13

Simultaneous, Nonlinear, Two-Dimensional Partial Differential Equations in Cylindrical Coordinates

This partial differential equation (PDE) application introduces the following mathematical concepts and computational methods:

1. Method of lines (MOL) solution of two simultaneous, nonlinear, two-dimensional (2D) PDEs in cylindrical coordinates.
2. The solution of PDEs with variable coefficients.
3. Analysis to regularize a singular point ($r = 0$ in cylindrical coordinates).
4. Detailed derivation of the PDEs based on conservation principles.
5. The origin of convection–diffusion–reaction PDEs.
6. Analysis of physical units to ensure consistency of a mathematical model.
7. Implementation of Dirichlet, Neumann, and third-type (mixed, Robin) boundary conditions.
8. Inclusion of nonlinear source terms in the PDE model.
9. Examination of the numerical and graphical output from the computer code to gain insight into the performance of the physical system described by the PDE model, which then suggests other conditions (e.g., model parameters) that might be investigated using the model and associated code.

We start with the derivation of the PDEs. The physical system is a convection–diffusion–reaction system, for example, a tubular reactor, which is modeled in cylindrical coordinates, (r, z) (we assume angular symmetry so that the third cylindrical coordinate, θ, is not required). The PDE system has two dependent variables, c_a and T_k, which physically are the reacting fluid concentration, c_a, and the fluid temperature, T_k. Thus we require two PDEs.

These two dependent variables are functions of three independent variables: r, the radial coordinate; z, the axial coordinate; and t, time, as illustrated by Figure 13.1. The solution will be $c_a(r, z, t)$ and $T_k(r, z, t)$ in numerical form as a function of r, z, t.

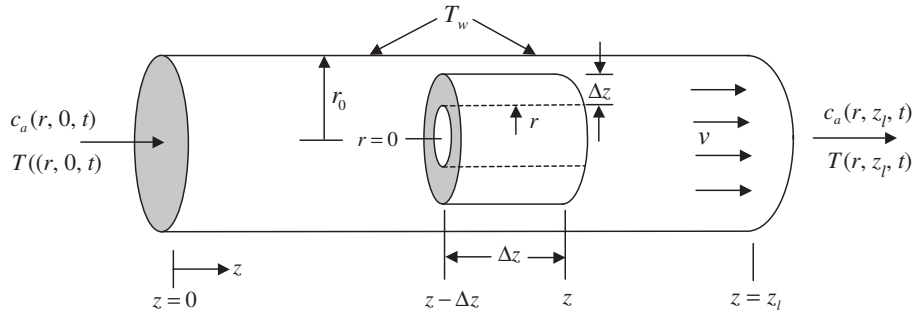

Figure 13.1. Representation of a convection–diffusion–reaction system in cylindrical coordinates

The mass balance for an incremental element of length Δz is (see Figure 13.1)

$$2\pi r\Delta r\Delta z\frac{\partial c_a}{\partial t} = 2\pi r\Delta z q_m|_r - 2\pi(r+\Delta r)\Delta z q_m|_{r+\Delta r} + 2\pi r\Delta r v c_a|_{z-\Delta z} - 2\pi r\Delta r v c_a|_z - 2\pi r\Delta r\Delta z k_r c_a^2$$

Division by $2\pi r\Delta r\Delta z$ and minor rearrangement gives

$$\frac{\partial c_a}{\partial t} = -\frac{(r+\Delta r)q_m|_{r+\Delta r} - r q_m|_r}{r\Delta r} - v\left(\frac{c_a|_z - c_a|_{z-\Delta z}}{\Delta z}\right) - k_r c_a^2$$

or in the limit $r \to 0$, $\Delta z \to 0$,

$$\frac{\partial c_a}{\partial t} = -\frac{1}{r}\frac{\partial(r q_m)}{\partial r} - v\frac{\partial c_a}{\partial z} - k_r c_a^2$$

We discuss in an appendix to this chapter the physical meaning of each of the terms in this PDE. Briefly, here is a summary. For the differential volume in r and z,

$$\frac{\partial c_a}{\partial t} \quad \text{rate of mass accumulation per unit volume}$$

$$-\frac{1}{r}\frac{\partial(r q_m)}{\partial r} \quad \text{rate of radial mass diffusion per unit volume}$$

$$-v\frac{\partial c_a}{\partial z} \quad \text{rate of axial mass convection per unit volume}$$

$$-k_r c_a^2 \quad \text{rate of reaction mass generation per unit volume}$$

If we now assume *Fick's first law* for the flux with a mass diffusivity, D_c,

$$q_m = -D_c\frac{\partial c_a}{\partial r} \tag{13.1}$$

we obtain

$$\frac{\partial c_a}{\partial t} = \frac{D_c}{r}\frac{\partial\left(r\frac{\partial c_a}{\partial r}\right)}{\partial r} - v\frac{\partial c_a}{\partial z} - k_r c_a^2$$

or expanding the radial group,

$$\frac{\partial c_a}{\partial t} = D_c\left(\frac{\partial^2 c_a}{\partial r^2} + \frac{1}{r}\frac{\partial c_a}{\partial r}\right) - v\frac{\partial c_a}{\partial z} - k_r c_a^2 \tag{13.2}$$

Equation (13.2) is the required material balance for $c_a(r, z, t)$ and would be sufficient for the calculation of $c_a(r, z, t)$ except that we will consider the case of an *exothermic reaction* that liberates heat so that system is *not isothermal*. That is, the reacting fluid temperature varies as a function of r, z, and t so that a second PDE is required, an energy balance, for the computation of the temperature $T_k(r, z, t)$. Further, Eq. (13.2) is coupled to the temperature through the reaction rate coefficient k_r as explained in the derivation of the energy balance.

Note also that the volumetric rate of reaction in Eq. (13.2), $k_r c_a^2$, is *second order*; that is, the rate of reaction is proportional to the square of the concentration c_a. This *kinetic term is therefore a source of nonlinearity* in the model.

The thermal energy balance for the incremental section is

$$\begin{aligned} 2\pi r\Delta r\Delta z\rho C_p\frac{\partial T_k}{\partial t} &= 2\pi r\Delta z q_h|_r - 2\pi(r+\Delta r)\Delta z q_h|_{r+\Delta r} \\ &\quad + 2\pi r\Delta r v\rho C_p T_k|_{z-\Delta z} - 2\pi r\Delta r v\rho C_p T_k|_z \\ &\quad - \Delta H 2\pi r\Delta r\Delta z k_r c_a^2 \end{aligned}$$

Division by $2\pi r\Delta r\Delta z\rho$ gives

$$C_p\frac{\partial T_k}{\partial t} = -\frac{(r+\Delta r)q_h|_{r+\Delta r} - rq_h|_r}{r\Delta r\rho} - vC_p\frac{(T_k|_z - T_k|_{z-\Delta z})}{\Delta z} - \frac{\Delta H k_r}{\rho}c_a^2$$

or in the limit $\Delta r \to 0$, $\Delta z \to 0$,

$$C_p\frac{\partial T_k}{\partial t} = -\frac{1}{\rho r}\frac{\partial(rq_h)}{\partial r} - vC_p\frac{\partial T_k}{\partial z} - \frac{\Delta H k_r}{\rho}c_a^2$$

We discuss in an appendix to this chapter the physical meaning of each of the terms in this PDE. Briefly, here is a summary. For the differential volume in r and z,

$C_p\frac{\partial T_k}{\partial t}$	rate of thermal energy accumulation per unit mass
$-\frac{1}{\rho r}\frac{\partial(rq_h)}{\partial r}$	rate of radial heat conduction per unit mass
$-vC_p\frac{\partial T_k}{\partial z}$	rate of axial thermal energy convection per unit mass
$-\frac{\Delta H k_r}{\rho}c_a^2$	rate of reaction thermal energy generation per unit mass

Also, there is one technical detail about this energy balance we should mention. The derivative in t is based on the specific heat C_p reflecting the fluid enthalpy. More generally, an energy balance is based on the specific heat C_v reflecting the fluid internal energy. For a liquid, for which pressure effects are negligible, the two are essentially the same, and therefore we use C_p.

If we now assume *Fourier's first law* for the flux with a thermal conductivity, k,

$$q_h = -k\frac{\partial T_k}{\partial r} \tag{13.3}$$

we obtain (after division by C_p)

$$\frac{\partial T_k}{\partial t} = \frac{D_t}{r}\frac{\partial\left(r\dfrac{\partial T_k}{\partial r}\right)}{\partial r} - v\frac{\partial T_k}{\partial z} - \frac{\Delta H k_r}{\rho C_p}c_a^2$$

or expanding the radial group,

$$\frac{\partial T_k}{\partial t} = D_t\left(\frac{\partial^2 T_k}{\partial r^2} + \frac{1}{r}\frac{\partial T_k}{\partial r}\right) - v\frac{\partial T_k}{\partial z} - \frac{\Delta H k_r}{\rho C_p}c_a^2 \tag{13.4}$$

where $D_t = k/\rho C_p$. Equation (13.4) is the required energy balance for $T(r, z, t)$.

The reaction rate constant, k_r, is given by

$$k_r = k_0\, e^{-E/(RT_k)} \tag{13.5}$$

Note from Eq. (13.5) that k_r is also nonlinearly related to the temperature T_k as e^{-E/RT_k}. This *Arrhenius temperature dependency* is a strong nonlinearity, and explains why chemical reaction rates are generally strongly dependent on temperature; of course, this depends, to some extent, on the magnitude of the *activation energy*, E. The variables and parameters of Eqs. (13.1)–(13.5) and the subsequent auxiliary conditions are summarized in Table 13.1 (in cgs units).

Equation (13.2) requires one initial condition (IC) in t (since it is first order in t), two boundary conditions (BCs) in r (since it is second order in r), and one BC in z (since it is first order in z).

$$c_a(r, z, t = 0) = c_{a0} \tag{13.6}$$

$$\frac{\partial c_a(r = 0, z, t)}{\partial r} = 0, \tag{13.7}$$

$$\frac{\partial c_a(r = r_0, z, t)}{\partial r} = 0 \tag{13.8}$$

$$c_a(r, z = 0, t) = c_{ae} \tag{13.9}$$

Equation (13.4) also requires one IC in t, two BCs in r, and one BC in z.

$$T_k(r, z, t = 0) = T_{k0} \tag{13.10}$$

$$\frac{\partial T_k(r = 0, z, t)}{\partial r} = 0 \tag{13.11}$$

$$k\frac{\partial T_k(r = r_0, z, t)}{\partial r} = h(T_w - T_k(r = r_0, z, t)) \tag{13.12}$$

$$T_k(r, z = 0, t) = T_{ke} \tag{13.13}$$

The solution to Eqs. (13.1)–(13.13) gives $c_a(r, z, t)$, $T_k(r, z, t)$ as a function of the independent variables r, z, t.

Table 13.1. Variables and parameters of the 2D PDE model

Variable/parameter	Symbol	Units/value
Reactant concentration	$c_a(r, z, t)$	gmol/cm^3
Temperature	$T_k(r, z, t)$	K
Reaction rate constant	k_r	cm^3/(gmol·s)
Time	t	s
Radial position	r	cm
Axial position	z	cm
Mass diffusion flux	q_m	gmol/(cm^2 · s)
Energy conduction flux	q_h	cal/(cm^2 · s)
Entering concentration	c_{ae}	0.01 gmol/cm^3
Entering temperature	T_{ke}	305 K
Initial concentration	c_{a0}	0 gmol/cm^3
Initial temperature	T_{k0}	305 K
Wall temperature	T_w	355 K
Reactor radius	r_0	2 cm
Reactor length	z_l	100 cm
Linear fluid velocity	v	1.0 cm/s
Mass diffusivity	D_c	0.1 cm^2/s
Thermal diffusivity	$D_t = k/(\rho C_p)$	0.1 cm^2/s
Fluid density	ρ	1.0 g/cm^3
Fluid specific heat	C_p	0.5 cal/(g·K)
Heat of reaction	ΔH	−10,000 cal/gmol
Specific rate constant	k_0	1.5×10^9 cm^3/(gmol·s)
Activation energy	E	15,000 cal/(gmol·K)
Gas constant	R	1.987 cal/gmol·K
Thermal conductivity	k	0.01 (cal·cm)/(s·cm^2·K)
Heat transfer coefficient	h	0.01 cal/(s·cm^2·K)

For physical problems, an important check of a PDE is *a check on the units of each term in the PDE*. Specifically, the units should be *consistent throughout each equation and make sense physically*. A units check of Eqs. (13.1)–(13.13) is detailed in an appendix to this chapter. For further detailed discussion related to chemical reactors, the reader is referred to [1].

A main program for the MOL of the preceding equations is given in Listing 13.1.

```
%
% Clear previous files
  clear all
  clc
%
% Global area
  global     nr     nz     dr     dz    drs    dzs...
               r      z     Dc     Dt     ca     Tk...
             cae    Tke      h      k      E      R...
             rk0      v    rho     Cp     Tw     dH...
           ncall
```

```
%
% Model parameters
  ca0=0.0;
  cae=0.01;
  Tk0=305.0;
  Tke=305.0;
  Tw=355.0;
  r0=2.0;
  zl=100.0;
  v=1.0;
  Dc=0.1;
  Dt=0.1;
  k=0.01;
  h=0.01;
  rho=1.0;
  Cp=0.5;
  rk0=1.5e+09;
  dH=-10000.0;
  E=15000.0;
  R=1.987;
%
% Grid in axial direction
  nz=20;
  dz=zl/nz;
  for i=1:nz
    z(i)=i*dz;
  end
%
% Grid in radial direction
  nr=7;
  dr=r0/(nr-1);
  for j=1:nr
    r(j)=(j-1)*dr;
  end
  drs=dr^2;
%
% Independent variable for ODE integration
  tf=200.0;
  tout=[0.0:50.0:tf]';
  nout=5;
  ncall=0;
%
% Initial condition
  for i=1:nz
  for j=1:nr
    ca(i,j)=ca0;
    Tk(i,j)=Tk0;
```

```
    y0((i-1)*nr+j)=ca(i,j);
    y0((i-1)*nr+j+nz*nr)=Tk(i,j);
  end
  end
%
% ODE integration
  reltol=1.0e-04; abstol=1.0e-04;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
  mf=2;
  if(mf==1)[t,y]=ode15s(@pde_1,tout,y0,options);end
  if(mf==2)[t,y]=ode15s(@pde_2,tout,y0,options);end
%
% 1D to 2D matrices
  for it=1:nout
  for i=1:nz
  for j=1:nr
    ca(it,i,j)=y(it,(i-1)*nr+j);
    Tk(it,i,j)=y(it,(i-1)*nr+j+nz*nr);
  end
  end
  end
%
% Display a heading and centerline output
  fprintf('\n  nr = %2d   nz = %2d\n',nr,nz);
  for it=1:nout
    fprintf('\n t = %4.1f\n',t(it));
  for i=1:nz
    fprintf(' z = %5.1f   ca(r=0,z,t) = %8.5f
              Tk(r=0,z,t) = %8.2f\n',z(i),ca(it,i,1), ...
              Tk(it,i,1));
  end
  end
  fprintf('\n ncall = %5d\n',ncall);
%
% Parametric plots
%
% Axial profiles
  figure(1);
  subplot(2,2,1)
  plot(z,ca(2,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=50)'); xlabel('z');
         ylabel('ca(r=0,z,t=50)')
  subplot(2,2,2)
  plot(z,Tk(2,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=50)'); xlabel('z');
         ylabel('Tk(r=0,z,t=50)')
  subplot(2,2,3)
```

```
  plot(z,ca(3,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=100)'); xlabel('z');
        ylabel('ca(r=0,z,t=100)')
  subplot(2,2,4)
  plot(z,Tk(3,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=100)'); xlabel('z');
        ylabel('Tk(r=0,z,t=100)')
  figure(2);
  subplot(2,2,1)
  plot(z,ca(4,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=150)'); xlabel('z');
        ylabel('ca(r=0,z,t=150)')
  subplot(2,2,2)
  plot(z,Tk(4,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=150)'); xlabel('z');
        ylabel('Tk(r=0,z,t=150)')
  subplot(2,2,3)
  plot(z,ca(5,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=200)'); xlabel('z');
        ylabel('ca(r=0,z,t=200)')
  subplot(2,2,4)
  plot(z,Tk(5,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=200)'); xlabel('z');
        ylabel('Tk(r=0,z,t=200)')
%
% Radial profiles (at t = tf)
  for i=1:nz,
  for j=1:nr
    ca_rad(i,j)=ca(5,i,j);
    Tk_rad(i,j)=Tk(5,i,j);
  end
  end
  figure(3);
  subplot(2,2,1)
  plot(r,ca_rad(1,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=5,t=200)'); xlabel('r');
        ylabel('ca(r,z=5,t=200)')
  subplot(2,2,2)
  plot(r,ca_rad(7,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=35,t=200)'); xlabel('r');
        ylabel('ca(r,z=35,t=200)')
  subplot(2,2,3)
  plot(r,ca_rad(14,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=70,t=200)'); xlabel('r');
        ylabel('ca(r,z=70,t=200)')
  subplot(2,2,4)
  plot(r,ca_rad(20,:)); axis([0 r0 0.0 0.01]);
```

```
  title('ca(r,z=100,t=200)'); xlabel('r');
        ylabel('ca(r,z=100,t=200)')
  figure(4);
  subplot(2,2,1)
  plot(r,Tk_rad(1,:)); axis([0 r0 305 450]);
  title('Tk(r,z=5,t=200)'); xlabel('r');
        ylabel('Tk(r,z=5,t=200)')
  subplot(2,2,2)
  plot(r,Tk_rad(7,:)); axis([0 r0 305 450]);
  title('Tk(r,z=35,t=200)'); xlabel('r');
        ylabel('Tk(r,z=35,t=200)')
  subplot(2,2,3)
  plot(r,Tk_rad(14,:)); axis([0 r0 305 450]);
  title('Tk(r,z=70,t=200)'); xlabel('r');
        ylabel('Tk(r,z=70,t=200)')
  subplot(2,2,4)
  plot(r,Tk_rad(20,:)); axis([0 r0 305 450]);
  title('Tk(r,z=100,t=200)'); xlabel('r');
        ylabel('Tk(r,z=100,t=200)')
%
% 3D plotting
  figure()
  surf(r,z,Tk_rad)
  axis([0 r0 0 zl 0 500]);
  xlabel('r - radial distance');
  ylabel('z - axial distance');
  zlabel('Tk - reactant temperature');
  title([{'Surface & contour plot of reactant temperature,
       Tk(r,z),'},{ 'at time t=200.0'}]);
  view(-130,62);
  figure()
  surf(r,z,ca_rad)
  axis([0 r0 0 zl 0 0.01]);
  xlabel('r - radial distance');
  ylabel('z - axial distance');
  zlabel('ca - reactant concentration');
  title([{'Surface & contour plot of reactant concentration,
       ca(r,z),'},{ 'at time t=200.0'}]);
  view(-130,62);
  rotate3d on
```

Listing 13.1. Main program `pde_1_main.m` for the solution of Eqs. (13.1)–(13.13)

We can note the following points about this main program:

1. After specification of a *global* area, the model parameters are defined numerically in accordance with the values listed in Table 13.1. These numerical values are then passed as global variables to the MOL ODE routine, as well as the dependent variables CA and TK of Eqs. (13.2) and (13.4), and parameters relating to the spatial grid in r and z (discussed next).

```
%
% Clear previous files
  clear all
  clc
%
% Global area
  global     nr      nz      dr      dz     drs     dzs...
               r       z      Dc      Dt      ca      Tk...
             cae     Tke       h       k       E       R...
             rk0       v     rho      Cp      Tw      dH...
           ncall
%
% Model parameters
  ca0=0.0;
  cae=0.01;
  Tk0=305.0;
  Tke=305.0;
  Tw=355.0;
  r0=2.0;
  zl=100.0;
  v=1.0;
  Dc=0.1;
  Dt=0.1;
  k=0.01;
  h=0.01;
  rho=1.0;
  Cp=0.5;
  rk0=1.5e+09;
  dH=-10000.0;
  E=15000.0;
  R=1.987;
```

2. The spatial grids in z and r are defined. Note that in the case of z, the grid starts at z(1)=dz, while for r, the grid starts at r(1)=0. The reason for this difference (the grid in z does not start at z=0) is explained subsequently. The total number of grid points is (7)(20) = 140, and since there are two PDEs, Eqs. (13.2) and (13.4), there will be (2)(140) = 280 ordinary differential equations (ODEs) programmed in the ODE routine.

```
%
% Grid in axial direction
  nz=20;
  dz=zl/nz;
  for i=1:nz
    z(i)=i*dz;
  end
%
% Grid in radial direction
  nr=7;
  dr=r0/(nr-1);
  for j=1:nr
    r(j)=(j-1)*dr;
  end
  drs=dr^2;
```

3. The interval in t is $0 \leq t \leq 200$ with an output interval of 50 so that the solution will be displayed five times.

```
%
% Independent variable for ODE integration
  tf=200.0;
  tout=[0.0:50.0:tf]';
  nout=5;
  ncall=0;
```

4. ICs (13.6) and (13.10) are programmed as

```
%
% Initial condition
  for i=1:nz
  for j=1:nr
    ca(i,j)=ca0;
    Tk(i,j)=Tk0;
    y0((i-1)*nr+j)=ca(i,j);
    y0((i-1)*nr+j+nz*nr)=Tk(i,j);
  end
  end
```

The two 2D ICs are then converted to a single 1D IC vector, `y0` (with 280 elements). The logic of this conversion of the two 2D matrices, `ca(i,j)` and

Tk(i,j), to a 1D vector (matrix) y0 in the two nested for loops should be studied to ensure an understanding of this process. Also, refer to Chapter 10 where this operation is discussed more fully and the Matlab reshape function has been used for a similar conversion.

5. The system of 280 ODEs is then integrated by a call to Matlab integrator ode15s. The ODE routine is pde_1.

```
%
% ODE integration
  reltol=1.0e-04; abstol=1.0e-04;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
  mf=2;
  if(mf==1)[t,y]=ode15s(@pde_1,tout,y0,options);end
  if(mf==2)[t,y]=ode15s(@pde_2,tout,y0,options);end
%
% 1D to 2D matrices
  for it=1:nout
  for i=1:nz
  for j=1:nr
    ca(it,i,j)=y(it,(i-1)*nr+j);
    Tk(it,i,j)=y(it,(i-1)*nr+j+nz*nr);
  end
  end
  end
```

The solution matrix y returned by ode15s is then converted to matrices ca and Tk that are now 3D to include the variation in t (as the third subscript it).

6. The centerline c_a and T_k (at $r = 0$ corresponding to j=1) are then displayed as a function of z with t as a parameter (by two nested for loops).

```
%
% Display a heading and centerline output
  fprintf('\n  nr = %2d   nz = %2d\n',nr,nz);
  for it=1:nout
    fprintf('\n t = %4.1f\n',t(it));
  for i=1:nz
    fprintf(' z = %5.1f   ca(r=0,z,t) = %8.5f   Tk(r=0,z,t)
            = %8.2f\n',z(i),ca(it,i,1),Tk(it,i,1));
  end
  end
  fprintf('\n ncall = %5d\n',ncall);
```

7. The centerline c_a and T_k are plotted as profiles in z with t as a parameter for the plots (at $t = 50, 100, 150, 200$).

```
%
% Parametric plots
%
% Axial profiles
  figure(1);
  subplot(2,2,1)
  plot(z,ca(2,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=50)'); xlabel('z');
        ylabel('ca(r=0,z,t=50)')
  subplot(2,2,2)
  plot(z,Tk(2,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=50)'); xlabel('z');
        ylabel('Tk(r=0,z,t=50)')
  subplot(2,2,3)
  plot(z,ca(3,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=100)'); xlabel('z');
        ylabel('ca(r=0,z,t=100)')
  subplot(2,2,4)
  plot(z,Tk(3,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=100)'); xlabel('z');
        ylabel('Tk(r=0,z,t=100)')
  figure(2);
  subplot(2,2,1)
  plot(z,ca(4,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=150)'); xlabel('z');
        ylabel('ca(r=0,z,t=150)')
  subplot(2,2,2)
  plot(z,Tk(4,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=150)'); xlabel('z');
        ylabel('Tk(r=0,z,t=150)')
  subplot(2,2,3)
  plot(z,ca(5,:,1)); axis([0 zl 0 0.01]);
  title('ca(r=0,z,t=200)'); xlabel('z');
        ylabel('ca(r=0,z,t=200)')
  subplot(2,2,4)
  plot(z,Tk(5,:,1)); axis([0 zl 305 450]);
  title('Tk(r=0,z,t=200)'); xlabel('z');
        ylabel('Tk(r=0,z,t=200)')
```

8. The radial profiles in c_a and T_k are plotted at $t = 200$ and $z = 5, 35, 70, 100$.

```
%
% Radial profiles (at t = tf)
  for i=1:nz,
  for j=1:nr
    ca_rad(i,j)=ca(5,i,j);
    Tk_rad(i,j)=Tk(5,i,j);
  end
  end
  figure(3);
  subplot(2,2,1)
  plot(r,ca_rad(1,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=5,t=200)'); xlabel('r');
        ylabel('ca(r,z=5,t=200)')
  subplot(2,2,2)
  plot(r,ca_rad(7,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=35,t=200)'); xlabel('r');
        ylabel('ca(r,z=35,t=200)')
  subplot(2,2,3)
  plot(r,ca_rad(14,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=70,t=200)'); xlabel('r');
        ylabel('ca(r,z=70,t=200)')
  subplot(2,2,4)
  plot(r,ca_rad(20,:)); axis([0 r0 0.0 0.01]);
  title('ca(r,z=100,t=200)'); xlabel('r');
        ylabel('ca(r,z=100,t=200)')
  figure(4);
  subplot(2,2,1)
  plot(r,Tk_rad(1,:)); axis([0 r0 305 450]);
  title('Tk(r,z=5,t=200)'); xlabel('r');
        ylabel('Tk(r,z=5,t=200)')
  subplot(2,2,2)
  plot(r,Tk_rad(7,:)); axis([0 r0 305 450]);
  title('Tk(r,z=35,t=200)'); xlabel('r');
        ylabel('Tk(r,z=35,t=200)')
  subplot(2,2,3)
  plot(r,Tk_rad(14,:)); axis([0 r0 305 450]);
  title('Tk(r,z=70,t=200)'); xlabel('r');
        ylabel('Tk(r,z=70,t=200)')
  subplot(2,2,4)
  plot(r,Tk_rad(20,:)); axis([0 r0 305 450]);
  title('Tk(r,z=100,t=200)'); xlabel('r');
        ylabel('Tk(r,z=100,t=200)')
```

The intention of this plotting is to indicate the detailed picture of the PDE solution that is available.

Part of the numerical output is listed in Table 13.2.

Table 13.2. Selected numerical output from `pde_1_main` of Listing 13.1

```
nr =  7   nz = 20

t =  0.0
z =   5.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  10.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  15.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  20.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  25.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  30.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  35.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  40.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  45.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  50.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  55.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  60.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  65.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  70.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  75.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  80.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  85.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  90.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z =  95.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00
z = 100.0   ca(r=0,z,t) =  0.00000   Tk(r=0,z,t) =   305.00

       .                                              .
       .                                              .
       .                                              .

             Output from t = 50, 100, 150 removed
       .                                              .
       .                                              .
       .                                              .

t = 200.0
z =   5.0   ca(r=0,z,t) =  0.00997   Tk(r=0,z,t) =   311.14
z =  10.0   ca(r=0,z,t) =  0.00990   Tk(r=0,z,t) =   320.31
z =  15.0   ca(r=0,z,t) =  0.00977   Tk(r=0,z,t) =   329.97
z =  20.0   ca(r=0,z,t) =  0.00956   Tk(r=0,z,t) =   339.31
z =  25.0   ca(r=0,z,t) =  0.00925   Tk(r=0,z,t) =   348.46
z =  30.0   ca(r=0,z,t) =  0.00882   Tk(r=0,z,t) =   357.94
z =  35.0   ca(r=0,z,t) =  0.00821   Tk(r=0,z,t) =   368.60
z =  40.0   ca(r=0,z,t) =  0.00736   Tk(r=0,z,t) =   381.88
z =  45.0   ca(r=0,z,t) =  0.00611   Tk(r=0,z,t) =   400.19
z =  50.0   ca(r=0,z,t) =  0.00454   Tk(r=0,z,t) =   421.72
z =  55.0   ca(r=0,z,t) =  0.00340   Tk(r=0,z,t) =   432.13
z =  60.0   ca(r=0,z,t) =  0.00283   Tk(r=0,z,t) =   429.28
z =  65.0   ca(r=0,z,t) =  0.00259   Tk(r=0,z,t) =   420.06
z =  70.0   ca(r=0,z,t) =  0.00247   Tk(r=0,z,t) =   409.35
```

```
z =  75.0   ca(r=0,z,t) =  0.00239   Tk(r=0,z,t) =   399.31
z =  80.0   ca(r=0,z,t) =  0.00233   Tk(r=0,z,t) =   390.68
z =  85.0   ca(r=0,z,t) =  0.00228   Tk(r=0,z,t) =   383.55
z =  90.0   ca(r=0,z,t) =  0.00224   Tk(r=0,z,t) =   377.79
z =  95.0   ca(r=0,z,t) =  0.00220   Tk(r=0,z,t) =   373.20
z = 100.0   ca(r=0,z,t) =  0.00217   Tk(r=0,z,t) =   369.56

ncall =   414
```

We can note the following points about this output:

1. ICs (13.6) and (13.10) are verified for `t=0` (with the initial values from Table 13.1 of $c_{a0} = 0$, $T_{k0} = 305$).
2. The solution at `t=200` indicates (a) a reduction in c_a from 0.00997 at $z = \Delta z = 5$ to 0.00217 at $z_l = 100$ due to consumption of the reactant, and (b) an increase in the temperature from 311.14 at $z = \Delta z = 5$ to 369.56 at $z_l = 100$ due to the exothermic reaction; also, the temperature goes through a maximum of 432.13 at $z = 55$ with subsequent cooling (beyond $z = 55$) due to the wall temperature $T_w = 355$). Note also that the first value of `z` for the solution is $z = \Delta z = 5$ since $z = 0$ is not part of the grid in z (where the entering conditions are $c_{ae} = 0.01$, $T_{ke} = 305$ from Table 13.1).

The properties of the solutions are also displayed in Figures 13.2–13.5

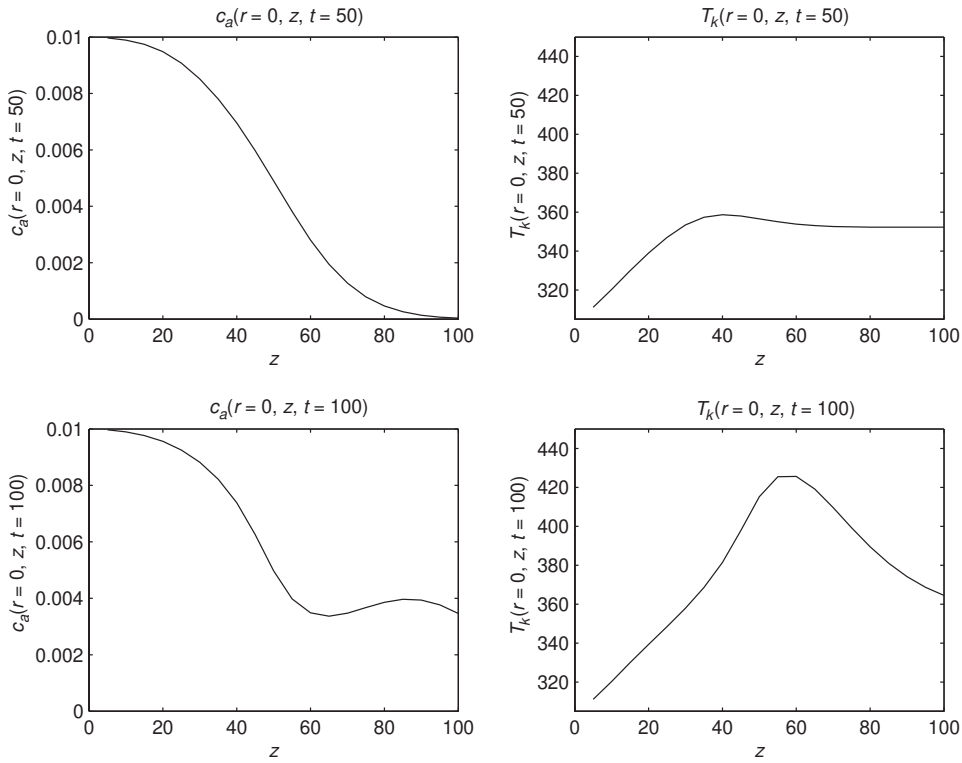

Figure 13.2. Plot of the solution of Eqs. (13.1)–(13.13) for $r = 0$, $0 \le z \le z_l$, $t = 50, 100$

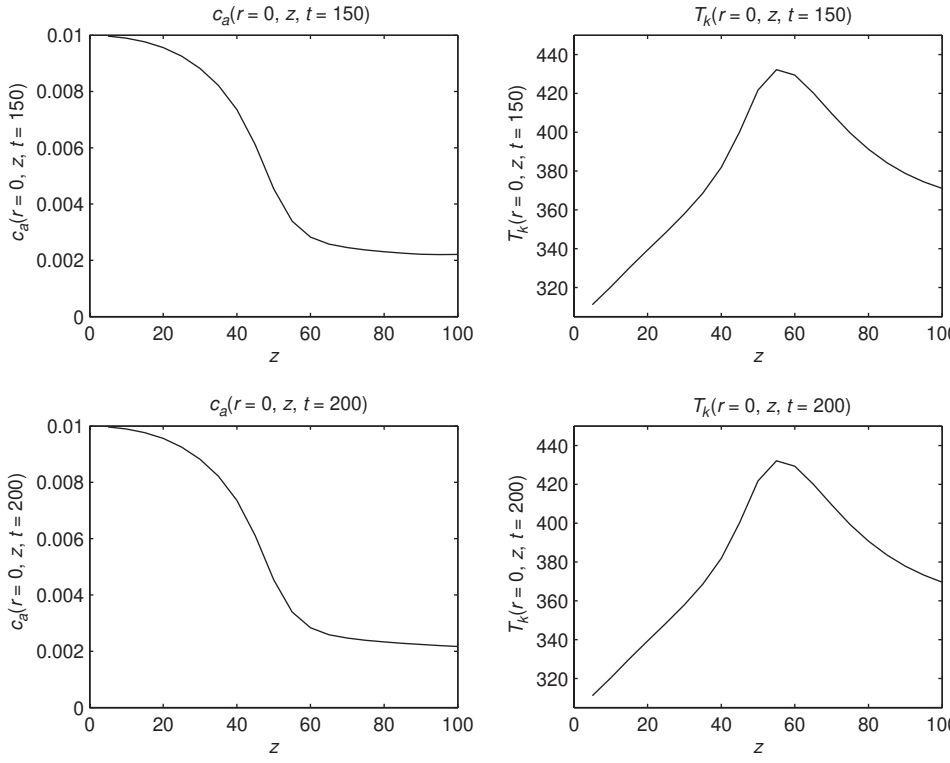

Figure 13.3. Plot of the solution of Eqs. (13.1)–(13.13) for $r = 0$, $0 \leq z \leq z_l$, $t = 150, 200$

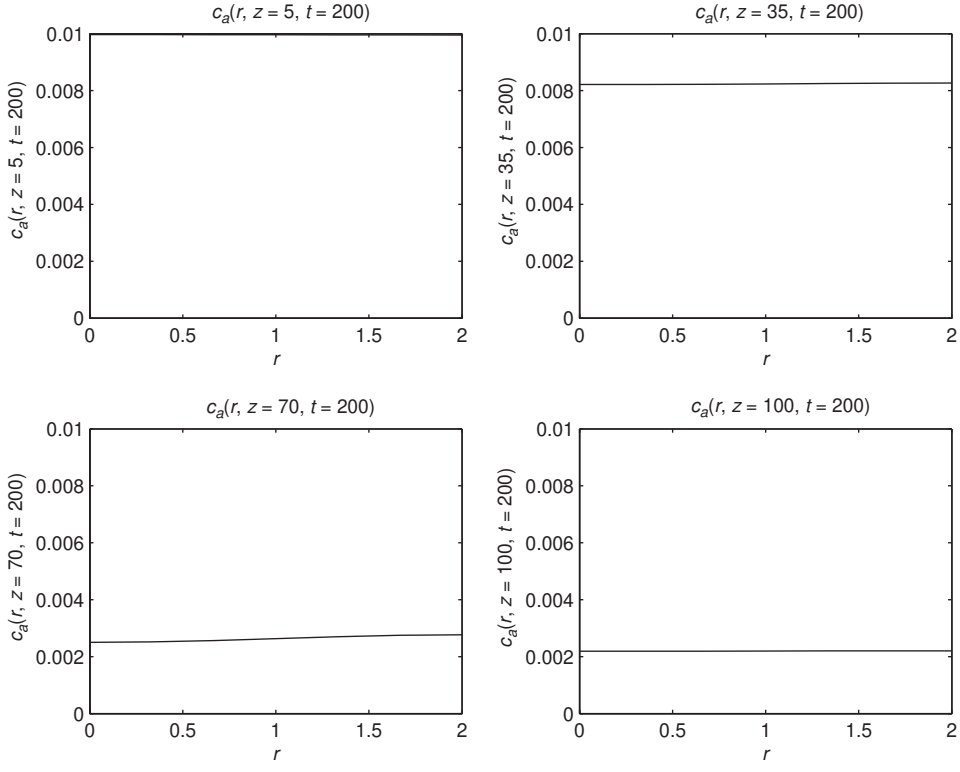

Figure 13.4. Plot of the solution c_a from Eqs. (13.1)–(13.13) for $0 \leq r \leq r_0$, $z = 5, 35, 70, 100$, $t = 200$

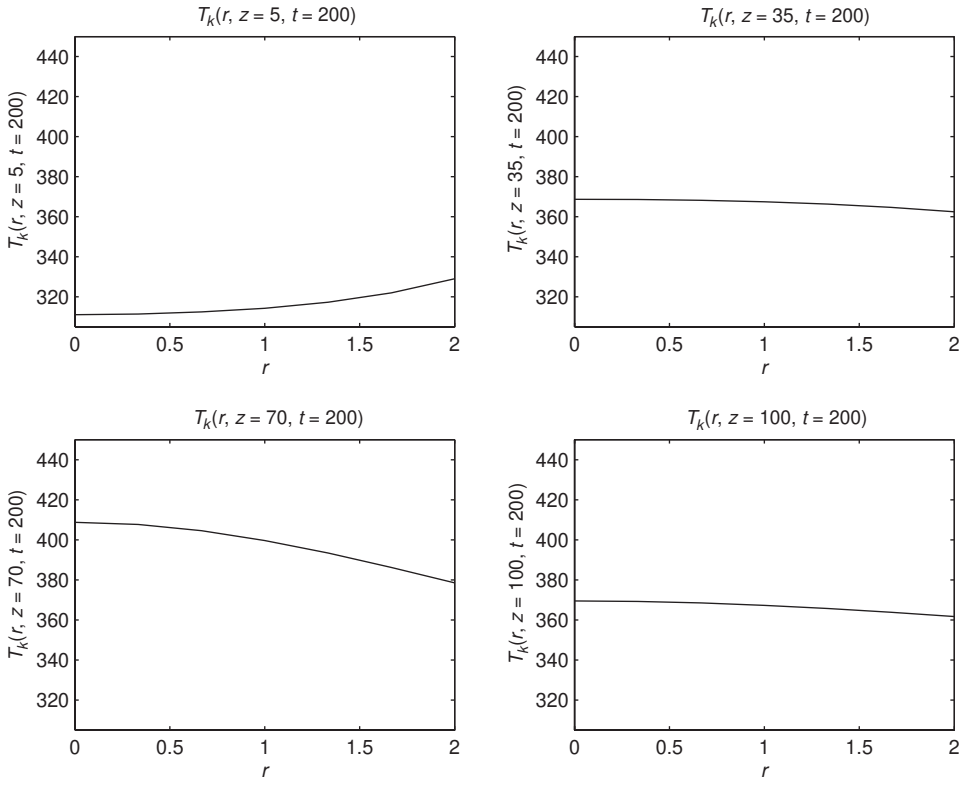

Figure 13.5. Plot of the solution T_k from Eqs. (13.1)–(13.13) for $0 \leq r \leq r_0$, $z = 5, 35, 70, 100$, $t = 200$

We can note the following points about these plots:

1. In Figure 13.2, the c_a and T_k profiles in z change appreciably with t, but in Figure 13.3, they have reached essentially a steady state (no further change with t). This result is also confirmed by the numerical output (although the output for $t = 150$ does not appear in Table 13.2 to save some space).
2. In Figure 13.4, the profiles in r indicate the consumption of the reactant with increasing z as expected; that is the reactor is accomplishing significant conversion of the reactant.
3. In Figure 13.5, the increase in temperature with z is evident; that is, as the reactant flows through the reactor, heat is liberated by the exothermic reaction causing a temperature increase.
4. In Figures 13.4 and 13.5, the radial profiles in c_a and T_k are relatively flat (relative to the case where the resistance to radial heat and mass transfer as reflected in D_c and D_t is greater, i.e., when these coefficients are smaller).
5. Figures 13.6 and 13.7 show clearly the radial variation of concentration and temperature along the axial length of the reactor at final time $t = 200$, when a steady-state profile has been reached. The concentration c_a (Figure 13.6) falls from the inlet value of 0.01 as the reactants flow along the length of the

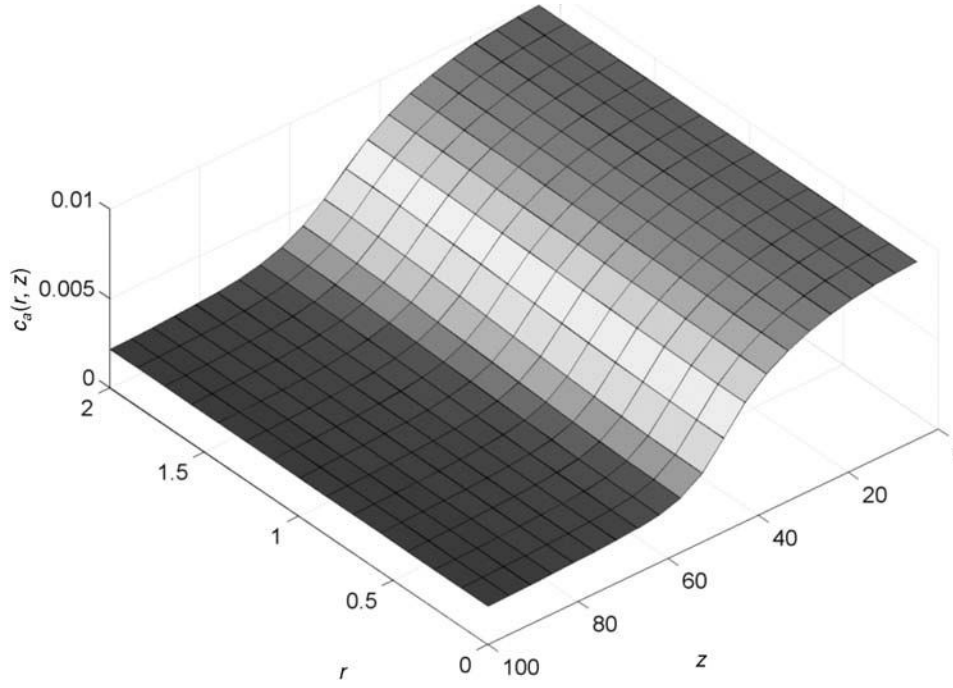

Figure 13.6. Surface plot of reactant concentration $c_a(z, t)$ at $t = 200$

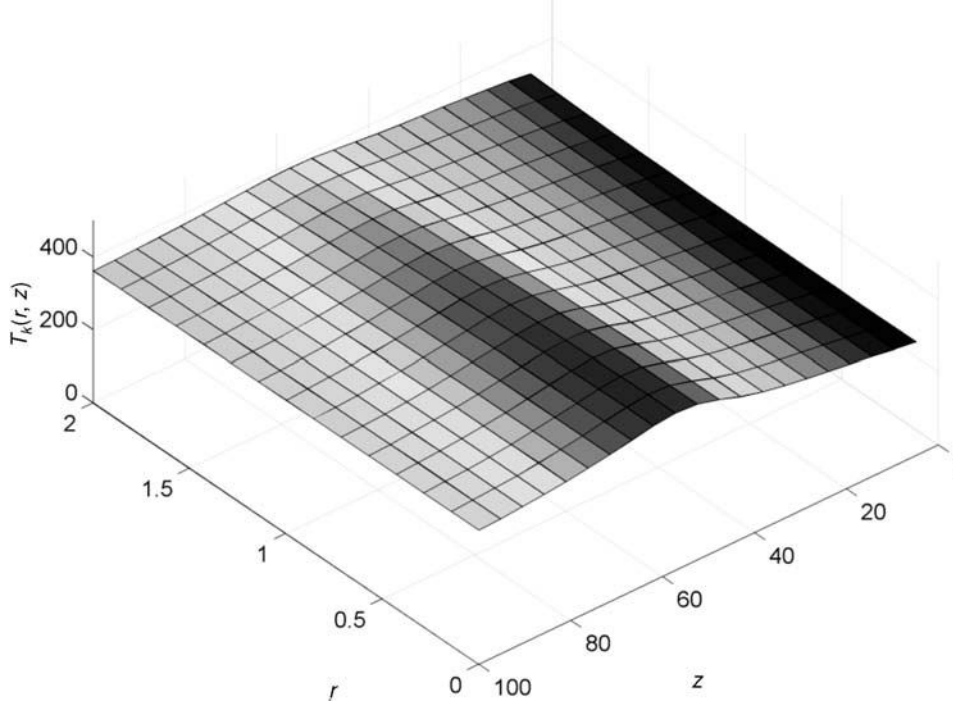

Figure 13.7. Surface plot of reactant temperature $T_k(r, z)$ at $t = 200$

reactor and steadies out to an equilibrium value of 0.00217 at the exit; the radial variation is relatively small. However, the temperature T_k (Figure 13.7) of the reactant rises initially as the reaction proceeds until it reaches a maximum half way along the reactor and then falls back due to the reactant being consumed as it continues its journey toward the reactor exit. We note that the radial temperature variation is significant around the axial midpoint of the reactor, being ≈40 K. This is because the rate at which heat is generated by the exothermic reaction is greater than the rate at which heat is carried away by radial mass and thermal diffusion – see also notes to Figures 13.4 and 13.5 given previously.

We now consider the programming of the ODEs in `pde_1` (see Listing 13.2).

```
  function yt=pde_1(t,y)
%
% Global area
  global     nr     nz     dr     dz    drs    dzs...
              r      z     Dc     Dt     ca     Tk...
            cae    Tke      h      k      E      R...
            rk0      v    rho     Cp     Tw     dH...
          ncall
%
% 1D to 2D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    ca(i,j)=y(ij);
    Tk(i,j)=y(ij+nr*nz);
  end
  end
%
% Step through the grid points in r and z
  for i=1:nz
  for j=1:nr
%
%   (1/r)*car, (1/r)*Tkr
    if(j==1)
      car(i,j)=2.0*(ca(i,j+1)-ca(i,j))/drs;
      Tkr(i,j)=2.0*(Tk(i,j+1)-Tk(i,j))/drs;
    elseif(j==nr)
      car(i,j)=0.0;
      Tkr(i,j)=(1.0/r(j))*(h/k)*(Tw-Tk(i,j));
    else
      car(i,j)=(1.0/r(j))*(ca(i,j+1)-ca(i,j-1))/(2.0*dr);
      Tkr(i,j)=(1.0/r(j))*(Tk(i,j+1)-Tk(i,j-1))/(2.0*dr);
    end
```

```
%
%   carr, Tkrr
    if(j==1)
      carr(i,j)=2.0*(ca(i,j+1)-ca(i,j))/drs;
      Tkrr(i,j)=2.0*(Tk(i,j+1)-Tk(i,j))/drs;
    elseif(j==nr)
      carr(i,j)=2.0*(ca(i,j-1)-ca(i,j))/drs;
      Tkf=Tk(i,j-1)+2.0*dr*h/k*(Tw-Tk(i,j));
      Tkrr(i,j)=(Tkf-2.0*Tk(i,j)+Tk(i,j-1))/drs;
    else
      carr(i,j)=(ca(i,j+1)-2.0*ca(i,j)+ca(i,j-1))/drs;
      Tkrr(i,j)=(Tk(i,j+1)-2.0*Tk(i,j)+Tk(i,j-1))/drs;
    end
%
%   caz, Tkz
    if(i==1)
      caz(i,j)=(ca(i,j)-cae)/dz;
      Tkz(i,j)=(Tk(i,j)-Tke)/dz;
    else
      caz(i,j)=(ca(i,j)-ca(i-1,j))/dz;
      Tkz(i,j)=(Tk(i,j)-Tk(i-1,j))/dz;
    end
%
%   PDEs
    rk=rk0*exp(-E/(R*Tk(i,j)))*ca(i,j)^2;
    cat(i,j)=Dc*(carr(i,j)+car(i,j))-v*caz(i,j)-rk;
    Tkt(i,j)=Dt*(Tkrr(i,j)+Tkr(i,j))-v*Tkz(i,j)...
            -dH/(rho*Cp)*rk;
  end
  end
%
% 2D to 1D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    yt(ij)=cat(i,j);
    yt(ij+nr*nz)=Tkt(i,j);
  end
  end
%
% Transpose and count
  yt=yt';
  ncall=ncall+1;
```

Listing 13.2. Function `pde_1` called by main program `pde_1_main`

We can note the following details about pde_1:

1. After the definition of the function and a *global* area for parameters and variables shared between routines, a 1D to 2D transformation permits programming in terms of the problem-oriented variables ca and Tk.

```
  function yt=pde_1(t,y)
%
% Global area
  global     nr     nz     dr     dz    drs    dzs...
              r      z     Dc     Dt     ca     Tk...
           cae    Tke      h      k      E      R...
           rk0      v    rho     Cp     Tw     dH...
         ncall
%
% 1D to 2D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    ca(i,j)=y(ij);
    Tk(i,j)=y(ij+nr*nz);
  end
  end
```

2. Computation of the nr*nz = 280 MOL ODEs is accomplished with a pair of nested for loops.

```
%
% Step through the grid points in r and z
  for i=1:nz
  for j=1:nr
%
%   (1/r)*car, (1/r)*Tkr
    if(j==1)
      car(i,j)=2.0*(ca(i,j+1)-ca(i,j))/drs;
      Tkr(i,j)=2.0*(Tk(i,j+1)-Tk(i,j))/drs;
    elseif(j==nr)
      car(i,j)=0.0;
      Tkr(i,j)=(1.0/r(j))*(h/k)*(Tw-Tk(i,j));
   else
      car(i,j)=(1.0/r(j))*(ca(i,j+1)-ca(i,j-1))/(2.0*dr);
      Tkr(i,j)=(1.0/r(j))*(Tk(i,j+1)-Tk(i,j-1))/(2.0*dr);
    end
```

The programming of the derivatives (in t) begins with the first-derivative radial groups in Eqs. (13.2) and (13.4). This programming requires some explanation.

(a) At $r = 0$ (j=1), the first-derivative radial group in eq. (13.2) is indeterminant, that is,

$$\frac{1}{r}\frac{\partial c_a}{\partial r} = 0/0$$

as a consequence of *homogeneous Neumann BC* (13.7). We therefore apply *l'Hospital's rule to resolve (regularize)* this indeterminant form.

$$\lim_{r\to 0}\frac{1}{r}\frac{\partial c_a}{\partial r} = \frac{\partial^2 c_a}{\partial r^2}$$

This second derivative can then be approximated as

$$\frac{\partial^2 c_a}{\partial r^2} \approx \frac{c_a(r=\Delta r, z, t) - 2c_a(r=0, z, t) + c_a(r=-\Delta r, z, t)}{\Delta r^2}$$
$$= 2\frac{c_a(r=\Delta r, z, t) - c_a(r=0, z, t)}{\Delta r^2}$$

where we have used BC (13.7) to eliminate the fictitious value $c_a(r=-\Delta r, z, t)$. The corresponding programming is (and also for T_k using BC (13.11))

```
if(j==1)
  car(i,j)=2.0*(ca(i,j+1)-ca(i,j))/drs;
  Tkr(i,j)=2.0*(Tk(i,j+1)-Tk(i,j))/drs;
```

The use of BC (13.7) is as follows. The finite-difference (FD) approximation of BC (13.7) is

$$\frac{\partial c_a(r=0, z, t)}{\partial r} \approx \frac{c_a(r=\Delta r, z, t) - c_a(r=-\Delta r, z, t)}{2\Delta r} = 0$$

Thus, the fictitious value $c_a(r=-\Delta r, z, t)$ is

$$c_a(r=-\Delta r, z, t) = c_a(r=\Delta r, z, t)$$

which is then used in the FD approximation of $\partial^2 c_a/\partial r^2$ (as earlier).

(b) At $r = r_0$ (j=nr), the first-derivative radial group in Eq. (13.2) is programmed as

```
elseif(j==nr)
  car(i,j)=0.0;
```

which reflects homogeneous Neumann BC (13.8).

(c) For T_k at $r = r_0$ (`j=nr`), the first-derivative radial group in Eq. (13.4) is first approximated through the use of *third-type BC (13.12)*. The approximation of this BC is

$$k\frac{\partial T_k(r = r_0, z, t)}{\partial r} \approx k\frac{T_k(r_0 + \Delta r, z, t) - T_k(r_o - \Delta r, z, t)}{2\Delta r} = h(T_w - T_k(r = r_0, z, t))$$

which can be rearranged to give the fictitious value

$$T_k(r_0 + \Delta r, z, t) = T_k(r_0 - \Delta r, z, t) + (2\Delta rh/k)(T_w - T_k(r = r_0, z, t)) \tag{13.14}$$

This result can then be substituted in the approximation of the radial group in T_k in Eq. (13.4).

$$\begin{aligned}
&\frac{1}{r_0}\frac{\partial T_k}{\partial r} \\
&\approx \frac{1}{r_0}\frac{T_k(r_0 + \Delta r, z, t) - T_k(r_0 - \Delta r, z, t)}{2\Delta r} \\
&= \frac{1}{r_0}\frac{T_k(r_0 - \Delta r, z, t) + (2\Delta rh/k)(T_w - T_k(r = r_0, z, t)) - T_k(r_0 - \Delta r, z, t)}{2\Delta r} \\
&= \frac{1}{r_0}(h/k)(T_w - T_k(r = r_0, z, t))
\end{aligned}$$

The corresponding programming is (with `j=nr`)

```
Tkr(i,j)=(1.0/r(j))*(h/k)*(Tw-Tk(i,j));
```

For $r \neq 0, r_0$, the approximation of the first-derivative radial group in Eq. (13.2) is

$$\frac{1}{r}\frac{\partial T_k}{\partial r} \approx \frac{1}{r}\frac{\partial T_k(r + \Delta r, z, t) - T_k(r - \Delta r, z, t)}{2\Delta r}$$

This approximation is programmed as (and also, for c_a)

```
else
    car(i,j)=(1.0/r(j))*(ca(i,j+1)-ca(i,j-1))/(2.0*dr);
    Tkr(i,j)=(1.0/r(j))*(Tk(i,j+1)-Tk(i,j-1))/(2.0*dr);
```

The programming of the second derivative in Eqs. (13.2) and (13.4)

```
%
%   carr, Tkrr
    if(j==1)
      carr(i,j)=2.0*(ca(i,j+1)-ca(i,j))/drs;
      Tkrr(i,j)=2.0*(Tk(i,j+1)-Tk(i,j))/drs;
    elseif(j==nr)
      carr(i,j)=2.0*(ca(i,j-1)-ca(i,j))/drs;
      Tkf=Tk(i,j-1)+2.0*dr*h/k*(Tw-Tk(i,j));
      Tkrr(i,j)=(Tkf-2.0*Tk(i,j)+Tk(i,j-1))/drs;
    else
      carr(i,j)=(ca(i,j+1)-2.0*ca(i,j)+ca(i,j-1))/drs;
      Tkrr(i,j)=(Tk(i,j+1)-2.0*Tk(i,j)+Tk(i,j-1))/drs;
    end
```

also requires some elaboration.

(a) At $r = 0$, BCs (13.7) and (13.11) apply. Thus the approximation of the second derivative for c_a becomes (with Eq. (13.7))

$$\frac{\partial^2 c_a(r=0,z,t)}{\partial r^2} = \frac{c_a(r=\Delta r,z,t) - 2c_a(r=0,z,t) + c_a(r=-\Delta r,z,t)}{\Delta r^2}$$
$$= 2\frac{c_a(r=\Delta r,z,t) - c_a(r=0,z,t)}{\Delta r^2}$$

This approximation and a similar one for T_k are coded as

```
if(j==1)
  carr(i,j)=2.0*(ca(i,j+1)-ca(i,j))/drs;
  Tkrr(i,j)=2.0*(Tk(i,j+1)-Tk(i,j))/drs;
```

(b) For $r = r_0$, a similar argument for the second derivative of c_a in r gives

$$\frac{\partial^2 c_a(r=r_0,z,t)}{\partial r^2} = 2\frac{c_a(r=r_0-\Delta r,z,t) - c_a(r=r_0,z,t)}{\Delta r^2}$$

which is coded as

```
elseif(j==nr)
  carr(i,j)=2.0*(ca(i,j-1)-ca(i,j))/drs;
```

(c) For the second derivative of T_k at $r = r_0$, we can use the fictitious value of $T_k(r_0 + \Delta r, z, t)$ from Eq. (13.14) that includes BC (13.12). This value can then be substituted in the usual approximation of a second derivative

$$\frac{\partial^2 T_k}{\partial r^2} \approx \frac{\partial T_k(r_0+\Delta r,z,t) - 2T_k(r_0,z,t) + T_k(r_0-\Delta r,z,t)}{2\Delta r}$$

The coding for these equations is (with `j=nr`)

```
Tkf=Tk(i,j-1)+2.0*dr*h/k*(Tw-Tk(i,j));
Tkrr(i,j)=(Tkf-2.0*Tk(i,j)+Tk(i,j-1))/drs;
```

(d) For $r \neq 0, r_0$, the usual approximation of a second derivative can be used without concern for BCs. The coding is

```
else
  carr(i,j)=(ca(i,j+1)-2.0*ca(i,j)+ca(i,j-1))/drs;
  Tkrr(i,j)=(Tk(i,j+1)-2.0*Tk(i,j)+Tk(i,j-1))/drs;
```

3. The coding for the derivatives in z in Eqs. (13.2) and (13.4), subject to BCs (13.9) and (13.13), is straightforward.

```
%
%   caz, Tkz
    if(i==1)
      caz(i,j)=(ca(i,j)-cae)/dz;
      Tkz(i,j)=(Tk(i,j)-Tke)/dz;
    else
      caz(i,j)=(ca(i,j)-ca(i-1,j))/dz;
      Tkz(i,j)=(Tk(i,j)-Tk(i-1,j))/dz;
    end
```

The only special requirement is to use the entering values of c_a and T_k at the first grid point (`i=1`), which, again, is for $z = \Delta z$. These entering values are `cae` and `Tke`; they do not require ODEs (since they are specified as parameters), and therefore the gridding does not have to start at $z = 0$. In other words, including $z = 0$ would require trivial ODEs for the condition $(\partial c_a(r, z = 0, t))/\partial t = (\partial T_k(r, z = 0, t))/\partial t = 0$; there would be $(2)(7) = 14$ such ODEs for the 2 PDEs (Eqs. (13.2) and (13.4)) each approximated over 7 points in r.

Note that the first-order derivatives in z in Eqs. (13.2) and (13.4) are approximated with *two-point upwind FDs*. This name comes from the fact that these derivatives represent convection, and we therefore use values of the dependent variables that are "upwind" (point `i-1`) of the point of the approximation (point `i`). We will not discuss the reasons for this choice of an FD other than to indicate that *some form of upwinding is generally required for convective terms to account for the influence of the flow*.

Also, these two-point upwind approximations have the significant limitation of being only first-order correct (the principal truncation error term is $O(\Delta z)$). Thus, a higher-order FD approximation or nonlinear approximations for the convective derivatives in PDEs could lead to a significant increase in the accuracy of the numerical solution. However, a discussion of the numerical integration of convective (hyperbolic) PDEs that propagate moving fronts is not included in this discussion. Numerical methods for the resolution of discontinuities and moving fronts are discussed extensively in the literature, for example, in references [2–4] and are implemented in available computer codes.

This completes the coding for all of the spatial derivatives in Eqs. (13.2) and (13.4). All that remains to program these equations is to put the derivatives together along with the reaction term. The coding for this is

```
%
%   PDEs
    rk=rk0*exp(-E/(R*Tk(i,j)))*ca(i,j)^2;
    cat(i,j)=Dc*(carr(i,j)+car(i,j))-v*caz(i,j)-rk;
    Tkt(i,j)=Dt*(Tkrr(i,j)+Tkr(i,j))-v*Tkz(i,j)...
            -dH/(rho*Cp)*rk;
  end
  end
```

The double `end` concludes the pair of nested `for` loops at the beginning that step the calculations through all 280 points (i.e., 280 ODEs).

4. Finally, there is a 2D to 1D matrix conversion to put all of the derivatives in t (`cat(i,j), Tkt(i,j)`) in a single vector (`yt`), which is then transposed as required by `ode15s`.

```
%
% 2D to 1D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    yt(ij)=cat(i,j);
    yt(ij+nr*nz)=Tkt(i,j);
  end
  end
%
% Transpose and count
  yt=yt';
  ncall=ncall+1;
```

The counter `ncall` displayed at the end of the solution has the value 414 (see Table 13.2), indicating a rather modest computational effort in computing the

MOL solution; in other words, integrator `ode15s` handled this problem with quite acceptable efficiency.

`pde_1` in Listing 13.2 is based on the explicit programming of the FDs for the MOL ODEs. We could, however, also use library routines for this purpose. To illustrate this idea, we consider an alternative ODE routine as given in Listing 13.3.

```
  function yt=pde_2(t,y)
%
% Global area
  global     nr     nz     dr     dz    drs    dzs...
              r     r0      z     zl     Dc     Dt...
             ca     Tk    cae    Tke      h      k...
              E      R    rk0      v    rho     Cp...
             Tw     dH  ncall
%
% 1D to 2D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    ca(i,j)=y(ij);
    Tk(i,j)=y(ij+nr*nz);
  end
  end
%
% Step through the grid points in r and z
  for i=1:nz
    ca1d=ca(i,:);
    Tk1d=Tk(i,:);
%
%   car, Tkr
    car1d=dss004(0.0,r0,nr,ca1d);
    car(i,:)=car1d;
    car(i,1)= 0.0;
    car(i,nr)=0.0;
    Tkr1d=dss004(0.0,r0,nr,Tk1d);
    Tkr(i,:)=Tkr1d;
    Tkr(i,1)= 0.0;
    Tkr(i,nr)=(h/k)*(Tw-Tk(i,nr));
%
%   carr, Tkrr
    car1d( 1)=0.0;
    car1d(nr)=0.0;
    nl=2;
    nu=2;
    carr1d=dss044(0.0,r0,nr,ca1d,car1d,nl,nu);
    carr(i,:)=carr1d;
    Tkr1d( 1)=0.0;
```

```
    Tkr1d(nr)=(h/k)*(Tw-Tk1d(nr));
    nl=2;
    nu=2;
    Tkrr1d=dss044(0.0,r0,nr,Tk1d,Tkr1d,nl,nu);
    Tkrr(i,:)=Tkrr1d;
  for j=1:nr
%
%   (1/r)*car, (1/r)*Tkr
    if(j~=1)
      car(i,j)=(1.0/r(j))*car(i,j);
      Tkr(i,j)=(1.0/r(j))*Tkr(i,j);
    end
    if(j==1)carr(i,j)=2.0*carr(i,j);end
    if(j==1)Tkrr(i,j)=2.0*Tkrr(i,j);end
%
%   caz, Tkz
    if(i==1)
      caz(i,j)=(ca(i,j)-cae)/dz;
      Tkz(i,j)=(Tk(i,j)-Tke)/dz;
    else
      caz(i,j)=(ca(i,j)-ca(i-1,j))/dz;
      Tkz(i,j)=(Tk(i,j)-Tk(i-1,j))/dz;
    end
%
%   PDEs
    rk=rk0*exp(-E/(R*Tk(i,j)))*ca(i,j)^2;
    cat(i,j)=Dc*(carr(i,j)+car(i,j))-v*caz(i,j)-rk;
    Tkt(i,j)=Dt*(Tkrr(i,j)+Tkr(i,j))-v*Tkz(i,j)...
            -dH/(rho*Cp)*rk;
  end
  end
%
% 2D to 1D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    yt(ij)=cat(i,j);
    yt(ij+nr*nz)=Tkt(i,j);
  end
  end
%
% Transpose and count
  yt=yt';
  ncall=ncall+1;
```

Listing 13.3. Function `pde_2` called by main program `pde_1_main`

We can note the following details about pde_2:

1. As in pde_1 in Listing 13.2, the function and a global area are defined, followed by a 1D to 2D matrix transformation.

```
  function yt=pde_1(t,y)
%
% Global area
  global     nr     nz     dr     dz    drs    dzs...
              r     r0      z     zl     Dc     Dt...
             ca     Tk    cae    Tke      h      k...
              E      R    rk0      v    rho     Cp...
             Tw     dH  ncall
%
% 1D to 2D matrices
  for i=1:nz
  for j=1:nr
    ij=(i-1)*nr+j;
    ca(i,j)=y(ij);
    Tk(i,j)=y(ij+nr*nz);
  end
  end
```

2. A for loop is used to step through z. Since the spatial differentiation dss routines compute numerical derivatives of 1D array (vector), a conversion from 2D to 1D is first required at each value of z.

```
%
% Step through the grid points in r and z
  for i=1:nz
    ca1d=ca(i,:);
    Tk1d=Tk(i,:);
```

3. Then the first-order radial derivatives can be computed by a call to dss004 ($\partial c_a/\partial r =$ car1d, $\partial T_k/\partial r =$ Tkr1d).

```
%
%   car, Tkr
    car1d=dss004(0.0,r0,nr,ca1d);
    car(i,:)=car1d;
    car(i,1)= 0.0;
    car(i,nr)=0.0;
```

```
Tkr1d=dss004(0.0,r0,nr,Tk1d);
Tkr(i,:)=Tkr1d;
Tkr(i,1)= 0.0;
Tkr(i,nr)=(h/k)*(Tw-Tk(i,nr));
```

Note the conversion back to 2D arrays and the use of BCs (13.7) and (13.8) (for c_a) and (13.11) and (13.12) (for T_k). Thus, through the use of the compact Matlab indexing (subscripting), the calculation of 2D spatial derivatives in r through the use of 1D routines (`dss004`) is facilitated.

4. Similarly, the second derivatives in r can be computed by calls to `dss044`. Note again the use of BCs (13.7), (13.8), (13.11), and (13.12) that are all declared Neumann (through `nl = nu = 2`) and the conversion from 2D to 1D before the calls to `dss044` and the 1D to 2D conversions after the calls to `dss044`.

```
%
%   carr, Tkrr
    car1d( 1)=0.0;
    car1d(nr)=0.0;
    nl=2;
    nu=2;
    carr1d=dss044(0.0,r0,nr,ca1d,car1d,nl,nu);
    carr(i,:)=carr1d;
    Tkr1d( 1)=0.0;
    Tkr1d(nr)=(h/k)*(Tw-Tk1d(nr));
    nl=2;
    nu=2;
    Tkrr1d=dss044(0.0,r0,nr,Tk1d,Tkr1d,nl,nu);
    Tkrr(i,:)=Tkrr1d;
```

5. To this point, the variable coefficient $1/r$ in the terms $(1/r)\,(\partial c_a/\partial r)$ and $(1/r)\,(\partial T_k/\partial r)$ has not been included. The following code accomplishes this (the complication is due to the singular point $r = 0$):

```
%
%   (1/r)*car, (1/r)*Tkr
    if(j~=1)
      car(i,j)=(1.0/r(j))*car(i,j);
      Tkr(i,j)=(1.0/r(j))*Tkr(i,j);
    end
    if(j==1)carr(i,j)=2.0*carr(i,j);end
    if(j==1)Tkrr(i,j)=2.0*Tkrr(i,j);end
```

For $r = 0$ (j=1), the indeterminate forms reduce to the second derivative as discussed previously; that is,

$$\lim_{r\to 0} \frac{1}{r}\frac{\partial c_a}{\partial r} = \frac{\partial^2 c_a}{\partial r^2}$$

$$\lim_{r\to 0} \frac{1}{r}\frac{\partial T_k}{\partial r} = \frac{\partial^2 T_k}{\partial r^2}$$

which are then combined with the preexisting second derivatives to give twice the second derivative (only at $r = 0$). Otherwise ($r \neq 0$), the variable coefficient $1/r$ can be programmed directly.

6. The first-order convective derivatives in z are approximated by the two-point upwind FDs discussed previously.

```
%
%   caz, Tkz
    if(i==1)
      caz(i,j)=(ca(i,j)-cae)/dz;
      Tkz(i,j)=(Tk(i,j)-Tke)/dz;
    else
      caz(i,j)=(ca(i,j)-ca(i-1,j))/dz;
      Tkz(i,j)=(Tk(i,j)-Tk(i-1,j))/dz;
    end
```

7. All of the derivatives in Eqs. (13.2) and (13.4) are now computed, so the PDEs can be programmed.

```
%
%   PDEs
    rk=rk0*exp(-E/(R*Tk(i,j)))*ca(i,j)^2;
    cat(i,j)=Dc*(carr(i,j)+car(i,j))-v*caz(i,j)-rk;
    Tkt(i,j)=Dt*(Tkrr(i,j)+Tkr(i,j))-v*Tkz(i,j)...
            -dH/(rho*Cp)*rk;
```

The resemblance of this coding to the PDEs, Eqs. (13.2) and (13.4), including Eq. (13.5), is one of the salient features of MOL analysis.

The call to pde_2 through ode15s is accomplished with the following code (mf = 2) in pde_1_main:

```
%
% ODE integration
  reltol=1.0e-04; abstol=1.0e-04;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
```

```
mf=2;
if(mf==1)[t,y]=ode15s(@pde_1,tout,y0,options);end
if(mf==2)[t,y]=ode15s(@pde_2,tout,y0,options);end
```

The output from `pde_1_main` and `pde_2` is very similar to the output from `pde_1_main` and `pde_1`, so we will not review it here. The concluding points we make are (1) library routines (e.g., `dss004, dss044`) can be used to calculate the spatial derivatives in PDEs, and (2) since calls to these routines can easily be changed (in the case of the `dss` routines, only the number changes, while the arguments remain the same), the effect of variation in the FD order is easily investigated (e.g., switching from `dss004, dss044` to `dss006, dss046`); in other words, we can easily investigate *p*-refinement as well as *h*-refinement (in which the number of grid points is changed).

We conclude this chapter with a few additional points (but it is not our intention to discuss here the design and control of a chemical reactor):

- Since in the analysis of a chemical reactor we would be particularly interested in the exiting conditions (at $z = z_l$), we could focus on $c_a(r, z = z_l, t)$ and $T(r, z = z_l, t)$ as the output to be listed and plotted (this output is included in Figures 13.4 and 13.5). Also, there may be significant radial variation at the exit $z = z_l$, so we could also compute the integrals

$$c_{a,avg}(t) = \int_0^{r_0} 2\pi r c_a(r, z_l, t) dr/(\pi r_0^2)$$

$$T_{k,avg}(t) = \int_0^{r_0} 2\pi r T_k(r, z_l, t) dr/(\pi r_0^2)$$

  to give the average exiting concentration and temperature $c_{a,avg}(t)$ and $T_{k,avg}(t)$. These integrals could be evaluated numerically using a well-established *quadrature method* such as *Simpson's rule*. In the case of Figures 13.4 and 13.5, the radial variation in $c_a(r, z = z_l, t)$ and $T_k(r, z = z_l, t)$ is not very great, but this could change substantially for a more highly exothermic reaction or an internal structure that increases the resistance to radial heat and mass transfer. These conditions could in turn lead to adverse higher peak temperatures.
- An important safety concern in the operation of an exothermic reactor may be the maximum internal temperature. For example, an excessive temperature might damage the reactor or its contents, or possibly even cause an explosion. This analysis could be used to predict the maximum temperature, which would occur along the centerline $r = 0$. A controller could then be added to the model that would reduce the wall temperature T_w as a function of the maximum temperature to cool the reactor as needed to limit the maximum temperature.
- Interesting parametric studies can be performed with the preceding code. For example, $T_w = 355$ was selected in order to start the chemical reaction. If $T_w = 305$ is used instead (corresponding to the initial and entering temperatures), the reaction does not take place so that the concentration for $0 \le z \le z_l$ eventually

reaches the entering value $c_{ae} = 0.01$ (starting at $c_{a0} = 0$), and the temperature for $0 \leq z \leq z_l$ essentially remains at the initial and entering values of 305.

- This model is rather basic in the sense that only one reactant is considered. More realistically, multiple reactants and products would be of interest and a mass balance would be required for each such component. Thus, additional PDEs would be required, but the procedure of the preceding analysis could be directly extended for a more complex model (with more PDEs).
- The name *convection–diffusion–reaction* system could also be given a more mathematical designation, for example, *hyperbolic–parabolic* system where the convective terms (involving the velocity v) are hyperbolic and the diffusion terms (involving the diffusivities D_c, D_t) are parabolic.
- Actually, the term *diffusivity* may not always be appropriate (particularly for a high-Reynolds-number situation) in the sense that the radial dispersion in a reactor or similar physical system is probably not due solely to molecular diffusion (which is generally a small effect) as much as stream splitting and flow around internal obstructions; this effect is also known as *eddy diffusion*. Thus the term *dispersion coefficient* might be more appropriate. Also, dispersion coefficients are frequently measured experimentally since they typically reflect the effects of complex internal flow patterns.
- The accuracy of the numerical solution can be investigated by changing the number of grid points in r and z and observing the effect on the output. For example, doubling `nr` has little effect on the solution, implying that `nr=7` is an adequate number of grid points in r. This *h-refinement* in which the grid spacing is changed is an important method for assessing the solution accuracy (*spatial convergence in r and z*) when an analytical solution is not available (usually the case in applications). But keep in mind that the total number of ODEs is `2*nr*nz` that increases rapidly with increasing *nr* and *nz*, so this practical limitation must be considered when changing the number of grid points.
- The accuracy of the integration in t could also be assessed by changing the error tolerances `abstol` and `reltol` before the call to `ode15s` and observing the effect on the solution. This check on the *temporal convergence* is also an important method for assessing the solution accuracy.
- In summary, an error analysis in both time and space is an important part of the numerical solution of a PDE model.

APPENDIX A

A.1. Units Check for Eqs. (13.1)–(13.13)

We detail here a check on the units of each term in the formulation of the PDE model (Eqs. (13.1)–(13.13)). This is an essential step in the formulation of a mathematical model for at least two reasons: (a) consistency of units throughout an equation is necessary for the equation to be correct, and (b) the units give an insight into the physical (also, chemical, and biological) meaning of each term of the equation and therefore an overall perspective of the applicability and usefulness of the equation.

Here, we tabulate the terms and associated units that are the starting point for Eqs. (13.1)–(13.13). Starting with the terms in the mass balance (leading to Eq. (13.2)), we have

Term	Units
$2\pi r\Delta r\Delta z\dfrac{\partial c_a}{\partial t}$	(cm)(cm)(cm)(gmol/cm^3)(1/s) = gmol/s
$2\pi r\Delta z q_m\|_r - 2\pi(r+\Delta r)\Delta z q_m\|_{r+\Delta r}$	(cm)(cm)(gmol/(cm$^2\cdot$ s)) = gmol/s
$2\pi r\Delta r v c_a\|_{z-\Delta z} - 2\pi r\Delta r v c_a\|_z$	(cm)(cm)(cm/s)(gmol/cm^3) = gmol/s
$-2\pi r\Delta r\Delta z k_r c_a^2$	(cm)(cm)(cm)(cm^3/(gmol·s)(gmol/cm^3)^2 = gmol/s

We see that each term has the net units of gmol/s. The physical interpretation of these units is (a) net accumulation or depletion (depending on whether the derivative in t is positive or negative) of the reactant in the incremental volume with dimensions Δr and Δz, (b) net diffusion rate into or out of the incremental volume, (c) net convection rate into or out of the incremental volume, and (d) rate of consumption of the reactant in the incremental volume (consumption because of the minus sign in $-2\pi\ldots$). Thus, we have ensured that the units are consistent throughout the equation and in the process, gained an understanding of the physical significance of each term. Also, we have used *cgs units*. Any other choice would be fine, for example, *mks or SI* (as long as they are metric!).

We now proceed with the units analysis of the remaining equations (Eqs. (13.1)–(13.13)).

Term	Units
$q_m = -D_c\dfrac{\partial c_a}{\partial r}$	(cm^2/s)(gmol/cm^3)(1/cm) = gmol/(cm$^2\cdot$ s)

This unit analysis of Eq. (13.1) is for *Fick's first law* that provides a mass diffusion flux, q_m; "flux" means a transfer rate *per unit area*.

Term	Units
$2\pi r\Delta r\Delta z\rho C_p\dfrac{\partial T_k}{\partial t}$	(cm)(cm)(cm)(g/cm^3)(cal/(g·K))(K/s) = cal/s
$2\pi r\Delta z q_h\|_r - 2\pi(r+\Delta r)\Delta z q_h\|_{r+\Delta r}$	(cm)(cm)(cal/(cm$^2\cdot$ s) = cal/s
$2\pi r\Delta r v\rho C_p T_k\|_{z-\Delta z} - 2\pi r\Delta r v\rho C_p T_k\|_z$	(cm)(cm)(cm/s)(g/cm^3)(cal/(g·K)(K) = cal/s
$-\Delta H 2\pi r\Delta r\Delta z k_r c_a^2$	(cal/gmol)(cm)(cm)(cm)(cm^3/(gmol·s)(gmol/cm^3)^2 = cal/s

The net units are the cal/s accumulating, diffusing, flowing, and being generated (by the exothermic reaction) in the incremental value with dimensions Δr and Δz. The heat of reaction, ΔH, by convention, is negative for an exothermic reaction so that the reaction term is positive (due to the minus in $-\Delta H$); see the numerical value of ΔH in Table 13.1.

Term	Units
$q_h = -k\dfrac{\partial T_k}{\partial r}$	((cal·cm)/(s·cm^2·K))(K/cm) = cal/(s·cm^2)

This unit analysis of Eq. (13.3) is for *Fourier's first law* that provides a thermal conduction flux, q_h.

Term Units

$$k_r = k_0\, e^{-E/(RT_k)} \quad (\text{cm}^3/(\text{gmol}\cdot\text{s})) \left[\exp \frac{\text{cal/gmol}}{(\text{cal}/(\text{gmol}\cdot\text{K}))(\text{K})} \right] = \text{cm}^3/(\text{gmol}\cdot\text{s})$$

Note that the argument of the exponential function (from Eq. (13.5)) is dimensionless. Equations (13.6)–(13.11) are essentially self-explanatory with respect to units.

Equation (13.12) has the following units:

Term Units

$$k\frac{\partial T_k(r = r_0, z, t)}{\partial r} = h(T_w - T_k(r = r_0, z, t)) \quad \text{(LHS) } ((\text{cal}\cdot\text{cm})/\text{s}\cdot\text{cm}^2\cdot\text{K}))(\text{K/cm}) = \text{cal}/(\text{s}\cdot\text{cm}^2)$$

$$\text{(RHS) } (\text{cal}/(\text{s}\cdot\text{cm}^2\cdot\text{K}))(\text{K}) = \text{cal}/(\text{s}\cdot\text{cm}^2)$$

Also, Eq. (13.13) is self-explanatory.

REFERENCES

[1] Rosner, D. E. (1986), *Transport Processes in Chemically Reacting Flow Systems*, Butterworth-Heinemann, Berlin

[2] Leveque, R. J. (2002), *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge

[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] Wesseling, P. (2001), *Principles of Computational Fluid Dynamics*, Springer, Berlin