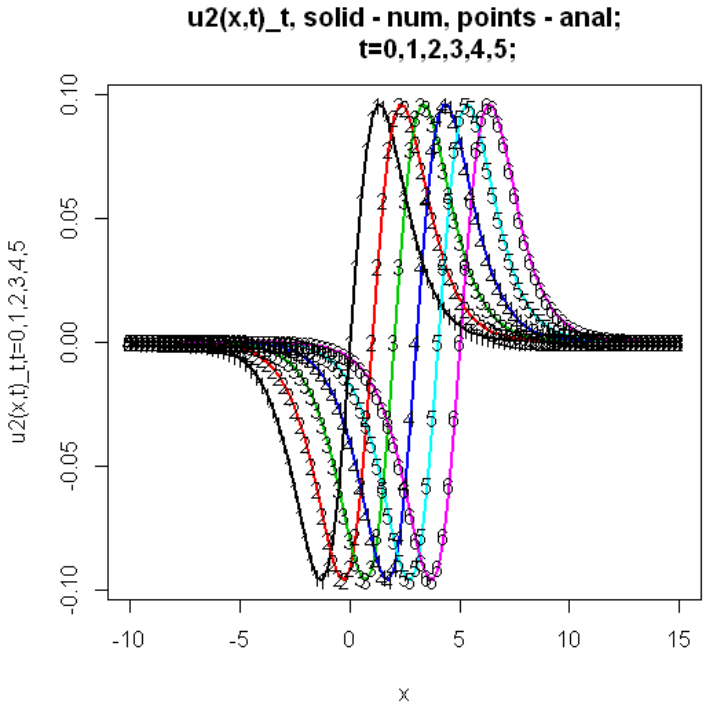

Figure 1.8: $\partial u_2/\partial t$ vs x with t as a parameter