Partial Differential Equations

W. E. Schiesser
Iacocca D307
111 Research Drive
Lehigh University
Bethlehem, PA 18015

(610) 758-4264 (office)
(610) 758-5057 (fax)

wes1@lehigh.edu

http://www.lehigh.edu/~ wes1/wes1.html

Partial Differential Equations

http://www.lehigh.edu/~ wes1/apci/28apr00.tex
http://www.lehigh.edu/~ wes1/apci/28apr00.ps
http://www.lehigh.edu/~ wes1/apci/28apr00.pdf
files.txt

1. Geometric and related classifications
2. Time and spatial discretization
3. Method of Lines
4. Finite differences
5. Finite elements
6. Finite volumes
7. Weighted residuals
8. Adaptive grids
9. References and software

Why differential equations?

Differential equations are possibly the most widely used form of mathematics in science and engineering.

What are differential equations?

Differential equations are equations with one or more derivatives.

What are partial differential equations (PDEs)?

PDEs are differential equations with two or more independent variables, typically time and one or more spatial dimensions.

Why solve PDEs using computers?

Essentially all realistic problems in differential equations are:

- High order (of order $n \times n$)
  - $10^0 \times 10^0$ systems are now routine
  - $10^1 \times 10^1$ to $10^2 \times 10^2$ systems are now rather commonplace
  - $10^3 \times 10^3$ systems are at the forefront of applications
- Nonlinear

We can solve differential equations (PDEs) analytically only if $n \leq 2$, and they are linear.

Representative analytical methods include separation of variables, Laplace transforms

What is the origin of such high order, nonlinear problems?

Examples: Solid state device simulation; dynamics of chemical reactors, separation systems

PDEs - (more than one independent variable, e.g., x, t)

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}; \text{solution: } u(x,t)$$

The solution is the dependent variable(s) as a function of the independent variables (that satisfies the PDE(s) and all of its auxiliary conditions).

PDEs are frequently written with subscript notation, e.g.,

$$\frac{\partial u}{\partial t} \Leftrightarrow u_t, \frac{\partial^2 u}{\partial x^2} \Leftrightarrow u_{xx}$$

so the PDE can be written in subscript notation as

$$u_t = u_{xx}$$

This notation makes the statement of PDEs more compact, and is also useful in their coding, e.g., for an arrays, we can use

$$u \Leftrightarrow u(i), u_t, \Leftrightarrow ut(i), u_{xx} \Leftrightarrow uxx(i)$$

In general, we will obtain the solution in numerical form, e.g., $u(x,t)$ will be tabulated (or plotted) values of $u(x,t)$ vs x and t..

Geometric and Related Classifications

The geometric classification of PDEs as *elliptic, hyperbolic and parabolic* has a rigorous definition for a single linear, second order PDE. Here we introduce a geometric classification that is less rigorous, but more general.

Order in z (BV)	Order in t (IV)	Classification	Example
1	1	First order hyperbolic	$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z}$ (advection equation)
2	2	Second order hyperbolic	$\frac{\partial^2 u}{\partial t^2} = c^2\frac{\partial^2 u}{\partial z^2}$ (wave equation)
2	1	Parabolic	$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial z^2}$ (Fourier's or Fick's second law)
2 (in y, z)	0	Elliptic	$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0$ (Laplace's equation)

Combinations of these classes of PDEs are also possible. For example

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z} + D\frac{\partial^2 u}{\partial z^2} \tag{1}$$

is a *hyperbolic-parabolic (or convective diffusion)* equation which could have the auxiliary conditions

$$u(z,0) = f(z) \tag{2}$$

$$-D\frac{\partial u(0,t)}{\partial z} = v(u_0 - u(0,t)) \tag{3}$$

$$\frac{\partial u(L,t)}{\partial z} = 0 \tag{4}$$

BCs (3) and (4) (note there are two since eq. (1) is second order in z) are the **Wilhelm Danckwerts** BCs (named after two well known ChEs); generally, eq. (4) is not appropriate for unsteady state problems, and can be replaced by

$$\frac{\partial u(L,t)}{\partial t} = -v\frac{\partial u(L,t)}{\partial z} \tag{5}$$

Note that eq. (5) is an acceptable BC for eq. (1) since it is only first order in z, while eq. (1) is second order in z (generally, a BC is at least one order lower than the highest order derivative in the PDE).

To repeat, we have the basic question, what do we mean by the solution to a PDE? The answer in general is *the dependent variable as a function of the independent variables.* Thus, for equations (1) to (4), the solution is a function, $u(z,t)$, in analytical or numerical form.

Eq. (1) can be extended in **cylindrical coordinates** to include a radial coordinate, r

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z} + D\left(\frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r}\right) \tag{6}$$

Since eq. (6) is second order in r, it requires two BCs, e.g.,

$$\frac{\partial u(0,z,t)}{\partial r} = 0 \tag{7}$$

Eq. (7) is a *Neumann BC* (reflecting symmetry around the centerline $r = 0$) since it defines the first derivative of the solution, $u(r,z,t)$, at $r = 0$.

A second BC might be

$$u(r_0,z,t) = u_w \tag{8}$$

Eq. (8) is a *Dirichlet BC* since it defines the dependent variable, $u(r,z,t)$, at the wall, $r = r_0$.

Eq. (3) is a combination of a Dirichlet BC and a Neumann BC (it contains both $u(0,t)$ and $\frac{\partial u(0,t)}{\partial z}$), and is generally termed a *BC of the third type or a Robin BC.*

Thus, the possible BCs are *Dirichlet, Neumann, third type, linear or nonlinear.*

The coefficient $\frac{1}{r}$ in eq. (6) is termed a *variable coefficient* (it is a function of the independent variable r). Note, however, that eq. (6) is linear (PDEs with variable coefficients may be linear or nonlinear).

If the third angular coordinate is added to eq. (6), it becomes

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z} + D\left(\frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}\right) \tag{9}$$

Eq. (9) is a *linear, initial value, hyperbolic-parabolic PDE (with variable coefficients) in cylindrical coordinates*; it requires two BCs in θ (generally unnecessary in most applications because of no variation of $u(r,\theta,z,t)$ with θ (note the "units" of the various derivatives in eq. (9)). Eq. (9), with $v = 0$, is the full *Fourier's second law in cylindrical coordinates* (a parabolic PDE). Note that eq. (9) is first order in t, and second order in r, θ and z, and therefore requires the requisite number of auxiliary conditions in each independent variable

Independent variable	Highest order derivative	Type of auxiliary condition	Example
t	1	Initial condition	$u(r,\theta,z,0) = u_0$
r	2	Neumann, linear	$\partial u(r_0,\theta,z,t)/\partial r = 0$
		Third type, nonlinear	$-k\partial u(r_1,\theta,z,t)/\partial r = \epsilon(u_a^4 - u^4(r_1,\theta,z,t))$
θ	2	Neumann, linear	$\partial u(r,0,z,t)/\partial\theta = 0$
		Neumann, linear	$\partial u(r,\pi,z,t)/\partial\theta = 0$
z	2	Neumann, linear	$\partial u(r,\theta,0,t)/\partial z = 0$
		Neumann, linear	$\partial u(r,\theta,L,t)/\partial z = 0$

These ICs and BCs imply ranges for the independent variables that are not specified above. For example, the second BC

$$\partial u(r_0, \theta, z, t)/\partial r = 0$$

means

$$\partial u(r_0, \theta, z, t)/\partial r = 0, 0 \le \theta \le \pi, 0 \le z \le L, t \ge 0$$

The equation in spherical coordinates is:

$$\frac{\partial u}{\partial t} = D\left\{\frac{\partial^2 u}{\partial r^2} + \frac{2}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial u}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 u}{\partial\varphi^2}\right\} \tag{10}$$

Eq. (10) is a *linear, initial value, parabolic PDE (with variable coefficients) in spherical coordinates* (Fourier's second law in spherical coordinates).

If we have a reaction in the sphere, with a significant heat effect

$$\frac{\partial u}{\partial t} = D\left\{\frac{\partial^2 u}{\partial r^2} + \frac{2}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial u}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 u}{\partial\varphi^2}\right\} + \Delta H k_0 e^{-E/(Ru)} \tag{11}$$

Eq. (10) is a *nonlinear, initial value, parabolic PDE (with variable coefficients) in spherical coordinates.*

All of these PDEs can be stated in a coordinate-independent format, e.g.,

$$\frac{\partial u}{\partial t} = D\nabla\cdot\nabla u - \nabla\cdot(\mathbf{v}u) - R \tag{12}$$

where:

$\nabla\cdot$ (*divergence of a vector*):

Coordinate system	Components
Cartesian	$\begin{bmatrix} [\nabla]_x = \dfrac{\partial}{\partial x} \\ [\nabla]_y = \dfrac{\partial}{\partial y} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{bmatrix}$
cylindrical	$\begin{bmatrix} [\nabla]_r = \dfrac{1}{r}\dfrac{\partial}{\partial r}(r\ \ ) \\ [\nabla]_\theta = \dfrac{1}{r}\dfrac{\partial}{\partial \theta} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{bmatrix}$
spherical	$\begin{bmatrix} [\nabla]_r = \dfrac{1}{r^2}\dfrac{\partial}{\partial r}(r^2\ \ ) \\ [\nabla]_\theta = \dfrac{1}{r\sin\theta}\dfrac{\partial}{\partial \theta}(\sin\theta\ \ ) \\ [\nabla]_\phi = \dfrac{1}{r\sin\theta}\dfrac{\partial}{\partial \phi} \end{bmatrix}$

∇ *(gradient of a scalar)* :

Coordinate system	Components
Cartesian	$\begin{bmatrix} [\nabla]_x = \dfrac{\partial}{\partial x} \\ [\nabla]_y = \dfrac{\partial}{\partial y} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{bmatrix}$
cylindrical	$\begin{bmatrix} [\nabla]_r = \dfrac{\partial}{\partial r} \\ [\nabla]_\theta = \dfrac{1}{r}\dfrac{\partial}{\partial \theta} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{bmatrix}$
spherical	$\begin{bmatrix} [\nabla]_r = \dfrac{\partial}{\partial r} \\ [\nabla]_\theta = \dfrac{1}{r}\dfrac{\partial}{\partial \theta} \\ [\nabla]_\phi = \dfrac{1}{r\sin\theta}\dfrac{\partial}{\partial \phi} \end{bmatrix}$

$\nabla \cdot \nabla$ (*divergence of the gradient of a scalar*):

Coordinate system	Component
Cartesian	$\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$
cylindrical	$(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial z^2}$
spherical	$\frac{1}{r^2}\frac{\partial}{\partial r}(r^2 \ \frac{\partial}{\partial r}) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}(\sin\theta \ \frac{\partial}{\partial \theta}) + \frac{1}{r^2 \sin^2\theta}\frac{\partial^2}{\partial \phi^2}$

The derivation of $\nabla \cdot \nabla$ (the Laplacian) follows directly from the preceding components of $\nabla\cdot$ (divergence of a vector) and ∇ (gradient of a scalar).

Cartesian coordinates:

$$\nabla \cdot \nabla = (\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}) \cdot (\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Cylindrical coordinates:

$$\nabla \cdot \nabla = (\mathbf{i}_r\frac{1}{r}\frac{\partial}{\partial r}(r) + \mathbf{j}_\theta\frac{1}{r}\frac{\partial}{\partial \theta} + \mathbf{k}_z\frac{\partial}{\partial z}) \cdot (\mathbf{i}_r\frac{\partial}{\partial r} + \mathbf{j}_\theta\frac{1}{r}\frac{\partial}{\partial \theta} + \mathbf{k}_z\frac{\partial}{\partial z})$$

$$= \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial}{\partial r}) + \frac{1}{r}\frac{\partial}{\partial \theta}(\frac{1}{r}\frac{\partial}{\partial \theta}) + \frac{\partial}{\partial z}(\frac{\partial}{\partial z})$$

$$= \frac{1}{r}(\frac{\partial}{\partial r} + r\frac{\partial^2}{\partial r^2}) + \frac{1}{r^2}\frac{\partial}{\partial \theta}\frac{\partial}{\partial \theta} + \frac{\partial}{\partial z}\frac{\partial}{\partial z}$$

$$= (\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial z^2}$$

Spherical coordinates:

$$\nabla \cdot \nabla = (\mathbf{i}_r \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 \ ) + \mathbf{j}_\theta \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta \ ) + \mathbf{k}_\phi \frac{1}{r\sin\theta}\frac{\partial}{\partial\phi}) \cdot (\mathbf{i}_r \frac{\partial}{\partial r} + \mathbf{j}_\theta \frac{1}{r}\frac{\partial}{\partial\theta} + \mathbf{k}_\phi \frac{1}{r\sin\theta}\frac{\partial}{\partial\phi})$$

$$= \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 \ \frac{\partial}{\partial r}) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta \ \frac{1}{r}\frac{\partial}{\partial\theta}) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\phi}(\frac{1}{r\sin\theta}\frac{\partial}{\partial\phi})$$

$$= \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 \ \frac{\partial}{\partial r}) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta \ \frac{\partial}{\partial\theta}) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}$$

To summarize, the classification of a PDE and its auxiliary conditions is specified in terms of

- Geometric classification: elliptic, hyperbolic, parabolic
- Coordinate system
- Number of dimensions
- Initial value or boundary value
- Linearity
- Coefficients: constant or variable
- Boundary conditions: Dirichlet, Neumann, third type; linear or non-linear

This classification is useful for:

- Defining the type of PDE(s)
- Suggesting numerical methods for the solution of the PDE(s)

Of the three classes of PDEs, hyperbolic PDEs are the most difficult to integrate numerically. This can be illustrated with some famous examples of first-order, hyperbolic PDEs:

(1) *One dimensional, isothermal Euler equations written in conservation form*:

$$\mathbf{u}_t + [\mathbf{f}(\mathbf{u})]_x = \mathbf{0}$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \end{bmatrix}, \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \end{bmatrix}$$

Substitution of $\mathbf{u}$ and $\mathbf{f}(\mathbf{u})$ into the general equation gives the one-dimensional Euler equations

$$\rho_t + (\rho u)_x = 0 \qquad \text{continuity}$$

$$(\rho u)_t + (\rho u^2 + p)_x = 0 \quad \text{momentum}$$

Also, an equation of state (EOS) is required (to give a 2×2 system for $\rho(x,t)$, $u(x,t)$)

$$p = p(\rho, T)$$

These PDEs are:

- Hyperbolic (first order in x and t)
- Source of sharp fronts and discontinuities
- Nonlinear
- Difficult to solve numerically (and impossible analytically)

- Limited in their application, e.g., they have no heat transfer or reaction terms

CFD codes are available for the solution of these Euler equations.

(2) *One dimensional, nonisothermal Euler equations with chemical reaction written in conservation form*:

$$\mathbf{u}_t + [\mathbf{f}(\mathbf{u})]_x = \mathbf{g}(\mathbf{u})$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ E \\ \rho Y_1 \\ \vdots \\ \rho Y_N \end{bmatrix}, \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \\ \rho u Y_1 \\ \vdots \\ \rho u Y_N \end{bmatrix}, \mathbf{g}(\mathbf{u}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_1(T, \rho, Y_1, \cdots, Y_N) \\ \vdots \\ r_N(T, \rho, Y_1, \cdots, Y_N) \end{bmatrix}$$

where Y_i is the mass fraction of component i and N is the number of reacting species; the energy is given by

$$E = -p + \frac{\rho u^2}{2} + \rho h$$

where h is the enthalpy per unit mass.

Substitution of $\mathbf{u}$, $\mathbf{f}(\mathbf{u})$ and $\mathbf{g}(\mathbf{u})$ into the general equation gives the one-dimensional Euler equations with chemical reactions

$$
\begin{array}{lc}
\rho_t + (\rho u)_x = 0 & \text{total continuity} \\
(\rho u)_t + (\rho u^2 + p)_x = 0 & \text{momentum} \\
E_t + [(E+p)u]_x = 0 & \text{energy} \\
(\rho Y_1)_t + [\rho u Y_1]_x = r_1(T, \rho, Y_1, \cdots, Y_N) & \text{component 1 continuity} \\
\vdots & \vdots \\
(\rho Y_N)_t + [\rho u Y_N]_x = r_N(T, \rho, Y_1, \cdots, Y_N) & \text{component N continuity}
\end{array}
$$

Note that an explicit heat transfer term and a heat of reaction term do not appear in the energy balance (these can be added as in the model tubular reactor PDEs derived in the first lecture). Again, an equation of state (EOS) is required

$$p = p(\rho, E, Y_1, \cdots Y_N), T = T(\rho, E, Y_1, \cdots Y_N)$$

This is a $(3+N) \times (3+N)$ for $\rho(x,t), u(x,t), E(x,t), Y_1(x,t), \cdots Y_N(x,t)$.

CFD codes are available for the solution of these Euler equations.

(3) *Multidimensional, isothermal Euler equations (without the energy equation or chemical reaction) written in conservation form:*

$$\mathbf{u}_t + \nabla \cdot [\mathbf{f}(\mathbf{u})] = \mathbf{0}$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho \mathbf{v} \end{bmatrix}, \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho v \\ \rho \mathbf{v}\mathbf{v} + p \end{bmatrix}$$

$\mathbf{v}$ is a vector, e.g., Substitution of $\mathbf{u}$ and $\mathbf{f}(\mathbf{u})$ into the general equation gives the one-dimensional Euler equations

$$
\begin{array}{ll}
\rho_t + \nabla \cdot (\rho \mathbf{v}) = 0 & \text{continuity} \\
(\rho \mathbf{v})_t + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla p = 0 & \text{momentum}
\end{array}
$$

where ∇ is an operator applied to a vector in the continuity equation, and to a scalar and a tensor in the momentum equation. Considering ∇ a bit further, in Cartesian coordinates (with $\mathbf{v} = \mathbf{i}v_x + \mathbf{j}v_y + \mathbf{k}v_z$)

$$\nabla = \mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}$$

For example

$$\nabla \cdot (\rho\mathbf{v}) = \left(\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}\right) \cdot (\mathbf{i}\rho v_x + \mathbf{j}\rho v_y + \mathbf{k}\rho v_z)$$

$$= \frac{\partial\left(\rho v_x\right)}{\partial x} + \frac{\partial(\rho v_y)}{\partial y} + \frac{\partial\left(\rho v_z\right)}{\partial z}$$

and the (scalar) continuity equation is

$$\frac{\partial \rho}{\partial t} + \frac{\partial\left(\rho v_x\right)}{\partial x} + \frac{\partial(\rho v_y)}{\partial y} + \frac{\partial\left(\rho v_z\right)}{\partial z} = 0$$

Note that for a constant density (incompressible) fluid,

$$\rho_t = 0$$

$$\frac{\partial\left(\rho v_x\right)}{\partial x} + \frac{\partial(\rho v_y)}{\partial y} + \frac{\partial\left(\rho v_z\right)}{\partial z} = \rho(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}) + v_x\frac{\partial \rho}{\partial x} + v_y\frac{\partial \rho}{\partial x} + v_z\frac{\partial \rho}{\partial x} = 0$$

or

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0$$

$$\nabla \cdot \mathbf{v} = 0$$

Thus, an incompressible fluid is *divergence free.*

Consider the term

$$\nabla \cdot (\rho \mathbf{v}\mathbf{v})$$

The product $\mathbf{vv}$ is actually a second order (nine-component) tensor. Thus

$$\rho \mathbf{v}\mathbf{v} = \begin{bmatrix} \rho v_x v_x & \rho v_x v_y & \rho v_x v_z \\ \rho v_y v_x & \rho v_y v_y & \rho v_y v_z \\ \rho v_z v_x & \rho v_z v_y & \rho v_z v_z \end{bmatrix}$$

$$\nabla \cdot (\rho \mathbf{v}\mathbf{v}) = \begin{bmatrix} \mathbf{i}\left\{ \frac{\partial}{\partial x} v_x(\rho v_x) + \frac{\partial}{\partial y} v_y(\rho v_x) + \frac{\partial}{\partial z} v_z(\rho v_x) \right\} \\ \\ \mathbf{j}\left\{ \frac{\partial}{\partial x} v_x(\rho v_y) + \frac{\partial}{\partial y} v_y(\rho v_y) + \frac{\partial}{\partial z} v_z(\rho v_y) \right\} \\ \\ \mathbf{k}\left\{ \frac{\partial}{\partial x} v_x(\rho v_z) + \frac{\partial}{\partial y} v_y(\rho v_z) + \frac{\partial}{\partial z} v_z(\rho v_z) \right\} \end{bmatrix}$$

Then, using this result (a vector) in the momentum equation

$$(\rho \mathbf{v})_t + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla p = 0$$

and equating like components

$$\frac{\partial}{\partial t}(\rho v_x) + \frac{\partial}{\partial x}(\rho v_x v_x) + \frac{\partial}{\partial y}(\rho v_x v_y) + \frac{\partial}{\partial z}(\rho v_x v_z) + \frac{\partial P}{\partial x} = 0 \text{ (}\mathbf{i}\text{ component)}$$

$$\frac{\partial}{\partial t}(\rho v_y) + \frac{\partial}{\partial x}(\rho v_y v_x) + \frac{\partial}{\partial y}(\rho v_y v_y) + \frac{\partial}{\partial z}(\rho v_y v_z) + \frac{\partial P}{\partial y} = 0 \text{ (}\mathbf{j}\text{ component)}$$

$$\frac{\partial}{\partial t}(\rho v_z) + \frac{\partial}{\partial x}(\rho v_z v_x) + \frac{\partial}{\partial y}(\rho v_z v_y) + \frac{\partial}{\partial z}(\rho v_z v_z) + \frac{\partial P}{\partial z} = 0 \text{ (}\mathbf{k}\text{ component)}$$

Again, these PDEs are:

- Hyperbolic (first order in x, y, z and t)
- Source of sharp fronts and discontinuities (but now in three dimensions)
- Nonlinear
- Difficult to solve numerically (and impossible analytically)
- Limited in their application, e.g., they have no heat transfer, reaction terms

CFD codes are available for the solution of these Euler equations.

These equations can be simplified through the continuity equation, e.g., for the **i** component:

$$v_x \frac{\partial}{\partial t}(\rho) + v_x \frac{\partial}{\partial x}(\rho v_x) + v_x \frac{\partial}{\partial y}(\rho v_y) + v_x \frac{\partial}{\partial z}(\rho v_z)$$

$$\rho \frac{\partial}{\partial t}(v_x) + \rho v_x \frac{\partial}{\partial x}(\rho v_x) + \rho v_y \frac{\partial}{\partial y}(\rho v_x) + \rho v_z \frac{\partial}{\partial z}(\rho v_x) + \frac{\partial P}{\partial x} = 0$$

or

$$v_x \left\{ \frac{\partial \rho}{\partial t} + \frac{\partial(\rho v_x)}{\partial x} + \frac{\partial(\rho v_y)}{\partial y} + \frac{\partial(\rho v_z)}{\partial z} \right\}$$

$$\rho \frac{\partial v_x}{\partial t} + \rho v_x \frac{\partial v_x}{\partial x} + \rho v_y \frac{\partial v_x}{\partial y} + \rho v_z \frac{\partial v_x}{\partial z} + \frac{\partial P}{\partial x} = 0$$

and the first row is zero through the continuity equation. In general,

$$\rho \mathbf{v}_t + \rho \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p = 0$$

which is an alternate form of the Euler equations.

Finally, if a viscosity term is added to the momentum equation

$$(\rho\mathbf{v})_t + \nabla \cdot (\rho\mathbf{v}\mathbf{v}) + \nabla p - \mu\nabla^2\mathbf{v} = 0$$

which are the *Navier Stokes equations.*

The preceding PDEs are written in Cartesian coordinates. As another general approach, we can start with PDEs written in terms of vector operators that are independent of the coordinate system, e.g.,

$\rho_t + \nabla \cdot (\rho\mathbf{v}) = 0$	continuity
$(\rho\mathbf{v})_t + \nabla \cdot (\rho\mathbf{v}\mathbf{v}) + \nabla p = 0$	momentum (Euler)
$\rho\mathbf{v}_t + \rho\mathbf{v}\cdot\nabla\mathbf{v} + \nabla p = 0$	momentum (Euler)
$(\rho\mathbf{v})_t + \nabla \cdot (\rho\mathbf{v}\mathbf{v}) + \nabla p - \mu\nabla^2\mathbf{v} = 0$	momentum (Navier Stokes)
$\rho\mathbf{v}_t + \rho\mathbf{v}\cdot\nabla\mathbf{v} + \nabla p - \mu\nabla^2\mathbf{v} = 0$	momentum (Navier Stokes)

With these coordinate-independent equations as the starting point, we can go to the three preceding tables to write equations in Cartesian, cylindrical or spherical coordinates. This approach can also be applied to other coordinate systems, perhaps fitted to an body that is not easily described in one of the well-established coordinate systems (body-fitted coordinates are an active area of CFD).

As an example of this simplification process, consider the x-component of the Navier Stokes equations

$$\rho\frac{\partial v_x}{\partial t} + \rho v_x\frac{\partial v_x}{\partial x} + \rho v_y\frac{\partial v_x}{\partial y} + \rho v_z\frac{\partial v_x}{\partial z} + \frac{\partial P}{\partial x} - \mu(\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2}) = 0$$

If we consider a one-dimensional problem with no pressure gradient, this equation reduces to

$$\rho\frac{\partial v_x}{\partial t} + \rho v_x\frac{\partial v_x}{\partial x} - \mu\frac{\partial^2 v_x}{\partial x^2} = 0$$

which is *Burgers' equation.* This is an unusual (and widely studied) PDE because

- It is nonlinear, yet has a known, exact (analytical solution)
- The solution exhibits moving fronts that can be made arbitarily sharp by decreasing the kinematic viscosity $\upsilon = \mu/\rho$.

Burgers' equation for $\upsilon = \mu/\rho = 0$ reduces to the *inviscid Burgers' equation* or *nonlinear advection equation.*

$$\frac{\partial v_x}{\partial t} + v_x \frac{\partial v_x}{\partial x} = 0$$

This equation can propagate discontinuties, and therefore provides a stringent test problem for PDEs.

As general as all of the preceding equations may appear, they are actually rather limited, e.g., they do not contain heat transfer or heat of reaction terms. As an added thought, general purpose codes that solve these equations may not be as general as we would like. We should at least be sure that we understand the underlying PDEs of these codes.

Therefore, a second approach to developing a PDE model is to start with a basic incremental volume and write the appropriate conservation equations for it (including all of the effects that are considered important). In other words, derive the mathematical model first, then look for, or write, a code that solves the model equations. The process should not be reversed, i.e., we should not start with a general code, then modify the model so that it fits into the code (and thereby possibly drop important terms that should be included).

This might seem like an inefficient way to proceed, i.e., not using an (expensive), readily available code, but rather writing a new code for the particular mathematical model. But the latter ensures that we are solving the appropriate model, and not one that has been modified to fit into the software (or worse, using an available code incorrectly by not solving the equations we think we are solving). In other words, the equations are the fundamental starting point, and the calculations (software) to solve the equations are secondary.

Additionally, we can examine a code written for the particular problem system in whatever detail is necessary to fully understand the calculations; a library code, however, may not provide the details of the calculations in the detail required to fully understand the calculations. As a minimum, we should fully understand what the library code is doing.

As a specific example of this second approach, consider the PDE for a heated pipe that originates from an energy balance written on a section of length Δz. This obviously is a simple physical system, but it leads to a PDE that is surprisingly difficult to solve numerically, depending on the conditions of the problem.

$$A\Delta z\rho C_p\frac{\partial T}{\partial t} = Av\rho C_pT_{z-\Delta z} - Av\rho C_pT_z + \pi D\Delta zh(T_w - T)$$

Division by $A\Delta z\rho C_p$ with $A = \pi D^2/4$ and minor rearrangement gives

$$\frac{\partial T}{\partial t} = -v\frac{(T_z - T_{z-\Delta z})}{\Delta z} + \frac{\pi Dh}{(\pi D^2/4)\rho C_p}(T_w - T)$$

In the limit as $\Delta z \rightarrow 0$, we arrive at the PDE

$$\frac{\partial T}{\partial t} = -v\frac{\partial T}{\partial z} + \frac{4h}{D\rho C_p}(T_w - T) \tag{1}$$

The initial condition (IC) is

$$T(z,0) = T_0 \tag{2}$$

and the boundary condition (BC) is

$$T(0,t) = T_e \tag{3}$$

Eq. (1) is a first order, linear hyperbolic PDE. Eqs. (1) to (3) provide a stringent test problem for numerical methods and software. We will use the analytical solution to evaluate the numerical methods.

If

$$u = T - T_w, \alpha = \frac{4h}{D\rho C_p} \tag{4}$$

eqs. (1) to (3) become

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z} - \alpha u \tag{5}$$

$$u(z,0) = T_0 - T_w \tag{6}$$

$$u(0,t) = T_e - T_w \tag{7}$$

If the Laplace transform of $u(z,t)$ with respect to t is defined as

$$L_t\left\{u(z,t)\right\} = \int_0^\infty u(z,t)e^{-st}dt = U(z,s)$$

eqs. (5) and (6) transform to

$$sU - (T_0 - T_w) = -v\frac{dU}{dz} - \alpha U$$

or

$$\frac{dU}{dz} = -(1/v)(s+\alpha)U + (1/v)(T_0 - T_w) \tag{8}$$

Eq. (8) is a linear, constant coefficient, nonohomogeneous ODE with homogeneous and particular solutions

$$U_h = c_1 e^{-(1/v)(s+\alpha)z}$$

$$U_p = \frac{T_0 - T_w}{s+\alpha}$$

Thus

$$U(z,s) = c_1 e^{-(1/v)(s+\alpha)z} + \frac{T_0 - T_w}{s+\alpha}$$

Application of BC (7) to evaluate constant c_1 gives

$$\frac{T_e - T_w}{s} = c_1 + \frac{T_0 - T_w}{s+\alpha}$$

or

$$c_1 = \frac{T_e - T_w}{s} - \frac{T_0 - T_w}{s+\alpha}$$

Thus

$$U(z,s) = \left(\frac{T_e - T_w}{s} - \frac{T_0 - T_w}{s+\alpha}\right) e^{-(1/v)(s+\alpha)z} + \frac{T_0 - T_w}{s+\alpha}$$

$$= e^{-(\alpha/v)z} \left(\frac{T_e - T_w}{s} - \frac{T_0 - T_w}{s+\alpha}\right) e^{-(z/v)s} + \frac{T_0 - T_w}{s+\alpha} \qquad (9)$$

Inversion of eq. (9) gives

$$u(z,t) = L_t^{-1}\{U(z,s)\}$$

$$= e^{-(\alpha/v)z}\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha\{t-(z/v)\}}\right\} h(t-(z/v)) + (T_0 - T_w)e^{-\alpha t}$$

where

$$\begin{aligned} h(t) = 0 \quad t < 0 \\ h(t) = 1 \quad t > 0 \end{aligned}$$

Thus

$$T(z,t) = T_w + u(z,t)$$

$$= T_w + e^{-(\alpha/v)z}\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha\{t-(z/v)\}}\right\} h(t-(z/v)) + (T_0 - T_w)e^{-\alpha t} \tag{10}$$

Check (being a little "loose" with $h(z,t)$ and it's derivative at $t = z/v$):

$$T(z,0) = T_w + 0 + (T_0 - T_w)e^{-0} = T_0$$

$$T(0,t) = T_w + e^0\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha t}\right\} + (T_0 - T_w)e^{-\alpha t} = T_e$$

PDE, eq. (1)	From eq. (10)
$\partial T/\partial t$	$e^{-(\alpha/v)z}\left\{-(T_0 - T_w)\, e^{-\alpha\{t-(z/v)\}}(-\alpha)\right\} + (T_0 - T_w)(-\alpha)e^{-\alpha t}$
$v\partial T/\partial z$	$v\left[-(\alpha/v)e^{-(\alpha/v)z}(T_e - T_w)\right]$
$-\alpha(T_w - T)$	$-\alpha T_w + \alpha[T_w + e^{-(\alpha/v)z}\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha\{t-(z/v)\}}\right\} h(t-(z/v))$ $+(T_0 - T_w)e^{-\alpha t}]$
0	0

Eq. (10) consists of two parts (as defined by $h(t-(z/v))$):

$$t < z/v, T(z,t) = T_w + (T_0 - T_w)e^{-\alpha t} \tag{10a}$$

$$t > z/v, T(z,t) = T_w + e^{-(\alpha/v)z}\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha\{t-(z/v)\}}\right\} + (T_0 - T_w)e^{-\alpha t} \tag{10b}$$

Eqs. (10a) and (10b) give a discontinuity at $t = z/v$. To see this, if we consider eq. (10) at $t = z/v$,

(a) with $u(t - z/v) = 0$,

$$T(z,t) = T_w + e^{-(\alpha/v)z}\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha\{t-(z/v)\}}\right\} 0 + (T_0 - T_w)e^{-\alpha t}$$

$$= T_w + (T_0 - T_w)e^{-\alpha t} \tag{10c}$$

(b) with $u(t - z/v) = 1$

$$T(z,t) = T_w + e^{-\alpha t}\left\{(T_e - T_w) - (T_0 - T_w)\, e^{-\alpha\{0\}}\right\}(1) + (T_0 - T_w)e^{-\alpha t}$$

$$= T_w + (T_e - T_0)e^{-\alpha t} + (T_0 - T_w)e^{-\alpha t} = T_w + (T_e - T_w)e^{-\alpha t} \tag{10d}$$

Only when $T_0 = T_e$ are eqs. (10c) and (10d) the same, i.e., the initial and entering conditions are the same. Otherwise, with $T_e \neq T_0$, a discontinuity is introduced that propagates through the system. *This is a basic property of hyperbolic PDEs, i.e., they propagate discontinuities, which is the reason they are the most difficult of the three classes of PDEs (elliptic, parabolic, hyperbolic) to solve numerically*; in particular, the discontinuity often occurs through an incompatibility between the initial and boundary conditions, e.g., $T_e \neq T_0$.

There are some other interesting special cases of eqs. (10a) and (10b)

$$\begin{array}{lll} t = 0 & & \\ & t < z/v & T(z,t) = T_0 \\ & t > z/v & \text{NA} \end{array} \tag{11a}$$

$$\begin{array}{lll} t \to \infty & & \\ & t < z/v & \text{NA} \\ & t > z/v & T(z,t) = T_w + e^{-(\alpha/v)z}(T_e - T_w) \end{array} \tag{11b}$$

Within eqs. (11b),

$$\begin{array}{l} t \to \infty \\ \qquad \begin{array}{lll} t > z/v & z = 0 & T(z,t) = T_e \\ t > z/v & z \to \infty & T(z,t) = T_w \end{array} \end{array} \tag{11c}$$

Also,

$$\begin{array}{l} \alpha = 0 \\ \qquad \begin{array}{ll} t < z/v & T(z,t) = T_0 \\ t > z/v & T(z,t) = T_e \end{array} \end{array} \tag{11d}$$

Eqs. (11d) in particular illustrates the difficulty in computing numerical solutions to first order, hyperbolic PDEs, i.e., the discontinuity generated at $z = 0$ where $T(z,t)$ switches from T_0 to T_e,) (again, first order, hyperbolic PDEs propagate discontinuities).

To summarize the case $\alpha = 0$, eqs. (1) to (3) reduce to

$$\frac{\partial T}{\partial t} + v\frac{\partial T}{\partial z} = 0 \tag{12a}$$

$$T(z,0) = T_0 \tag{12b}$$

$$T(0,t) = T_e \tag{12c}$$

for which the solution is

$$T(z,t) = T_0, t < z/v$$

$$T(z,t) = T_e, t > z/v$$

or

$$T(z,t) = T_0 + (T_e - T_0)h(t - z/v) \tag{13}$$

If $T_0 = 0, T_e = 1$, this reduces to

$$T(z,t) = h(t - z/v) \tag{14}$$

where again $h(t)$ is the Heaviside unit step function. Note that eq. (14) indicates the Heaviside unit step function propagates along the system with velocity v.

Another approach to eq. (11b) is to start with eq. (1) at steady state

$$\frac{\partial T}{\partial t} = -v\frac{\partial T}{\partial z} + \frac{4h}{D\rho C_p}(T_w - T) = 0 \tag{15}$$

Since eq. (15) has only one independent variable, z, it is actually an ODE. The corresponding auxiliary condition is eq. (3) at steady state

$$T(0) = T_e \tag{16}$$

The solution to eq. (15) will be a homogeneous part plus a particular part

$$T_h(z) = c_1 e^{-(\alpha/v)z}, \alpha = \frac{4h}{D\rho C_p}$$

$$T_p(z) = T_w$$

Thus

$$T(z) = T_w + c_1 e^{-(\alpha/v)z}$$

Then from eq. (16),

$$T_e = T_w + c_1 e^{-0}$$

and the solution is

$$T(z) = T_w + (T_e - T_w)e^{-(\alpha/v)z}$$

which is eq. (11b).

We now consider:

- How eqs. (1) to (3) can be programmed
- The comparison of this numerical solution with the analytical solution, eq. (10).

In order to program a PDE, we generally start by defining a spatial grid, in this case, a grid in z. An index, i, is used to denote a particular position along the grid. The spatial derivative in eq. (1), $\frac{\partial T}{\partial z}$, is then approximated algebraically, in this case by a finite difference, at each of the grid points. For example, at the general grid point i

$$\frac{\partial T_i}{\partial z} \cong \frac{T_i - T_{i-1}}{\Delta z}$$

This leads to a system of ODEs, one at each grid point, that approximate the PDE of eq. (1)

$$\frac{dT_i}{dz} = -v(\frac{T_i - T_{i-1}}{\Delta z}) + \frac{\pi D h}{(\pi D^2/4)\rho C_p}(T_w - T_i), i = 1, 2, \cdots, n \tag{17}$$

(note that in deriving eq. (17), we have essentiually reversed the process of deriving eq. (1) starting with an incremental heat balance). This system of simultanenous ODEs can then be integrated by any of the ODE integration methods we discussed previously, e.g., Euler's method, the modified Euler method, RKF45, BDF, etc. In the following program we use Euler's method to provide coding that is easy to understand (but, of course, Euler's method is only first order correct, so we might eventually want to switch to a more acccurate ODE integrator, perhaps with automatic step size adjustment).

The details of the procedure are most easily understood by considering the actual computer programming, as illustrated by the following program.

Time and Spatial Discretization

We now have to discretize the time derivative and the spatial derivative in eq. (1). A Matlab program that does this is listed below (this program illustrates all of the essential features for a complete PDE solution):

```
%   Fluid flow through a heated pipe
%
%   The PDE that models the heated pipe is
%
%      T = -v*T  + (4*h)/(D*rho*Cp)*(Tw - T)                 (1)
%       t      z
%
%   with the initial and boundary conditions
%
%      T(z,0) = T0, T(0,t) = Te                           (2)(3)
%
%   where
%
%      T     fluid temperature
%
%      t     time
%
%      z     position along the pipe
%
%      v     fluid velocity
%
%      h     fluid to pipe wall heat transfer coefficient
%
%      D     pipe diameter
%
%      rho   fluid density
%
%      Cp    fluid specific heat
```

```
%
%      T0    initial fluid temperature
%
%      Te    entering fluid temperature
%
%      Tw    pipe wall temperature
%
%      L     pipe length
%
%   Of particular interest is the fluid exiting temperature, T(L,t).
%
%   The method of lines (MOL) solution of eq. (1) is coded below.
%   Specifically, the spatial derivative in eq. (1) is replaced
%   by a three point, finite difference (FD) approximation, both
%   centered and upwind.  The resulting system of ODEs in t,
%   defined on a 21-point grid in z, is then integrated by the
%   Euler method.
%
%   The analytical solution to eqs. (1) to (3) is also programmed
%   for comparison with the numerical solution.
%
%   The following code is run for the four special cases:
%
%      Case 1: h = 2, T0 = 25, Te = 25, Tw = 100
%
%      Case 2: h = 0.2, T0 = 25, Te = 25, Tw = 100
%
%      Case 3: h = 0, T0 = 25, Te = 25, Tw (NA)
%
%      Case 4: h = 0, T0 = 25, Te = 100, TW (NA)
%
% Open output files
  fid1=fopen('htex1.out','w');
  fid2=fopen('htex1.plt','w');
%
% Model parameters (defined in the comments above)
  cp=1.0;
  rho=1.0;
  v=0.5;
  d=1.0;
  zl=1.0;
  n=21;
  dz=zl/(n-1);
%
% Select three point, finite differencing (FD) of spatial
% derivative
%
%   ifd = 1: Centered approximations
%
%   ifd = 2: Two point upwind approximation
%
%   ifd = 3: Five point, biased upwind approximation
%
%   ifd = 4: van Leer flux limiter
%
  ifd=1;
%
% Cases listed above
  for ncase=1:4
%
% Initial, final times, integration interval, number of Euler
% steps for each output
  t=0.0; tf=5.0; h=0.001; nout=250;
  if ncase==1
     hc=2.0;
     T0=25.0;
     Te=25.0;
     Tw=100.0;
  end
  if ncase==2
     hc=0.2;
     T0=25.0;
     Te=25.0;
     Tw=100.0;
  end
  if ncase==3
```

```
     hc=0.0;
     T0=25.0;
     Te=25.0;
     Tw=100.0;
  end
  if ncase==4
     hc=0.0;
     T0=25.0;
     Te=100.0;
     Tw=100.0;
  end
%
% Initial conditions
  for i=1:n
    T(i)=T0;
  end
%
% Write h
  fprintf(fid1,'\n\n ncase = %5d    h = %10.3e\n\n',ncase,h);
%
% Write heading
  fprintf(fid1,'   t     t-zl/v   T(zl,t)  Ta(zl,t)      diff\n');
%
% Integrate until t = tf
  while t < tf*1.0001
%
% Monitor solution by displaying t
  t
%
% Write selected output, including analytical solution and difference
% between numerical and analytical solutions
  alpha=4.0*hc/(d*rho*cp);
  Ta=Tw+(T0-Tw)*exp(-alpha*t);
  if(t > zl/v)Ta=Ta+exp(-alpha/v*zl)*((Te-Tw)-(T0-Tw)*...
                    exp(-alpha*(t-zl/v))); end
  diff=T(n)-Ta;
  fprintf(fid1,'%5.2f%10.2f%10.2f%10.2f%10.3f\n',...
          t,t-zl/v,T(n),Ta,diff);
  fprintf(fid2,'%5.2f%10.2f%10.2f%10.2f%10.3f\n',...
          t,t-zl/v,T(n),Ta,diff);
%
% Take nout Euler steps
  for iout=1:nout
%
%   Temporal derivatives
%
%     Centered approximations
      if(ifd==1)
%
%       Boundary condition at z = 0
        T(1)=Te;
%
%       Spatial derivative
        [Tz]=dss002(0.0,zl,n,T);
%
%       Temporal derivative
        Tt(1)=0.0;
        for i=2:n
          Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
        end
%
%     End of three point centered approximation
      end
%
%     Two point upwind approximation
      if(ifd==2)
%
%       Boundary condition at z = 0
        T(1)=Te;
%
%       Spatial derivative
        [Tz]=dss012(0.0,zl,n,T,v);
%
%       Temporal derivative
        Tt(1)=0.0;
        for i=2:n
```

```
          Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
        end
%
%     End of two point upwind approximation
      end
%
%     Five point, biased upwind approximation
      if(ifd==3)
%
%       Boundary condition at z = 0
        T(1)=Te;
%
%       Spatial derivative
        [Tz]=dss020(0.0,zl,n,T,v);
%
%       Temporal derivative
        Tt(1)=0.0;
        for i=2:n
          Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
        end
%
%     End of five point, biased upwind approximation
      end
%
%     van Leer flux limiTer
      if(ifd==4)
%
%       Boundary condition at z = 0
        T(1)=Te;
%
%       Spatial derivative
        [Tz]=vanl2(0.0,zl,n,T,v);
%
%       Temporal derivative
        Tt(1)=0.0;
        for i=2:n
          Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
        end
```

```
%
%     End of van Leer flux limiter
      end
%
%   Take Euler step
    for i=1:n
      T(i)=T(i)+Tt(i)*h;
    end
    t=t+h;
%
% Next Euler step
  end
%
% Next output
  end
%
% Next case
  end
```

We can note the following points about this program:

(1) The model parameters are first set

```
% Model parameters (defined in the comments above)
  cp=1.0;
  rho=1.0;
  v=0.5;
  d=1.0;
  zl=1.0;
  n=21;
  dz=zl/(n-1);
```

(2) A spatial differentiator for the derivative T_z is selected

```
% Select finite differencing (FD) of spatial derivative
```

```
%
%   ifd = 1: Centered approximations
%
%   ifd = 2: Two point upwind approximation
%
%   ifd = 3: Five point, biased upwind approximation
%
%   ifd = 4: van Leer flux limiter
%
  ifd=1;
```

The details of these differentiators will be considered below.

(3) The integration parameters and the specific values of the model parameters for one of four cases are selected

```
%   The following code is run for the four special cases:
%
%      Case 1: h = 2, T0 = 25, Te = 25, Tw = 100
%
%      Case 2: h = 0.2, T0 = 25, Te = 25, Tw = 100
%
%      Case 3: h = 0, T0 = 25, Te = 25, Tw (NA)
%
%      Case 4: h = 0, T0 = 25, Te = 100, TW (NA)

% Cases listed above
  for ncase=1:4
%
% Initial, final times, integration interval, number of Euler
% steps for each output
  t=0.0; tf=5.0; h=0.001; nout=250;
  if ncase==1
     hc=2.0;
     T0=25.0;
     Te=25.0;
     Tw=100.0;
  end
  if ncase==2
     hc=0.2;
     T0=25.0;
     Te=25.0;
     Tw=100.0;
  end
  if ncase==3
     hc=0.0;
     T0=25.0;
     Te=25.0;
     Tw=100.0;
  end
  if ncase==4
     hc=0.0;
     T0=25.0;
     Te=100.0;
     Tw=100.0;
  end
```

If the integration step h in the Euler integration is increased above $h = 0.001$, the stability limit of the Euler method is exceeded, i.e., $|h\lambda| < 2$, and the calculation goes unstable.

(4) The 21 initial conditions are set (for the 21-point grid in z)

```
% Initial conditions
  for i=1:n
    T(i)=T0;
  end
```

(5) A heading for the output is next

```
%
% Write h
```

```
  fprintf(fid1,'\n\n ncase = %5d    h = %10.3e\n\n',ncase,h);
%
% Write heading
  fprintf(fid1,'   t     t-zl/v   T(zl,t)  Ta(zl,t)      diff\n');
```

Note that both the numerical and analytical solutions, and their difference are displayed in fid1.

(6) The time stepping begins at t = 0 and ends at t = tf

```
% Integrate until t = tf
  while t < tf*1.0001
%
% Monitor solution by displaying t
  t
```

(7) The analytical solution. eq. (10), is evaluated, and the numerical and analytical solutions and their differenerence are then written to files, fid1 and fid2

```
% Write selected output, including analytical solution and difference
% between numerical and analytical solutions
  alpha=4.0*hc/(d*rho*cp);
  Ta=Tw+(T0-Tw)*exp(-alpha*t);
  if(t > zl/v)Ta=Ta+exp(-alpha/v*zl)*((Te-Tw)-(T0-Tw)*...
                    exp(-alpha*(t-zl/v))); end
  diff=T(n)-Ta;
  fprintf(fid1,'%5.2f%10.2f%10.2f%10.2f%10.3f\n',...
          t,t-zl/v,T(n),Ta,diff);
  fprintf(fid2,'%5.2f%10.2f%10.2f%10.2f%10.3f\n',...
          t,t-zl/v,T(n),Ta,diff);
```

(8) nout (= 250) Euler steps are taken for the n (= 21) ODEs in t

```
% Take nout Euler steps
  for iout=1:nout
```

(9) The calculation of the time derivatives of eq. (1), T_t, requires first the evaluation of the spatial derivative, T_z. This is done by one of the four spatial differentiators selected by ifd (see (2) above). The RHS of eq (1) is then computed in each case

```
%   Temporal derivatives
%
%     Centered approximations
      if(ifd==1)
%
%       Boundary condition at z = 0
        T(1)=Te;
%
%       Spatial derivative
        [Tz]=dss002(0.0,zl,n,T);
%
%       Temporal derivative
        Tt(1)=0.0;
        for i=2:n
          Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
        end
%
%     End of three point centered approximation
      end
%
%     Two point upwind approximation
      if(ifd==2)
%
%       Boundary condition at z = 0
        T(1)=Te;
%
%       Spatial derivative
        [Tz]=dss012(0.0,zl,n,T,v);
%
```

```
%         Temporal derivative
          Tt(1)=0.0;
          for i=2:n
            Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
          end
%
%       End of two point upwind approximation
        end
%
%       Five point, biased upwind approximation
        if(ifd==3)
%
%         Boundary condition at z = 0
          T(1)=Te;
%
%         Spatial derivative
          [Tz]=dss020(0.0,zl,n,T,v);
%
%         Temporal derivative
          Tt(1)=0.0;
          for i=2:n
            Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
          end
%
%       End of five point, biased upwind approximation
        end
%
%       van Leer flux limiter
        if(ifd==4)
%
%         Boundary condition at z = 0
          T(1)=Te;
%
%         Spatial derivative
          [Tz]=vanl2(0.0,zl,n,T,v);
%
%         Temporal derivative
          Tt(1)=0.0;
        for i=2:n
          Tt(i)=-v*Tz(i)+alpha*(Tw-T(i));
        end
%
%     End of van Leer flux limiter
      end
```

(10) Once the n (= 21) time derivatives are computed, the n ODEs are integrated by the Euler method to move the solution along (as parallel lines in z)

```
%   Take Euler step
    for i=1:n
      T(i)=T(i)+Tt(i)*h;
    end
    t=t+h;
```

(11) The loop for the nout (= 250) Euler steps is then completed and the numerical and analytical solutions are printed before the next set of nout Euler steps is taken (see (7) above). First, the test for the final time, tf, is checked (see (6) above).

```
% Next Euler step
  end
%
% Next output
  end
%
% Next case
  end
```

The preceding Matlab code is in

http://www.lehigh.edu/ wes1/apci/htex1.m

The extension of this code to systems of ODEs/PDEs is straightforward; their time derivatives would be added, and the corresponding ODEs would be integrated, either by the Euler method, or by a more accurate method, e.g.,

- The modified Euler method, the classical Runge Kutta method, the Runge Kutta Fehlberg method, either fixed step or variable step
- A library integrator called where the Euler integration is programmed, e.g., a call to RKF45, LSODE, LSODES, DASSL, a NAG ODE integrator, a Matlab ODE integrator (ode23, ode45)

This modular and flexible approach to programming PDEs is termed the *method of lines* (MOL). Additionally, the algorithm for calculating the spatial derivative(s) can easily be changed (by calling another routine in place of one of the library differentiators).

The output of the preceding program for ifd = 1 (three point centered FD) is listed below (and is plotted in htex1d1.ps)

```
ncase =     1    h =  1.000e-03

  t      t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00      -2.00     25.00     25.00     0.000
0.25      -1.75     89.93     89.85     0.081
0.50      -1.50     98.65     98.63     0.022
0.75      -1.25     99.82     99.81     0.004
1.00      -1.00     99.98     99.97     0.001
1.25      -0.75    100.00    100.00     0.000
1.50      -0.50    100.00    100.00     0.000
1.75      -0.25    100.00    100.00     0.000
2.00       0.00    100.00    100.00     0.000
2.25       0.25    100.00    100.00     0.000
2.50       0.50    100.00    100.00     0.000
2.75       0.75    100.00    100.00     0.000
3.00       1.00    100.00    100.00     0.000
3.25       1.25    100.00    100.00     0.000
```

```
3.50      1.50   100.00    100.00     0.000
3.75      1.75   100.00    100.00     0.000
4.00      2.00   100.00    100.00     0.000
4.25      2.25   100.00    100.00     0.000
4.50      2.50   100.00    100.00     0.000
4.75      2.75   100.00    100.00     0.000
5.00      3.00   100.00    100.00     0.000


ncase =     2    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     38.60     38.60     0.005
0.50     -1.50     49.73     49.73     0.008
0.75     -1.25     58.85     58.84     0.010
1.00     -1.00     66.31     66.30     0.011
1.25     -0.75     72.42     72.41     0.011
1.50     -0.50     77.41     77.41    -0.003
1.75     -0.25     81.37     81.51    -0.134
2.00      0.00     84.10     84.86    -0.756
2.25      0.25     85.22     84.86     0.363
2.50      0.50     85.03     84.86     0.168
2.75      0.75     84.69     84.86    -0.164
3.00      1.00     84.81     84.86    -0.047
3.25      1.25     84.90     84.86     0.039
3.50      1.50     84.81     84.86    -0.052
3.75      1.75     84.82     84.86    -0.035
4.00      2.00     84.85     84.86    -0.005
4.25      2.25     84.82     84.86    -0.036
4.50      2.50     84.83     84.86    -0.025
4.75      2.75     84.84     84.86    -0.019
5.00      3.00     84.83     84.86    -0.030


ncase =     3    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
```

```
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     25.00     25.00     0.000
0.50     -1.50     25.00     25.00     0.000
0.75     -1.25     25.00     25.00     0.000
1.00     -1.00     25.00     25.00     0.000
1.25     -0.75     25.00     25.00     0.000
1.50     -0.50     25.00     25.00     0.000
1.75     -0.25     25.00     25.00     0.000
2.00      0.00     25.00     25.00     0.000
2.25      0.25     25.00     25.00     0.000
2.50      0.50     25.00     25.00     0.000
2.75      0.75     25.00     25.00     0.000
3.00      1.00     25.00     25.00     0.000
3.25      1.25     25.00     25.00     0.000
3.50      1.50     25.00     25.00     0.000
3.75      1.75     25.00     25.00     0.000
4.00      2.00     25.00     25.00     0.000
4.25      2.25     25.00     25.00     0.000
4.50      2.50     25.00     25.00     0.000
4.75      2.75     25.00     25.00     0.000
5.00      3.00     25.00     25.00     0.000


ncase =     4    h =  1.000e-03

  t      t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     25.00     25.00     0.000
0.50     -1.50     25.00     25.00     0.000
0.75     -1.25     25.00     25.00     0.000
1.00     -1.00     25.00     25.00     0.000
1.25     -0.75     25.03     25.00     0.027
1.50     -0.50     25.63     25.00     0.627
1.75     -0.25     30.73     25.00     5.733
2.00      0.00     51.17     25.00    26.166
2.25      0.25     90.33    100.00    -9.665
2.50      0.50    117.36    100.00    17.363
2.75      0.75    103.95    100.00     3.951
```

```
 3.00      1.00     88.96    100.00   -11.039
 3.25      1.25    103.31    100.00     3.309
 3.50      1.50    104.99    100.00     4.991
 3.75      1.75     93.70    100.00    -6.300
 4.00      2.00    102.14    100.00     2.136
 4.25      2.25    102.54    100.00     2.540
 4.50      2.50     95.71    100.00    -4.288
 4.75      2.75    102.78    100.00     2.784
 5.00      3.00    100.18    100.00     0.176
```

We can note the following points about this output:

(1) The three point centered FD approximations give good accuracy for Cases 1 to 3 which are for relatively smooth problems. In particular, the FD solution is exact for Case 3 (the constant solution is differentiated exactly).

(2) The centered FD approximations do not give an accurate solution for the discontinuous solution of Case 4. This result indicates that *centered approximations should not be used for nonsmooth solutions of first order hyperbolic PDEs.*

The library differentiator for the three point centered FD approximations is listed below

```
      function [ux]=dss002(xl,xu,n,u)
%...
%...  FUNCTION DSS002 COMPUTES THE FIRST DERIVATIVE, U , OF A
%...                                                 X
%...  VARIABLE U OVER THE SPATIAL DOMAIN XL LE X LE XU
%...
%...  ARGUMENT LIST
%...
%...     XL      LOWER BOUNDARY VALUE OF X (INPUT)
%...
%...     XU      UPPER BOUNDARY VALUE OF X (INPUT)
%...
%...     N       NUMBER OF GRID POINTS IN THE X DOMAIN INCLUDING THE
```

```
%...             BOUNDARY POINTS (INPUT)
%...
%...    U        ONE-DIMENSIONAL ARRAY CONTAINING THE VALUES OF U AT
%...             THE N GRID POINT POINTS FOR WHICH THE DERIVATIVE IS
%...             TO BE COMPUTED (INPUT)
%...
%...    UX       ONE-DIMENSIONAL ARRAY CONTAINING THE NUMERICAL
%...             VALUES OF THE DERIVATIVES OF U AT THE N GRID POINTS
%...             (OUTPUT)
%...
%...  THE WEIGHTING COEFFICIENTS CAN BE SUMMARIZED AS
%...
%...         -3   4  -1
%...
%...    1/2  -1   0   1
%...
%...          1  -4   3
%...
%...  WHICH ARE THE COEFFICIENTS REPORTED BY BICKLEY FOR N = 2, M =
%...  1, P = 0, 1, 2 (BICKLEY, W. G., FORMULAE FOR NUMERICAL DIFFER-
%...  ENTIATION, MATH. GAZ., VOL. 25, 1941).
%...
%...  COMPUTE THE SPATIAL INCREMENT
      dx=(xu-xl)/(n-1);
      r2fdx=1./(2.*dx);
      nm1=n-1;
%...
%...  LEFT END POINT.  THE FOLLOWING CODING HAS BEEN FORMATTED
%     SO THAT THE NUMERICAL WEIGHTING COEFFICIENTS CAN BE MORE
%...  EASILY ASSOCIATED WITH THE BICKLEY MATRIX LISTED ABOVE)
      ux(1)=r2fdx*...
      (     -3.   *u(  1)     +4.   *u(  2)     -1.   *u(  3));
%...
%...  INTERIOR POINTS
      for i=2:nm1
      ux(i)=r2fdx*...
      (     -1.   *u(i-1)     +0.   *u(  i)     +1.   *u(i+1));
      end
```

```
%...
%...  RIGHT END POINT
      ux(n)=r2fdx*...
      (      1.   *u(n-2)     -4.   *u(n-1)     +3.   *u(  n));
```

The derivation of the FD approximations will be considered in the next section.

The preceding output for Case = 4 illustrates the first type of numerical distortion that is common in the numerical integration of first order hyperbolic PDEs, that is, *numerical oscillation.* In order to eliminate this oscillation, we consider two point upwind (noncentered) approximations, as implemented with ifd = 2. The output of the preceding program for ifd = 2 is listed below (and is plotted in htex1d2.ps)

```
 ncase =     1    h =  1.000e-03

   t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
 0.00     -2.00     25.00     25.00     0.000
 0.25     -1.75     89.93     89.85     0.081
 0.50     -1.50     98.65     98.63     0.022
 0.75     -1.25     99.82     99.81     0.004
 1.00     -1.00     99.98     99.97     0.001
 1.25     -0.75    100.00    100.00     0.000
 1.50     -0.50    100.00    100.00     0.000
 1.75     -0.25    100.00    100.00    -0.001
 2.00      0.00    100.00    100.00    -0.001
 2.25      0.25    100.00    100.00    -0.001
 2.50      0.50    100.00    100.00    -0.001
 2.75      0.75    100.00    100.00    -0.001
 3.00      1.00    100.00    100.00    -0.001
 3.25      1.25    100.00    100.00    -0.001
 3.50      1.50    100.00    100.00    -0.001
 3.75      1.75    100.00    100.00    -0.001
 4.00      2.00    100.00    100.00    -0.001
 4.25      2.25    100.00    100.00    -0.001
 4.50      2.50    100.00    100.00    -0.001
```

```
4.75      2.75    100.00    100.00    -0.001
5.00      3.00    100.00    100.00    -0.001


ncase =     2    h =  1.000e-03

  t      t-zl/v   T(zl,t)  Ta(zl,t)       diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     38.60     38.60     0.005
0.50     -1.50     49.73     49.73     0.008
0.75     -1.25     58.85     58.84     0.010
1.00     -1.00     66.30     66.30     0.003
1.25     -0.75     72.34     72.41    -0.073
1.50     -0.50     77.00     77.41    -0.414
1.75     -0.25     80.25     81.51    -1.256
2.00      0.00     82.21     84.86    -2.644
2.25      0.25     83.22     84.86    -1.635
2.50      0.50     83.66     84.86    -1.194
2.75      0.75     83.83     84.86    -1.026
3.00      1.00     83.89     84.86    -0.971
3.25      1.25     83.90     84.86    -0.955
3.50      1.50     83.91     84.86    -0.950
3.75      1.75     83.91     84.86    -0.949
4.00      2.00     83.91     84.86    -0.949
4.25      2.25     83.91     84.86    -0.949
4.50      2.50     83.91     84.86    -0.949
4.75      2.75     83.91     84.86    -0.949
5.00      3.00     83.91     84.86    -0.949


ncase =     3    h =  1.000e-03

  t      t-zl/v   T(zl,t)  Ta(zl,t)       diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     25.00     25.00     0.000
0.50     -1.50     25.00     25.00     0.000
0.75     -1.25     25.00     25.00     0.000
1.00     -1.00     25.00     25.00     0.000
```

```
1.25     -0.75     25.00     25.00     0.000
1.50     -0.50     25.00     25.00     0.000
1.75     -0.25     25.00     25.00     0.000
2.00      0.00     25.00     25.00     0.000
2.25      0.25     25.00     25.00     0.000
2.50      0.50     25.00     25.00     0.000
2.75      0.75     25.00     25.00     0.000
3.00      1.00     25.00     25.00     0.000
3.25      1.25     25.00     25.00     0.000
3.50      1.50     25.00     25.00     0.000
3.75      1.75     25.00     25.00     0.000
4.00      2.00     25.00     25.00     0.000
4.25      2.25     25.00     25.00     0.000
4.50      2.50     25.00     25.00     0.000
4.75      2.75     25.00     25.00     0.000
5.00      3.00     25.00     25.00     0.000


ncase =     4    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     25.00     25.00     0.000
0.50     -1.50     25.00     25.00     0.000
0.75     -1.25     25.01     25.00     0.008
1.00     -1.00     25.25     25.00     0.247
1.25     -0.75     27.24     25.00     2.243
1.50     -0.50     34.27     25.00     9.275
1.75     -0.25     47.86     25.00    22.861
2.00      0.00     64.76     25.00    39.764
2.25      0.25     79.80    100.00   -20.201
2.50      0.50     90.08    100.00    -9.924
2.75      0.75     95.76    100.00    -4.243
3.00      1.00     98.40    100.00    -1.604
3.25      1.25     99.46    100.00    -0.544
3.50      1.50     99.83    100.00    -0.168
3.75      1.75     99.95    100.00    -0.047
4.00      2.00     99.99    100.00    -0.012
```

```
 4.25      2.25    100.00    100.00    -0.003
 4.50      2.50    100.00    100.00    -0.001
 4.75      2.75    100.00    100.00     0.000
 5.00      3.00    100.00    100.00     0.000
```

We can note the following points about this output:

(1) The two point point FD approximations give good accuracy for Cases 1 to 3 which again are for relatively smooth problems. In particular, the FD solution is exact for Case 3 (the constant solution is differentiated exactly).

(2) The upwind FD approximations do not give an accurate solution for the discontinuous solution of Case 4. Although the previous numerical oscillation has been eliminated, there is now significant *numerical diffusion*, which is the second common form of numerical distortion in the solution of first order hyperbolic PDEs.

The library differentiator for the two point upwind FD approximations is listed below. Note that the direction of the flow is required to use upwinding (it is the sixth input argument of the routine, v). This is a consequence of the "flow of information" in the solution that is taken into account by the approximation, i.e., upwind points have a greater influence on the solution than downwind points. This characteristic indicates an important feature of hyperbolic PDEs, that is, they have a preferred direction; physically, this preferred direction in convective systems is the direction of flow.

Experience has indicated that some form of upwinding (to take into account the preferred direction) is essential for strongly hyperbolic systems. In other words, centered approximations for hyperbolic systems generally don't work (consider the ifd = 1 results discussed previously). In order to use upwinding, the direction of flow must be known (which may not be easy to determine in complex, multidimensional flow systems). Upwinding in the wrong direction ("downwinding") usually leads to unstable solutions.

```
      function [ux]=dss012(xl,xu,n,u,v)
%...
%...  FUNCTION DSS012 IS AN APPLICATION OF FIRST-ORDER DIRECTIONAL
```

```
%...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
%...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
%...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS WITH THE
%...  SIMPLEST FORM
%...
%...                          U  + V*U  = 0                        (1)
%...                           T      X
%...
%...  THE FIRST FOUR PARAMETERS, XL, XU, N AND U, ARE THE SAME AS
%...  FOR FUNCTION DSS002.  THE FIFTH PARAMETER, V, MUST BE PROVIDED
%...  TO DSS012 SO THAT THE DIRECTION OF FLOW IN EQUATION (1) CAN BE
%...  USED TO SELECT THE APPROPRIATE FINITE DIFFERENCE APPROXIMATION
%...  FOR THE FIRST-ORDER SPATIAL DERIVATIVE IN EQUATION (1), U .
%...  THE CONVENTION FOR THE SIGN OF V IS                      X
%...
%...     FLOW LEFT TO RIGHT              V GT 0
%...     (I.E., IN THE DIRECTION         (I.E., THE SIXTH ARGUMENT IS
%...     OF INCREASING X)                POSITIVE IN CALLING DSS012)
%...
%...     FLOW RIGHT TO LEFT              V LT 0
%...     (I.E., IN THE DIRECTION         (I.E., THE SIXTH ARGUMENT IS
%...     OF DECREASING X)                NEGATIVE IN CALLING DSS012)
%...
%...  COMPUTE THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFERENCE
%...  APPROXIMATION DEPENDING ON THE SIGN OF V IN EQUATION (1).
      dx=(xu-xl)/(n-1);
      if v > 0
%...
%...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
              ux(1)=(u(2)-u(1))/dx;
              for i=2:n
                 ux(i)=(u(i)-u(i-1))/dx;
              end
      end
%...
%...     (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
      if v < 0
              nm1=n-1;
```

```
          for i=1:nm1
             ux(i)=(u(i+1)-u(i))/dx;
          end
          ux(n)=(u(n)-u(n-1))/dx;
     end
```

The origin of the FD approximation is evident (it can also be considered a "stirred-tanks-in-series" approximation).

The preceding two outputs for Case = 4 illustrates the two types of numerical error that are common in the numerical integration of first order hyperbolic PDEs, that is, *numerical oscillation* and *numerical diffusion*. In order to reduce this error, we can consider combining the two approaches, i.e., using a biased upwind approximation (which is not fully centered or fully upwind, but a combination of the two). This is done by executing the preceding program with ifd = 3 for a five point biased upwind (5pbu) approximation coded in function dss020 listed subsequently (note that in calling the 5pbu routine, the direction of flow is required since the approximations use upwinding); the output is listed below (and is plotted in htex1d3.ps)

```
ncase =     1    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     89.93     89.85     0.081
0.50     -1.50     98.65     98.63     0.022
0.75     -1.25     99.82     99.81     0.004
1.00     -1.00     99.98     99.97     0.001
1.25     -0.75    100.00    100.00     0.000
1.50     -0.50    100.00    100.00     0.000
1.75     -0.25    100.00    100.00     0.000
2.00      0.00    100.00    100.00     0.000
2.25      0.25    100.00    100.00     0.000
2.50      0.50    100.00    100.00     0.000
2.75      0.75    100.00    100.00     0.000
3.00      1.00    100.00    100.00     0.000
3.25      1.25    100.00    100.00     0.000
```

```
3.50      1.50   100.00   100.00     0.000
3.75      1.75   100.00   100.00     0.000
4.00      2.00   100.00   100.00     0.000
4.25      2.25   100.00   100.00     0.000
4.50      2.50   100.00   100.00     0.000
4.75      2.75   100.00   100.00     0.000
5.00      3.00   100.00   100.00     0.000


ncase =     2    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     38.60     38.60     0.005
0.50     -1.50     49.73     49.73     0.008
0.75     -1.25     58.85     58.84     0.010
1.00     -1.00     66.31     66.30     0.007
1.25     -0.75     72.44     72.41     0.036
1.50     -0.50     77.34     77.41    -0.075
1.75     -0.25     81.65     81.51     0.142
2.00      0.00     84.58     84.86    -0.279
2.25      0.25     84.87     84.86     0.008
2.50      0.50     84.86     84.86     0.004
2.75      0.75     84.86     84.86     0.003
3.00      1.00     84.86     84.86    -0.003
3.25      1.25     84.86     84.86     0.002
3.50      1.50     84.86     84.86    -0.001
3.75      1.75     84.86     84.86     0.001
4.00      2.00     84.86     84.86     0.000
4.25      2.25     84.86     84.86     0.000
4.50      2.50     84.86     84.86     0.000
4.75      2.75     84.86     84.86     0.000
5.00      3.00     84.86     84.86     0.000


ncase =     3    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
```

```
0.00      -2.00      25.00      25.00      0.000
0.25      -1.75      25.00      25.00      0.000
0.50      -1.50      25.00      25.00      0.000
0.75      -1.25      25.00      25.00      0.000
1.00      -1.00      25.00      25.00      0.000
1.25      -0.75      25.00      25.00      0.000
1.50      -0.50      25.00      25.00      0.000
1.75      -0.25      25.00      25.00      0.000
2.00       0.00      25.00      25.00      0.000
2.25       0.25      25.00      25.00      0.000
2.50       0.50      25.00      25.00      0.000
2.75       0.75      25.00      25.00      0.000
3.00       1.00      25.00      25.00      0.000
3.25       1.25      25.00      25.00      0.000
3.50       1.50      25.00      25.00      0.000
3.75       1.75      25.00      25.00      0.000
4.00       2.00      25.00      25.00      0.000
4.25       2.25      25.00      25.00      0.000
4.50       2.50      25.00      25.00      0.000
4.75       2.75      25.00      25.00      0.000
5.00       3.00      25.00      25.00      0.000


ncase =     4    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00      -2.00      25.00      25.00      0.000
0.25      -1.75      25.00      25.00      0.000
0.50      -1.50      25.00      25.00     -0.003
0.75      -1.25      24.97      25.00     -0.031
1.00      -1.00      25.26      25.00      0.256
1.25      -0.75      24.19      25.00     -0.809
1.50      -0.50      28.03      25.00      3.026
1.75      -0.25      15.27      25.00     -9.732
2.00       0.00      67.21      25.00     42.208
2.25       0.25     103.99     100.00      3.991
2.50       0.50      97.68     100.00     -2.315
2.75       0.75     101.67     100.00      1.673
```

```
 3.00      1.00      99.05     100.00     -0.946
 3.25      1.25     100.51     100.00      0.514
 3.50      1.50      99.74     100.00     -0.260
 3.75      1.75     100.12     100.00      0.119
 4.00      2.00      99.95     100.00     -0.045
 4.25      2.25     100.01     100.00      0.010
 4.50      2.50     100.00     100.00      0.004
 4.75      2.75      99.99     100.00     -0.008
 5.00      3.00     100.01     100.00      0.008
```

We can note the following points about this output:

(1) The five point biased upwind (5pbu) FD approximations give good accuracy for Cases 1 to 3 which again are relatively smooth problems. In particular, the FD solution is exact for Case 3 (the constant solution is differentiated exactly).

(2) The 5pbu FD approximations give an improvement over centered approximations (ifd = 1) and low order upwinding (ifd = 2) by reducing both the numerical diffusion and oscillation, but these numerical errors still persist. This is a consequence of Godunov's theorem to be discussed subsequently. Briefly, we have probably done about as well as we can do with linear (FD) approximations, i.e., computing approximations to derivatives using linear combinations of the dependent variable, for example

```
      ux(  i)=r4fdx*...
      (  -1.     *u(i-3)...
         +6.     *u(i-2)...
        -18.     *u(i-1)...
        +10.     *u(i  )...
         +3.     *u(i+1));
```

We will therefore next turn to a nonlinear FD approximation as the final case (ifd = 4).

The library differentiator for the 5pbu FD approximations (ifd = 3) is listed below

```
      function [ux]=dss020(xl,xu,n,u,v)
%...
%...  SUBROUTINE DSS020 IS AN APPLICATION OF FOURTH-ORDER DIRECTIONAL
%...  DIFFERENCING IN THE NUMERICAL METHOD OF LINES.  IT IS INTENDED
%...  SPECIFICALLY FOR THE ANALYSIS OF CONVECTIVE SYSTEMS MODELLED BY
%...  FIRST-ORDER HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS AS DIS-
%...  CUSSED IN SUBROUTINE DSS012.  THE COEFFICIENTS OF THE FINITE
%...  DIFFERENCE APPROXIMATIONS USED HEREIN ARE TAKEN FROM BICKLEY, W.
%...  G., FORMULAE FOR NUMERICAL DIFFERENTIATION, THE MATHEMATICAL
%...  GAZETTE, PP. 19-27, 1941, N = 4, M = 1, P = 0, 1, 2, 3, 4.  THE
%...  IMPLEMENTATION IS THE **FIVE-POINT BIASED UPWIND FORMULA** OF
%...  M. B. CARVER AND H. W. HINDS, THE METHOD OF LINES AND THE
%...  ADVECTION EQUATION, SIMULATION, VOL. 31, NO. 2, PP. 59-69,
%...  AUGUST, 1978
%...
%...  COMPUTE THE COMMON FACTOR FOR EACH FINITE DIFFERENCE APPROXIMATION
%...  CONTAINING THE SPATIAL INCREMENT, THEN SELECT THE FINITE DIFFER-
%...  ENCE APPROXIMATION DEPENDING ON THE SIGN OF V (SIXTH ARGUMENT).
      dx=(xu-xl)/(n-1);
      r4fdx=1./(12.*dx);
%...
%...     (1)  FINITE DIFFERENCE APPROXIMATION FOR POSITIVE V
      if v > 0.
      ux(  1)=r4fdx*...
      ( -25.     *u(  1)...
        +48.     *u(  2)...
        -36.     *u(  3)...
        +16.     *u(  4)...
         -3.     *u(  5));
      ux(  2)=r4fdx*...
      (  -3.     *u(  1)...
        -10.     *u(  2)...
        +18.     *u(  3)...
         -6.     *u(  4)...
         +1.     *u(  5));
      ux(  3)=r4fdx*...
      (  +1.     *u(  1)...
          -8.     *u(  2)...
          +0.     *u(  3)...
          +8.     *u(  4)...
          -1.     *u(  5));
       nm1=n-1;
       for i=4:nm1
       ux(  i)=r4fdx*...
       (  -1.     *u(i-3)...
          +6.     *u(i-2)...
         -18.     *u(i-1)...
         +10.     *u(i  )...
          +3.     *u(i+1));
       end
       ux(  n)=r4fdx*...
       (  +3.     *u(n-4)...
         -16.     *u(n-3)...
         +36.     *u(n-2)...
         -48.     *u(n-1)...
         +25.     *u(n  ));
       end
%...
%...   (2)  FINITE DIFFERENCE APPROXIMATION FOR NEGATIVE V
       if v < 0.
       ux(  1)=r4fdx*...
       ( -25.     *u(  1)...
         +48.     *u(  2)...
         -36.     *u(  3)...
         +16.     *u(  4)...
          -3.     *u(  5));
       nm3=n-3;
       for i=2:nm3
       ux(  i)=r4fdx*...
       (  -3.     *u(i-1)...
         -10.     *u(i  )...
         +18.     *u(i+1)...
          -6.     *u(i+2)...
          +1.     *u(i+3));
       end
```

```
ux(n-2)=r4fdx*...
(  +1.     *u(n-4)...
   -8.     *u(n-3)...
   +0.     *u(n-2)...
   +8.     *u(n-1)...
   -1.     *u(n  ));
ux(n-1)=r4fdx*...
(  -1.     *u(n-4)...
   +6.     *u(n-3)...
  -18.     *u(n-2)...
  +10.     *u(n-1)...
   +3.     *u(n  ));
ux(  n)=r4fdx*...
(  +3.     *u(n-4)...
  -16.     *u(n-3)...
  +36.     *u(n-2)...
  -48.     *u(n-1)...
  +25.     *u(n  ));
end
```

The 5pbu FD approximations are derived from the Taylor series as discussed in the next section.

To conclude this discussion of time and spatial discretization, we consider the use of the van Leer flux limiter to compute spatial derivatives (by setting ifd = 4 in the preceding program). The objective is to eliminate numerical diffusion and oscillation as much as possible (in the solution of first order, hyperbolic PDEs). The output is first listed (and plotted in htex1d4.ps)

```
ncase =     1    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     89.93     89.85     0.081
0.50     -1.50     98.65     98.63     0.022
0.75     -1.25     99.82     99.81     0.004
1.00     -1.00     99.98     99.97     0.001
```

```
1.25     -0.75    100.00    100.00     0.000
1.50     -0.50    100.00    100.00     0.000
1.75     -0.25    100.00    100.00     0.000
2.00      0.00    100.00    100.00     0.000
2.25      0.25    100.00    100.00     0.000
2.50      0.50    100.00    100.00     0.000
2.75      0.75    100.00    100.00     0.000
3.00      1.00    100.00    100.00     0.000
3.25      1.25    100.00    100.00     0.000
3.50      1.50    100.00    100.00     0.000
3.75      1.75    100.00    100.00     0.000
4.00      2.00    100.00    100.00     0.000
4.25      2.25    100.00    100.00     0.000
4.50      2.50    100.00    100.00     0.000
4.75      2.75    100.00    100.00     0.000
5.00      3.00    100.00    100.00     0.000


ncase =     2    h =  1.000e-03

  t     t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00     -2.00     25.00     25.00     0.000
0.25     -1.75     38.60     38.60     0.005
0.50     -1.50     49.73     49.73     0.008
0.75     -1.25     58.85     58.84     0.010
1.00     -1.00     66.31     66.30     0.011
1.25     -0.75     72.42     72.41     0.011
1.50     -0.50     77.42     77.41     0.011
1.75     -0.25     81.51     81.51     0.003
2.00      0.00     84.23     84.86    -0.628
2.25      0.25     84.79     84.86    -0.073
2.50      0.50     84.77     84.86    -0.087
2.75      0.75     84.77     84.86    -0.088
3.00      1.00     84.77     84.86    -0.087
3.25      1.25     84.77     84.86    -0.087
3.50      1.50     84.77     84.86    -0.087
3.75      1.75     84.77     84.86    -0.087
4.00      2.00     84.77     84.86    -0.087
```

```
4.25       2.25     84.77      84.86    -0.087
4.50       2.50     84.77      84.86    -0.087
4.75       2.75     84.77      84.86    -0.087
5.00       3.00     84.77      84.86    -0.087


ncase =     3    h =  1.000e-03

  t      t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00      -2.00     25.00      25.00     0.000
0.25      -1.75     25.00      25.00     0.000
0.50      -1.50     25.00      25.00     0.000
0.75      -1.25     25.00      25.00     0.000
1.00      -1.00     25.00      25.00     0.000
1.25      -0.75     25.00      25.00     0.000
1.50      -0.50     25.00      25.00     0.000
1.75      -0.25     25.00      25.00     0.000
2.00       0.00     25.00      25.00     0.000
2.25       0.25     25.00      25.00     0.000
2.50       0.50     25.00      25.00     0.000
2.75       0.75     25.00      25.00     0.000
3.00       1.00     25.00      25.00     0.000
3.25       1.25     25.00      25.00     0.000
3.50       1.50     25.00      25.00     0.000
3.75       1.75     25.00      25.00     0.000
4.00       2.00     25.00      25.00     0.000
4.25       2.25     25.00      25.00     0.000
4.50       2.50     25.00      25.00     0.000
4.75       2.75     25.00      25.00     0.000
5.00       3.00     25.00      25.00     0.000


ncase =     4    h =  1.000e-03

  t      t-zl/v   T(zl,t)  Ta(zl,t)      diff
0.00      -2.00     25.00      25.00     0.000
0.25      -1.75     25.00      25.00     0.000
0.50      -1.50     25.00      25.00     0.000
```

```
 0.75      -1.25      25.00      25.00      0.000
 1.00      -1.00      25.00      25.00      0.000
 1.25      -0.75      25.00      25.00      0.000
 1.50      -0.50      25.01      25.00      0.008
 1.75      -0.25      31.29      25.00      6.286
 2.00       0.00      66.42      25.00     41.423
 2.25       0.25      91.65     100.00     -8.351
 2.50       0.50      98.61     100.00     -1.390
 2.75       0.75      99.82     100.00     -0.182
 3.00       1.00      99.98     100.00     -0.021
 3.25       1.25     100.00     100.00     -0.002
 3.50       1.50     100.00     100.00      0.000
 3.75       1.75     100.00     100.00      0.000
 4.00       2.00     100.00     100.00      0.000
 4.25       2.25     100.00     100.00      0.000
 4.50       2.50     100.00     100.00      0.000
 4.75       2.75     100.00     100.00      0.000
 5.00       3.00     100.00     100.00      0.000
```

We can note the following points about this output:

(1) The van Leer limited gives good accuracy for Cases 1 to 3 which again are relatively smooth problems. In particular, the van Leer limiter is exact for Case 3 (the constant solution is differentiated exactly).

(2) The van Leer limiter gives an improvement over centered approximations (ifd = 1), low order upwinding (ifd = 2) and higher order biased upwinding (ifd = 3) by reducing further the numerical diffusion and eliminating the oscillation. Clearly, flux limiters are effective, and their performance can be improved by going to higher order limiters (e.g., van Leer is second order; Superbee is third order).

The library differentiator for the van Leer limiter (ifd = 4) is listed below (this routine is based on a Fortran routine first coded by WES and improved by Alain vande Wouwer and Philippe Saucez)

```
      function [ux]=vanl2(xl,xu,n,u,v)
```

```
dx=(xu-xl)/(n-1);
delta=1.0e-05;
if v >= 0.0
  for i=3:n-1
    if(abs(u(i)-u(i-1))<delta)
      phi(i)=0.0;
    else
      r(i)=(u(i+1)-u(i))/(u(i)-u(i-1));
      phi(i)=(r(i)+abs(r(i)))/(1.0+abs(r(i)));
    end
    if(abs(u(i-1)-u(i-2))<delta)
      phi(i-1)=0.0;
    else
      r(i-1)=(u(i)-u(i-1))/(u(i-1)-u(i-2));
      phi(i-1)=(r(i-1)+abs(r(i-1)))/(1.0+abs(r(i-1)));
    end
    flux2=u(i  )+(u(i  )-u(i-1))*phi(i  )/2.0;
    flux1=u(i-1)+(u(i-1)-u(i-2))*phi(i-1)/2.0;
    ux(i)=(flux2-flux1)/dx;
  end
  ux(1)=(-u(1)+u(2))/dx;
  ux(2)=(-u(1)+u(2))/dx;
  ux(n)=(u(n)-u(n-1))/dx;
end
if v < 0.0
  for i=2:n-2
    if(abs(u(i)-u(i+1))<delta)
      phi(i)=0.0;
    else
      r(i)=(u(i-1)-u(i))/(u(i)-u(i+1));
      phi(i)=(r(i)+abs(r(i)))/(1.0+abs(r(i)));
    end
    if(abs(u(i+1)-u(i+2))<delta)
      phi(i+1)=0.0;
    else
      r(i+1)=(u(i)-u(i+1))/(u(i+1)-u(i+2));
      phi(i+1)=(r(i+1)+abs(r(i+1)))/(1.0+abs(r(i+1)));
    end
    flux2=u(i  )+(u(i  )-u(i+1))*phi(i  )/2.0;
    flux1=u(i+1)+(u(i+1)-u(i+2))*phi(i+1)/2.0;
    ux(i)=(flux2-flux1)/dx;
  end
ux(1)=(u(2)-u(1))/dx;
ux(n-1)=(-u(n-1)+u(n))/dx;
ux(n  )=(-u(n-1)+u(n))/dx;
end
```

The van Leer limiter has two basic sections for positive and negative velocities and therefore the direction of flow is required as an input. Within each case (positive or negative velocity),

- van Leer computes a ratio at grid point i, r(i), that is a measure of the change in the slope of the solution
- A nonlinear limiting function, phi(i), is then applied to r(i)
- From that limited slope, the "flux" is computed at two neighboring grid points
- The difference in the two fluxes is then the numerical approximation of the spatial derivative, ux(i), that is returned from the function to be used in the PDE
- For phi(i) = 0, van Leer reduces to two point upwinding
- More generally, van Leer is a form of nonlinear, two point upwinding as evident when expressed in the following way (from M. Berzins). For

  the basic hyperbolic equation written in conservative form

  $$u_t + F(u)_x = 0$$

  the convective term can be approximated as:

$$F(u)_x = \left[1 + \frac{B(r_j)}{2} - \frac{B(r_{j-1})}{2r_{j-1}}\right] \frac{u_j - u_{j-1}}{\Delta x}$$

where

$$B(r_j) = \frac{r_j + |r_j|}{1 + r_j}, r_j = \frac{u_{j+1} - u_j}{u_j - u_{j-1}}$$

Note that when $r_j = 0, B(r_j) = 0$, and we have just two point upwinding (ifd = 2). The nonlinearity of $B(r_j)$ is clear.

To conclude this section, we have considered strongly convective systems that have as the basic differential group

$$u_t + F(u)_x$$

or for a constant velocity system

$$u_t + vu_x$$

If we solve the linear advection equation

$$u_t + vu_x = 0 \tag{1}$$

subject to the initial and boundary conditions

$$u(x,0) = f(x), u(0,t) = g(t)$$

and if the functions $f(x)$ and $g(t)$ introduce a discontinuity, e.g.,

$$u(x,0) = 0, u(0,t) = 1 \qquad (2)(3)$$

we have a form of the *Riemann problem.*

The solution to eqs. (1), (2) and (3) is

$$u(x,t) = h(x - vt) \tag{4}$$

where $h(t)$ is the Heaviside function

$$\begin{aligned} h(t) &= 0, t < 0 \\ h(t) &= 1, t > 0 \end{aligned}$$

In a sense, eq (4) indicates that this Riemann problem is impossible to solve numerically since at $x - vt = 0$, the derivative u_x is infinite. Thus, about the best we can do is to use an *approximate Riemann solver*, and the preceding discussion indicates that a flux limiter is a good choice (at least as good as we have right now). In other words, we will generally not be able to solve problems exactly that have moving discontinuites, e.g., eq. (4).

On the other hand, we might argue that physically, there are few problems that have this characteristic; some form of smoothing (e.g., diffusion) is generally going to occur that will reduce the numerical severity of the problem. However, problems with steep moving fronts occur in many important applications such as adsorption and chromatography. Therefore, the continuing development of effective numerical methods for strongly convective (hyperbolic) PDEs is an active area of research, particularly the testing of algorithms, e.g., van Leer, Superbee. For example:

- Toro, E. F., *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Springer, Berlin, 1st ed., (1997) has 387 references to the CFD literature (2nd ed. appeared in 1999)
- Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol I: *Fundamentals of Numerical Discretizations*, Wiley, 1988; Vol II: *Computational Methods for Inviscid and Viscous Flows*, Wiley, 1990

Finally, linear approximations can be effective if the problem is sufficiently smooth, but they are limited by Godunov's theorem which states (loosely):

There are no linear approximations to the Riemann problem, higher than first order, that are nonoscillatory. The preceding heat exchanger example demonstrated this conclusion.

The files pertaining to this heat exchanger example are in:

http://www.lehigh.edu/~ wes1/apci/htex1.ppt
http://www.lehigh.edu/~ wes1/apci/htex1.m
http://www.lehigh.edu/~ wes1/apci/dss002.m
http://www.lehigh.edu/~ wes1/apci/dss012.m
http://www.lehigh.edu/~ wes1/apci/dss020.m
http://www.lehigh.edu/~ wes1/apci/vanl2.m

Method of Lines

The preceding heat exchanger example illustrates a basic approach to the numerical integration of PDEs generally termed the *method of lines* (MOL). This is essentially a two-step process as the heat exchanger example illustrated:

- The spatial (boundary value) derivatives are approximated algebraically, for example, by finite differences or a flux limiter, on a spatial grid
- The resulting system of ODEs in the initial value variable (typically time) is then integrated by an initial value ODE integrator (e.g., Euler's method, the modified Euler method, RKF45, BDF)

The advantages of this approach are:

- The temporal and spatial integrations are separated in the sense that they can be treated separately in the coding; this adds a very attractive degree of flexibility, including the use of library routines for the temporal and spatial integrations
- We can build on the new developments in ODE integrators to do PDEs

- All of the major classes of PDEs can be accommodated (elliptic, hyperbolic, parabolic). Elliptic problems will generally require the addition of a pseudo initial value derivative that is then integrated to effectively zero (a steady state or equilibrium condition); this is offer easier than attacking the elliptic problem directly

- Systems of ODEs, DAEs and PDEs (in one, two and three spatial dimensions plus time), linear and nonlinear, can be naturally accommodated within the MOL; for example, ODEs or DAEs can serve as boundary conditions for PDEs

The MOL is not a single method of solution of ODE/DAE/PDE systems. Rather, it is a flexible, open-ended approach that is limited only by the ingenuity of the analyst.

To illustrate the MOL analysis of a PDE system, we return to the problem considered in the first lecture (28jan00.tex, 28jan00.ps, 28jan00.pdf): A tubular reactor, as depicted in the accompanying diagram (see file reactor.ppt), has a significant heat effect so that cooling at the wall is being considered to remove the heat of reaction.

Consequently, significant radial heat and concentration profiles develop. The reactor is therefore modeled in terms of radial position, axial position and time; since there are three independent variables, we will use PDEs.

Our objective in this analysis is to determine what comes out of the reactor, i.e., the average concentrations and temperature, so that we can determine if it achieves the required operating performance. To do this, we construct a mathematical model.

The material balance for an incremental element of length Δz is (see accompanying diagram)

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

$$+2\pi r\Delta r v c_a|_{z-\Delta z} - 2\pi r\Delta r v c_a|_z$$

$$-2\pi r\Delta r\Delta z k_r c_a^2$$

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

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

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

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

If we now assume Fick's first law for the flux with a constant diffusivity, D

$$q_m = -D\frac{\partial c_a}{\partial r}$$

we obtain

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

or

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

Equation (1) is the required material balance for $c_a(r,z,t)$.

The energy balance for the incremental section is

$$2\pi r\Delta r\Delta z\rho C_p\frac{\partial T}{\partial t} = 2\pi r\Delta z q_h|_r - 2\pi(r+\Delta r)\Delta z q_h|_{r+\Delta r}$$

$$+2\pi r\Delta r v\rho C_p T|_{z-\Delta z} - 2\pi r\Delta r v\rho C_p T|_z$$

$$-\Delta H 2\pi r\Delta r\Delta z k_r c_a^2$$

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

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

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

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

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

$$q_h = -k\frac{\partial T}{\partial r}$$

we obtain

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C_p r}\frac{\partial(r\frac{\partial T}{\partial r})}{\partial r} - v\frac{\partial T}{\partial z} - \frac{\Delta H k_r}{\rho C_p}c_a^2$$

or

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C_p}(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}) - v\frac{\partial T}{\partial z} - \frac{\Delta H k_r}{\rho C_p}c_a^2 \qquad (2)$$

Eq. (2) is the required energy balance for $T(r,z,t)$.

The reaction rate constant, k_r, is given by

$$k_r = k_0 e^{-E/(RT)} \qquad (3)$$

The variables and parameters of eqs. (1), (2) and (3) are summarized in the following table (in cgs units)

Reactant concentration	c_a	Solution to eq. (1)
Temperature	T	Solution to eq. (2)
Reaction rate constant	k_r	From eq. (3)
Time	t	
Radial position	r	
Axial position	z	
Entering concentration	c_{a0}	0.01
Entering temperature	T_0	305
Wall temperature	T_w	$305, 355$
Reactor radius	r_0	2
Reactor length	z_l	100
Linear velocity	v	1.0
Mass diffusivity	D	0.1
Thermal diffusivity	$k/(\rho C_p)$	0.1
Liquid density	ρ	1.0
Liquid specific heat	C_p	0.5
Heat of reaction	ΔH	$-10,000$
Specific rate constant	k_0	1.5×10^9
Activiation energy	E	$15,000$
Gas constant	R	1.987

Eq. (1) requires one initial condition (IC), two boundary conditions (BCs) in r and one BC in z

$$c_a(r, z, 0) = 0 \tag{4}$$

$$\frac{\partial c_a(0, z, t)}{\partial r} = 0, \frac{\partial c_a(r_0, z, t)}{\partial r} = 0 \qquad (5)(6)$$

$$c_a(r, 0, t) = c_{a0} \tag{7}$$

Eq. (2) also requires one IC, two BCs in r and one BC in z

$$T(r,z,0) = T_0 \tag{8}$$

$$\frac{\partial T(0,z,t)}{\partial r} = 0, T(r_0,z,t) = T_w \tag{9)(10}$$

$$T(r,0,t) = T_0 \tag{11}$$

The solution to this problem (and in general, to problems in ODEs and PDEs) are the dependent variables ($c_a(r,z,t)$,$T(r,z,t)$) as a function of the independent variables (r,z,t).

Since we are intertested in the exiting conditions, we will compute $c_a(r,z_l,t), T(r,z_l,t)$. Also, there may be significant radial profiles at the exit, so we will also compute the integrals

$$c_{a,avg}(t) = \int_0^{r_0} 2\pi r c_a(r,z_l,t)dr/(\pi r_0^2) \tag{12}$$

$$T_{avg}(t) = \int_0^{r_0} 2\pi r T(r,z_l,t)dr/(\pi r_0^2) \tag{13}$$

Note that this problem includes 2-D PDEs, transcendental equations and integrals. Because of its complexity (number of equations and their non-linearity), an analytical solution is precluded, and we must use numerical methods.

A MOL code for the solution of the preceding problem is listed below

```
% Dynamic analysis of a plug flow reactor
%
% The following equations model a plug flow reactor (PFR)
%
```

```
%   Material balance
%
%   ca  = -v*ca  + D*(ca   + (1/r)ca ) - k*ca^2                 (1)
%     t        z        rr          r
%
%   Energy balance
%
%   T  = -v*T  + k*/(rho*Cp)*(T   + (1/r)*T )                   (2)
%    t       z                 rr          r
%
%        - (dH*k/(rho*Cp)*ca^2
%
%   kr = k0*exp(-E/(R*T))                                       (3)
%
% The variables and parameters for this model are (in cgs units)
%
%   Reactant concentration          ca        (eq. (1))
%
%   Temperature                     T         (eq. (3))
%
%   Reaction rate constant          kr        (eq. (3))
%
%   Time                            t
%
%   Radial position                 r
%
%   Axial position                  z
%
%   Entering concentration          ca0          0.01
%
%   Entering temperature            T0           305
%
%   Wall temperature                TW           305
%                                                355
%
%   Reactor radius                  r0           2
%
%   Reactor length                  zl           100
```

```
%
%   Linear velocity                v            1
%
%   Mass diffusivity               D            0.1
%
%   Thermal diffusivity     alpha = k/(rho*Cp)   0.1
%
%   Liquid density                 rho          1.0
%
%   Liquid specific heat           Cp           0.5
%
%   Heat of reaction               dH          -10,000
%
%   Specific rate constant         k0           1.5e+09
%                                               2.0e+09
%
%   Activation energy              E            15,000
%
%   Gas constant                   R            1.987
%
% Note that there are two cases, corresponding to the two
% values of the specific rate constant, k0.  This is a
% sensitive parameter since it multiplies a stongly nonlinear
% term (Arrhenius temperature dependency) in eq. (3).  Thus,
% two values are programmed k0 = 1.5e+09, 2.0e+09, to indicate
% the effect of k0.
%
% The solution to this system, ca(r,z,t) (from eq. (1)) and
% T(r,z,t) (from eq. (2)) is computed by a fixed step Euler
% integration in t.  Also, the choice of an Euler step is
% sensitive because of the nonlinearity of eq. (3).
%
% Open output file
  fid1=fopen('reactor.out','w');
%
% Model parameters (defined in the comments above)
  ca0=0.01;
  T0=305.0;
```

```
%
% Case 1: Cooled reactor wall
% Tw=305.0;
%
% Case 2: Heated reactor wall
  Tw=355.0;
  r0=2.0;
  zl=100.0;
  v=1.0;
  D=0.1;
  alpha=0.1;
  rho=1.0;
  Cp=0.5;
  dH=-10000.0;
  E=15000.0;
  R=1.987;
%
% Grid in axial direction
  nz=21;
  dz=zl/(nz-1);
  for j=1:nz
    z(j)=(j-1)*dz;
  end
%
% Grid in radial direction
  nr=5;
  dr=r0/(nr-1);
  for i=1:nr
    r(i)=(i-1)*dr;
  end
  drs=dr^2;
%
% Case 1: Low reaction rate constant
% rk0=1.5e+09;
%
% Case 2: High reaction rate constant
  rk0=2.0e+09;
%
```

```
% Series of solutions for different Euler steps
  for ncase=1:1
%
% Initial, final times
  t=0.0; tf=300.0;
%
% Integration step, number of Euler steps for each output
  if ncase==1 h=0.1; nout=100; end
%
% Initial conditions
  for j=1:nz
  for i=1:nr
    ca(i,j)=0.0;
     T(i,j)=T0;
  end
  end
%
% Write h
  fprintf(fid1,'\n\n ncase = %5d    h = %10.3e\n\n',ncase,h);
%
% Write heading
  fprintf(fid1,'     t   ca(r,zl,t)  T(r,zl,t)\n');
%
% Integrate until t = tf
  while t < tf*1.0001
%
% Monitor solution by displaying t
  t
%
% Write selected output (a quadrature, e.g., Simpson's
% rule, could be added here for the average output ca, T)
  for i=1:nr
    fprintf(fid1,'%6.1f%12.6f%10.2f\n',t,ca(i,nz),T(i,nz));
  end
  fprintf(fid1,'\n');
%
% Take nout Euler steps
  for iout=1:nout
```

```
%
%   Temporal derivatives
%
%     Cover the nr x nz grid
%
%       Entering conditions (z = 0)
        for i=1:nr
          ca(i,1)=ca0;
          cat(i,1)=0.0;
          T(i,1)=T0;
          Tt(i,1)=0.0;
        end
%
%       Rest of reactor
        for j=2:nz
%
%         Centerline (r = 0)
          rk=rk0*exp(-E/(R*T(1,j)));
          cat(1,j)=-v*(ca(1,j)-ca(1,j-1))/dz...
                  +4.0*D*(ca(2,j)-ca(1,j))/drs...
                  -rk*ca(1,j)^2;
          Tt(1,j)=-v*(T(1,j)-T(1,j-1))/dz...
                  +4.0*alpha*(T(2,j)-T(1,j))/drs...
                  -dH*rk/(rho*Cp)*ca(1,j)^2;
%
%         Wall (r = r0)
          rk=rk0*exp(-E/(R*Tw));
          cat(nr,j)=-v*(ca(nr,j)-ca(nr,j-1))/dz...
                   +2.0*D*(ca(nr-1,j)-ca(nr,j))/drs...
                   -rk*ca(nr,j)^2;
           T(nr,j)=Tw;
          Tt(nr,j)=0.0;
%
%         Interior, r ~= 0 and r ~= r0
          for i=2:nr-1
            rk=rk0*exp(-E/(R*T(i,j)));
            cat(i,j)=-v*(ca(i,j)-ca(i,j-1))/dz...
                   +D*(ca(i+1,j)-2.0*ca(i,j)+ca(i-1,j))/drs...
                   +D*(1.0/r(i)*(ca(i+1)-ca(i-1)))/(2.0*dr)...
                   -rk*ca(i,j)^2;
            Tt(i,j)=-v*(T(i,j)-T(i,j-1))/dz...
                   +alpha*(T(i+1,j)-2.0*T(i,j)+T(i-1,j))/drs...
                   +alpha*(1.0/r(i)*(T(i+1,j)-T(i-1,j)))/(2.0*dr)...
                   -dH*rk/(rho*Cp)*ca(i,j)^2;
%
%         Next r
          end
%
%       Next z
        end
%
%   All temporal derivatives are computed
%
%   Take Euler step
    for j=1:nz
    for i=1:nr
      ca(i,j)=ca(i,j)+cat(i,j)*h;
       T(i,j)= T(i,j)+ Tt(i,j)*h;
    end
    end
    t=t+h;
%
% Next Euler step
  end
%
% Next output
  end
%
% Next case
  end
```

We can note the following points about this program:

(1) The model parameters are first defined, including the variation in the

wall temperature, Tw (initially, consideration was given to cooling the wall by keeping it at the initial temperature Tw = 305, but execution of the program indicated that very little reaction occurred; thus, the wall in the second case is actually heated to Tw = 355 to produce a significant reaction as reflected in the consumption of the reactant, i.e., reduction of the entering concentration $c_a(r, 0, t) = 0.01$).

```
% Model parameters (defined in the comments above)
  ca0=0.01;
  T0=305.0;
%
% Case 1: Cooled reactor wall
% Tw=305.0;
%
% Case 2: Heated reactor wall
  Tw=355.0;
  r0=2.0;
  zl=100.0;
  v=1.0;
  D=0.1;
  alpha=0.1;
  rho=1.0;
  Cp=0.5;
  dH=-10000.0;
  E=15000.0;
  R=1.987;
```

(2) The spatial grids in z and r are then defined numerically over the intervals $0 \le z \le z_l$ and $0 \le r \le r_0$ (with nz = 21 and nr = 5 points, respectively)

```
% Grid in axial direction
  nz=21;
  dz=zl/(nz-1);
  for j=1:nz
    z(j)=(j-1)*dz;
  end
```

```
%
% Grid in radial direction
  nr=5;
  dr=r0/(nr-1);
  for i=1:nr
    r(i)=(i-1)*dr;
  end
  drs=dr^2;
```

(3) Variation of the specific rate constant, k is also included (this can be considered as a type of parameter sensitivity analysis, perhaps to compare the model output with experimental data)

```
% Case 1: Low reaction rate constant
% rk0=1.5e+09;
%
% Case 2: High reaction rate constant
  rk0=2.0e+09;
```

(4) The code provides for multiple runs, but only one run is coded. Within each run, the parameters controlling the integration in t are set, including the initial and final values of t, the euler integration step, h, and the number of Euler steps between outputs, nout:

```
% Series of solutions for different Euler steps
  for ncase=1:1
%
% Initial, final times
  t=0.0; tf=300.0;
%
% Integration step, number of Euler steps for each output
  if ncase==1 h=0.1; nout=100; end
```

(5) The initial conditions for the two dependent variables, c_a (eq. (5)) and T (eq. (8), are set

```
% Initial conditions
  for j=1:nz
  for i=1:nr
    ca(i,j)=0.0;
     T(i,j)=T0;
  end
  end
```

Note that there are nz x nr = 21 x 5 = 105 ODEs in the MOL approximation of the two PDEs, eqs. (1) and (2)

(6) Headings are printed for the output

```
% Write h
  fprintf(fid1,'\n\n ncase = %5d    h = %10.3e\n\n',ncase,h);
%
% Write heading
  fprintf(fid1,'     t   ca(r,zl,t)  T(r,zl,t)\n');
```

(7) The integration in t starts with a test if t has reached the final value, tf

```
% Integrate until t = tf
  while t < tf*1.0001
%
% Monitor solution by displaying t
  t
```

(8) The solution is printed (including the initial conditions in the first pass through this code)

```
% Write selected output (a quadrature, e.g., Simpson's
% rule, could be added here for the average output ca, T)
  for i=1:nr
```

```
    fprintf(fid1,'%6.1f%12.6f%10.2f\n',t,ca(i,nz),T(i,nz));
  end
  fprintf(fid1,'\n');
```

If the average exiting concentration and temperature, according to eqs. (12) and (13), are to be included in the output, a quadrature calculation, e.g., Simpson's rule, could be included at this point (the radial grid has an odd number of points, nr = 5, as required by Simpson's rule)

(9) nout Euler steps are taken (between outputs), starting with the definition of the boundary conditions

```
% Take nout Euler steps
  for iout=1:nout
%
%   Temporal derivatives
%
%     Cover the nr x nz grid
%
%       Entering conditions (z = 0)
        for i=1:nr
          ca(i,1)=ca0;
          cat(i,1)=0.0;
          T(i,1)=T0;
          Tt(i,1)=0.0;
        end
```

Note that the entering conditions, $c_a(r, 0, t)$ and $T(r, 0, t)$ are set according to BCs (7) and (11). Since these entering conditions are constant, their time derivatives are set to zero. All four of these conditions are specified with the second index set to one which is the first grid point in z (the entrance to the reactor).

(10) The remaining points in z, $2 \leq j \leq nz (= 21)$ are then covered. The calculations for $r = 0$ (with the first index set to one) are done first

```
%
%       Rest of reactor
        for j=2:nz
%
%         Centerline (r = 0)
          rk=rk0*exp(-E/(R*T(1,j)));
          cat(1,j)=-v*(ca(1,j)-ca(1,j-1))/dz...
                  +4.0*D*(ca(2,j)-ca(1,j))/drs...
                  -rk*ca(1,j)^2;
          Tt(1,j)=-v*(T(1,j)-T(1,j-1))/dz...
                  +4.0*alpha*(T(2,j)-T(1,j))/drs...
                  -dH*rk/(rho*Cp)*ca(1,j)^2;
```

The Arrhenius temperature dependency is first calculated (note the use of the centerline temperature, $T(0, z, t)$)

```
          rk=rk0*exp(-E/(R*T(1,j)));
```

The temporal derivative from eq. (1)

$$\frac{\partial c_a}{\partial t}$$

is then calculated

```
          cat(1,j)=-v*(ca(1,j)-ca(1,j-1))/dz...
                  +4.0*D*(ca(2,j)-ca(1,j))/drs...
                  -rk*ca(1,j)^2;
```

We can note the following points in this coding of the temporal derivative:

(10.1) The convective derivative in eq. (1)

$$-v\frac{\partial c_a}{\partial z}$$

is approximated by a two point upwind FD

```
-v*(ca(1,j)-ca(1,j-1))/dz
```

The calculation of $\frac{\partial c_a}{\partial z}$ could also have been done by a call to function dss012 as illustrated by the preceding heat exchanger example. Note also that since the index in z starts at j = 2, the value of ca(1,j-1) is defined (by BC (7), discussed in (9) above).

(10.2) The diffusive derivatives in eq. (1)

$$D(\frac{\partial^2 c_a}{\partial r^2} + \frac{1}{r}\frac{\partial c_a}{\partial r})$$

are approximated by three point, centered FDs, including BC (5)

```
+4.0*D*(ca(2,j)-ca(1,j))/drs
```

There is a complication at $r = 0$ that was taken into account in the preceding coding. The second term at $r = 0$ is indetermine as a consquence of BC (5)

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

If we apply l'Hospital's rule to the indeterminant form by differentiating the numerator and denominator with respect to r

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

Thus, at $r = 0$, the diffusion group becomes

$$D(\frac{\partial^2 c_a}{\partial r^2} + \frac{1}{r}\frac{\partial c_a}{\partial r}) = 2D\frac{\partial^2 c_A}{\partial r^2}$$

with the FD approximation

$$2D\frac{\partial^2 c_a}{\partial r^2} \cong 2D\frac{c_a(2) - 2c_a(1) + c_a(0)}{\Delta r^2}$$

We then eliminate the fictitious value $c_a(0)$ using BC (5)

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

or

$$c_a(0) = c_a(2)$$

The final result for the diffusion group at $r = 0$ is then

$$D(\frac{\partial^2 c_a}{\partial r^2} + \frac{1}{r}\frac{\partial c_a}{\partial r}) \cong 2D\frac{c_a(2) - 2c_a(1) + c_a(0)}{\Delta r^2} \cong 4D\frac{c_a(2) - c_a(1)}{\Delta r^2}$$

which is programmed as indicated above.

(10.3) The reaction term

$$-k_r c_a^2$$

is programmed as

```
-rk*ca(1,j)^2;
```

Note the ease with which the nonlinear, second order reaction term can be programmed (illustrating the power of numerical methods, i.e., nonlinearities are easily included in the analysis).

(10.4) The temporal derivative of temperature from eq. (2),

$$\frac{\partial T}{\partial t}$$

is programmed in the same way

```
          Tt(1,j)=-v*(T(1,j)-T(1,j-1))/dz...
                  +4.0*alpha*(T(2,j)-T(1,j))/drs...
                  -dH*rk/(rho*Cp)*ca(1,j)^2;
```

Again, the ease of programming the nonlinear Arrhenius temperature dependency in the reaction term (discussed above)

```
          rk=rk0*exp(-E/(R*T(1,j)));
```

is apparent.

(11) The temporal derivatives at the wall are then programmed

```
%         Wall (r = r0)
          rk=rk0*exp(-E/(R*Tw));
          cat(nr,j)=-v*(ca(nr,j)-ca(nr,j-1))/dz...
                   +2.0*D*(ca(nr-1,j)-ca(nr,j))/drs...
                   -rk*ca(nr,j)^2;
           T(nr,j)=Tw;
          Tt(nr,j)=0.0;
```

The Arrhenius temperature is first programmed using the wall temperature, Tw

```
          rk=rk0*exp(-E/(R*Tw));
```

The temporal derivative from eq. (1)

$$\frac{\partial c_a}{\partial t}$$

is then calculated

```
cat(nr,j)=-v*(ca(nr,j)-ca(nr,j-1))/dz...
          +2.0*D*(ca(nr-1,j)-ca(nr,j))/drs...
          -rk*ca(nr,j)^2;
```

We can note the following points in this coding of the temporal derivative:

(11.1) The convective derivative in eq. (1)

$$-v\frac{\partial c_a}{\partial z}$$

is again approximated by a two point upwind FD

```
-v*(ca(nr,j)-ca(nr,j-1))/dz
```

Note the use of the subscript nr since we are now considering eq. (1) at the wall ($r = r_0$).

(11.2) The diffusive derivatives in eq. (1)

$$D(\frac{\partial^2 c_a}{\partial r^2} + \frac{1}{r}\frac{\partial c_a}{\partial r})$$

are approximated by three point, centered FDs, including BC (6)

```
+2.0*D*(ca(nr-1,j)-ca(nr,j))/drs
```

To review the details, since at the wall (from BC (6)),

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

the second term in the diffusion group is zero and only the first term has to be considered

$$D\frac{\partial^2 c_a}{\partial r^2} \cong D\frac{c_a(nr+1) - 2c_a(nr) + c_a(nr-1)}{\Delta r^2}$$

But $c_a(nr+1)$ is a fictitious value that can be eliminated by a FD approximation of BC (6)

$$\frac{\partial c_a(r_0,z,t)}{\partial r} \cong \frac{c_a(nr+1) - c_a(nr-1)}{2\Delta r} = 0$$

or

$$c_a(nr+1) = c_a(nr-1)$$

and

$$D\frac{\partial^2 c_a}{\partial r^2} \cong D\frac{c_a(nr+1) - 2c_a(nr) + c_a(nr-1)}{\Delta r^2} = 2D\frac{c_a(nr-1) - c_a(nr)}{\Delta r^2}$$

which is programmed as indicated above.

(11.3) The reaction term

$$-k_r c_a^2$$

is programmed as

```
-rk*ca(nr,j)^2;
```

(again, using the subscript nr).
(11.4) The energy balance is easily programmed since the wall temperature is constant

```
 T(nr,j)=Tw;
Tt(nr,j)=0.0;
```

Note that the temporal derivative is zeroed since the wall temperature is constant.

(12) The temporal derivatives of eqs. (1) and (2) at the interrior points (with $r \neq 0$ and $r \neq r_0$) are then programmed

```
%          Interior, r ~= 0 and r ~= r0
           for i=2:nr-1
             rk=rk0*exp(-E/(R*T(i,j)));
             cat(i,j)=-v*(ca(i,j)-ca(i,j-1))/dz...
                    +D*(ca(i+1,j)-2.0*ca(i,j)+ca(i-1,j))/drs...
                    +D*(1.0/r(i)*(ca(i+1)-ca(i-1)))/(2.0*dr)...
                    -rk*ca(i,j)^2;
             Tt(i,j)=-v*(T(i,j)-T(i,j-1))/dz...
                    +alpha*(T(i+1,j)-2.0*T(i,j)+T(i-1,j))/drs...
                    +alpha*(1.0/r(i)*(T(i+1,j)-T(i-1,j)))/(2.0*dr)...
                    -dH*rk/(rho*Cp)*ca(i,j)^2;
%          Next r
           end
%
%        Next z
         end
```

This programming appears to be relatively complicated, but actually it is just the systematic application of two point upwind FD approximations for the convective derivative in z, and three point centered approximations for the diffusive derivatives in r (no special cases have to be considered as with $r = 0$ and $r = r_0$). Note that this coding concludes the loop in z (index j) initiated at the beginning of the programming of the temporal derivatives, so that two end statements are required.

(13) Finally, all nz x nr dependent variables are advanced in time through the application of Euler's method

```
%   All temporal derivatives are computed
%
```

```
%   Take Euler step
    for j=1:nz
    for i=1:nr
      ca(i,j)=ca(i,j)+cat(i,j)*h;
       T(i,j)= T(i,j)+ Tt(i,j)*h;
    end
    end
    t=t+h;
```

(14) The time stepping is concluded, followed by the looping for output (after nout Euler steps) and finally the loop for different cases is concluded

```
% Next Euler step
  end
%
% Next output
  end
%
% Next case
  end
```

In summary, although the preceding coding is relatively detailed, it follows the usual MOL two-step process:

- Evaluation of the initial value derivatives (RHSs of the PDEs), which includes the calculation of the spatial derivatives and application of the BCs, to produce a set of approximating ODEs
- Integration of the ODEs to produce a new set of dependent variable values that are then used to evaluate now initial value derivatives

The output from this program is:

```
ncase =     1    h =  1.000e-01

    t    ca(r,zl,t)  T(r,zl,t)
  0.0     0.000000    305.00
  0.0     0.000000    305.00
  0.0     0.000000    305.00
  0.0     0.000000    305.00
  0.0     0.000000    305.00

 10.0     0.000000    333.05
 10.0     0.000000    334.51
 10.0     0.000000    338.58
 10.0     0.000000    344.22
 10.0     0.000000    350.00
             .              .
             .              .
 50.0     0.000030    349.94
 50.0     0.000030    349.94
 50.0     0.000030    349.96
 50.0     0.000030    349.98
 50.0     0.000030    350.00
             .              .
             .              .
100.0     0.003540    352.84
100.0     0.003541    352.65
100.0     0.003544    352.08
100.0     0.003547    351.18
100.0     0.003549    350.00
             .              .
             .              .
150.0     0.004622    355.44
150.0     0.004624    355.06
150.0     0.004632    353.96
150.0     0.004643    352.23
150.0     0.004649    350.00
             .              .
             .              .
  200.0    0.004624    355.45
  200.0    0.004626    355.07
  200.0    0.004634    353.96
  200.0    0.004645    352.23
  200.0    0.004651    350.00
             .            .
             .            .
  250.0    0.004624    355.45
  250.0    0.004626    355.07
  250.0    0.004634    353.96
  250.0    0.004645    352.23
  250.0    0.004651    350.00
             .            .
             .            .
  300.0    0.004624    355.45
  300.0    0.004626    355.07
  300.0    0.004634    353.96
  300.0    0.004645    352.23
  300.0    0.004651    350.00
```

This output would be a good candidate for plotting, or better, visualization. For example, centerline profiles could be plotted, ca(0,z,t), T(0,z,t).

Also, some experimentation with the temporal and spatial integration should be done, for example:

- Use a higher (than Euler) order ODE integrator such as RKF45. Also, the Euler coding could be replaced by a library integrator such as Matlab ode23, ode45. This basically amounts to p refinement in time

- The ODE integration step h could also be varied; this amounts to h refinement in time

- Use a higher (than two point upwind) spatial differentiator. FDs would probably be ok for the same reason that they were satisfactory in the heat exchanger, Case = 1, 2; i.e., the solution is probably sufficiently smooth for FDs to give give accuracy, free of numerical diffusion and oscillation

  - This amounts to p refinement in space
  - The spatial differentiation can be programmed with library routines rather than explicitly
  - Centered approximations should be used for the diffusion spatial derivatives in r (there is no preferred direction)
  - Noncentered approximations should be used for the convection derivatives in z (some form of upwinding is required; centered approximations should not be used)

- The spatial integration intervals in r and z could be varied (the number of grid points in r and z); the amounts to h refinement in space

The files pertaining to this tubular reactor example are in:

http://www.lehigh.edu/˜ wes1/apci/reactor.ppt
http://www.lehigh.edu/˜ wes1/apci/reactor.m

Finally,

"The method of lines has become the most widely used solution technique for large-scale time-dependent partial differential equations", G. Fairweather and I. Gladwell, Preface to the special issue on the method of lines, Applied Numerical Mathematics, v. 20, 1996, pp 1-2

In the preceding examples, we have used FD approximations that were either preprogrammed, e.g., in dss002.m, vanl2.m, or were merely stated, then used, e.g., in the reactor application. We now consider a systematic procedure for generating FD approximations for virtually any situation, that is, FDs of any order for derivatives of any order, including BCs.

The files pertaining to this heat exchanger example are in:

http://www.lehigh.edu/˜ wes1/apci/htex1.ppt
http://www.lehigh.edu/˜ wes1/apci/htex1.m

http://www.lehigh.edu/˜ wes1/apci/dss002.m
http://www.lehigh.edu/˜ wes1/apci/dss012.m
http://www.lehigh.edu/˜ wes1/apci/dss020.m
http://www.lehigh.edu/˜ wes1/apci/vanl2.m

Finite Differences

If we applying the MOL to the one-dimensional Fourier's second law in cylindrical coordinates,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} \tag{1}$$

or in subscript notation

$$u_t = u_{rr} + (1/r)u_r$$

we need a FD approximation for the first and second derivatiuves, u_r and u_{rr}.

Considering the first derivative, we define a spatial grid in r with N grid points and index $i = 1, 2, \cdots, N$, the left grid point corresponds to $i = 1$. We can use two Taylor series centered at grid point $i = 1$

$$u_2 = u_1 + \frac{du_1}{dr}\Delta r + \frac{d^2u_1}{dr^2}\frac{\Delta r^2}{2!} + \frac{d^3u_1}{dr^3}\frac{\Delta r^3}{3!} + \cdots$$

$$u_0 = u_1 + \frac{du_1}{dr}(-\Delta r) + \frac{d^2u_1}{dr^2}\frac{(-\Delta r)^2}{2!} + \frac{d^3u_1}{dr^3}\frac{(-\Delta r)^3}{3!} + \cdots$$

Subtraction of the second series from the first gives

$$u_2 - u_0 = 2\frac{du_1}{dr}\frac{\Delta r}{1!} + 2\frac{d^3u_1}{dr^3}\frac{\Delta r^3}{3!}\cdots$$

or, since we are interested in an approximation for the first derivative,

$$\frac{du_1}{dr} \cong \frac{u_2 - u_0}{2\Delta r} - \frac{d^3u_1}{dr^3}\frac{\Delta r^2}{3!}$$

and we therefore have the well known centered, second order FD approximation for a first derivative

$$\frac{du_1}{dr} \cong \frac{u_2 - u_0}{2\Delta r} + O(\Delta r^2) \tag{2}$$

Note, however, that eq. (2) uses u_0 which is outside the grid, i.e., it is u at a fictitious point.

We, however, can use another approach that eliminates the use of fictitious points, starting with Taylor series expansions based on only the boundary and interior points. Thus,

$$u_2 = u_1 + \frac{du_1}{dr}\Delta r + \frac{d^2u_1}{dr^2}\frac{\Delta r^2}{2!} + \frac{d^3u_1}{dr^3}\frac{\Delta r^3}{3!} + \cdots$$

$$u_3 = u_1 + \frac{du_1}{dr}2\Delta r + \frac{d^2u_1}{dr^2}\frac{(2\Delta r)^2}{2!} + \frac{d^3u_1}{dr^3}\frac{(2\Delta r)^3}{3!} + \cdots$$

The derivative of interest is $\frac{du_1}{dr}$, and we wish to drop as many terms past this derivative as possible to achieve maximum accuracy.

If we multiply the first Taylor series by 4, and subtract the second Taylor series from it, we can drop out the second derivative terms (involving $\frac{d^2u_1}{dr^2}$)

$$4\left(u_2 = u_1 + \frac{du_1}{dr}\Delta r + \frac{d^2u_1}{dr^2}\frac{\Delta r^2}{2!} + \frac{d^3u_1}{dr^3}\frac{\Delta r^3}{3!} + \cdots\right)$$

$$u_3 = u_1 + \frac{du_1}{dr}2\Delta r + \frac{d^2u_1}{dr^2}\frac{(2\Delta r)^2}{2!} + \frac{d^3u_1}{dr^3}\frac{(2\Delta r)^3}{3!} + \cdots$$

or

$$4u_2 - u_3 = 3u_1 + 2\frac{du_1}{dr}\Delta r + 0\frac{d^2u_1}{dr^2}\frac{\Delta r^2}{2!} + (-4)\frac{d^3u_1}{dr^3}\frac{\Delta r^3}{3!} + \cdots$$

or, truncating after the third derivative terms, then solving for the derivative of interest, $\frac{du_1}{dr}$,

$$\frac{du_1}{dr} = \frac{-3u_1 + 4u_2 - u_3}{2\Delta r} + \frac{4}{2}\frac{d^3u_1}{dr^3}\frac{\Delta r^2}{3!}$$

Note that this second order approximation for the first derivative requires only boundary (u_1) and interior (u_2, u_3) values.

A similar analysis at the right boundary (grid point $i = n$) gives

$$\frac{du_N}{dr} = \frac{3u_N - 4u_{N-1} + u_{N-2}}{2\Delta r} + O(\Delta r^2) \tag{3}$$

At the interior points, we can use the centered approximation (this follows from eq. (2))

$$\frac{du_i}{dr} = \frac{u_{i+1} - u_{i-1}}{2\Delta r} + O(\Delta r^2) \tag{4}$$

Then for the entire domain, we can form a differentiation matrix (or computational stencil)

$$\frac{d\mathbf{u}}{dr} = \frac{1}{2\Delta r}\begin{bmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{bmatrix}\mathbf{u} + O(\Delta r^2)$$

where row 1 is used for grid point $i = 1$, row 2 for $i = 2$ to $i = N - 1$, and row 3 for $i = N$. Note that the matrix is antisymmetric with respect to the center element (antisymmetric around 0).

To make explicit the points at which each row of the differentiation is used, we can write it as

$$\frac{d\mathbf{u}}{dr} = \frac{1}{2\Delta r}\left[\begin{array}{ccccc} -3 & 4 & -1 & & i=1 \\ -1 & 0 & 1 & & i=2,\cdots,N-1 \\ 1 & -4 & 3 & & i=N \end{array}\right]\mathbf{u} + O(\Delta r^2) \tag{5}$$

The differentiation matrix of eq. (5) is programmed in function dss002.m.

Again, note that FD approximations are linear, i.e., the derivative at each grid point is calculated as a weighted sum of the neighboring dependent variable values, as illustrated by eqs. (3) and (4).

To obtain a FD approximation for the second derivative in eq. (1), we can use the same two Taylor series as before (but now written at a grid point i to give a general result)

$$u_{i+1} = u_i + \frac{du_i}{dr}\Delta r + \frac{d^2u_i}{dr^2}\frac{\Delta r^2}{2!} + \frac{d^3u_i}{dr^3}\frac{\Delta r^3}{3!} + \cdots$$

$$u_{i-1} = u_i + \frac{du_i}{dr}(-\Delta r) + \frac{d^2u_i}{dr^2}\frac{(-\Delta r)^2}{2!} + \frac{d^3u_i}{dr^3}\frac{(-\Delta r)^3}{3!} + \cdots$$

Adding these two series (rather than subtracting them as before) gives

$$u_{i+1} + u_{i-1} = 2u_i + 2\frac{d^2u_i}{dr^2}\frac{\Delta r^2}{2!} + 2\frac{d^4u_i}{dr^4}\frac{\Delta r^4}{4!}$$

Solving for the second derivative, we have

$$\frac{d^2u_i}{dr^2} \cong \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta r^2} - 2\frac{d^4u_i}{dr^4}\frac{\Delta r^2}{4!}$$

or

$$\frac{d^2u_i}{dr^2} \cong \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta r^2} + O(h^2) \tag{6}$$

which is the well known centered, second order, FD approximation for the second derivative.

Note again that eq. (6) references u_0 when $i = 1$, so we would have to somehow provide this fictitious value, generally through a BC at $i = 1$. An alternate approach is to use a noncentered difference for the second derivative at $i = 1$ that requires only the boundary and interior points. The derivation is a bit more involved than for first derivatives, but it has been done and programmed in library spatial differentiators.

For example, the code is given below for a spatial differentiation routine (dss042.m):

```
      function [uxx]=dss042(xl,xu,n,u,ux,nl,nu)
%...
%...  FUNCTION DSS042 COMPUTES A SECOND-ORDER APPROXIMATION OF A
%...  SECOND-ORDER DERIVATIVE, WITH OR WITHOUT THE NORMAL DERIVATIVE
%...  AT THE BOUNDARY.
%...
%...  ARGUMENT LIST
%...
%...     XL      LEFT VALUE OF THE SPATIAL INDEPENDENT VARIABLE (INPUT)
%...
%...     XU      RIGHT VALUE OF THE SPATIAL INDEPENDENT VARIABLE (INPUT)
%...
%...     N       NUMBER OF SPATIAL GRID POINTS, INCLUDING THE END
%...             POINTS (INPUT)
%...
%...     U       ONE-DIMENSIONAL ARRAY OF THE DEPENDENT VARIABLE TO BE
%...             DIFFERENTIATED (INPUT)
%...
%...     UX      ONE-DIMENSIONAL ARRAY OF THE FIRST DERIVATIVE OF U.
%...             THE END VALUES OF UX, UX(1) AND UX(N), ARE USED IN
%...             NEUMANN BOUNDARY CONDITIONS AT X = XL AND X = XU,
%...             DEPENDING ON THE ARGUMENTS NL AND NU (SEE THE DE-
%...             SCRIPTION OF NL AND NU BELOW)
```

```
%...
%...     UXX     ONE-DIMENSIONAL ARRAY OF THE SECOND DERIVATIVE OF U
%...             (OUTPUT)
%...
%...     NL      INTEGER INDEX FOR THE TYPE OF BOUNDARY CONDITION AT
%...             X = XL (INPUT).  THE ALLOWABLE VALUES ARE
%...
%...                1 - DIRICHLET BOUNDARY CONDITION AT X = XL
%...                    (UX(1) IS NOT USED)
%...
%...                2 - NEUMANN BOUNDARY CONDITION AT X = XL
%...                    (UX(1) IS USED)
%...
%...     NU      INTEGER INDEX FOR THE TYPE OF BOUNDARY CONDITION AT
%...             X = XU (INPUT).  THE ALLOWABLE VALUES ARE
%...
%...                1 - DIRICHLET BOUNDARY CONDITION AT X = XU
%...                    (UX(N) IS NOT USED)
%...
%...                2 - NEUMANN BOUNDARY CONDITION AT X = XU
%...                    (UX(N) IS USED)
%...
%...  GRID SPACING
      dx=(xu-xl)/(n-1);
%...
%...  CALCULATE UXX AT THE LEFT BOUNDARY, WITHOUT UX
      if nl==1
      uxx(1)=((     2.)*u(  1)...
             +(    -5.)*u(  2)...
             +(     4.)*u(  3)...
             +(    -1.)*u(  4))/(dx^2);
%...
%...  CALCULATE UXX AT THE LEFT BOUNDARY, INCLUDING UX
      elseif nl==2
      uxx(1)=((    -7.)*u(  1)...
             +(     8.)*u(  2)...
             +(    -1.)*u(  3))/(2.*dx^2)...
             +(    -6.)*ux( 1) /(2.*dx);
      end
%...
%...  CALCULATE UXX AT THE RIGHT BOUNDARY, WITHOUT UX
      if nu==1
      uxx(n)=((     2.)*u(n  )...
             +(    -5.)*u(n-1)...
             +(     4.)*u(n-2)...
             +(    -1.)*u(n-3))/(dx^2);
%...
%...  CALCULATE UXX AT THE RIGHT BOUNDARY, INCLUDING UX
      elseif nu==2
      uxx(n)=((    -7.)*u(n  )...
             +(     8.)*u(n-1)...
             +(    -1.)*u(n-2))/(2.*dx^2)...
             +(     6.)*ux(n ) /(2.*dx);
      end
%...
%...  CALCULATE UXX AT THE INTERIOR GRID POINTS
      for i=2:n-1
         uxx(i)=(u(i+1)-2.*u(i)+u(i-1))/dx^2;
      end
```

Similar routines are available for fourth, sixth, eighth and tenth order approximations of a second derivative, including Dirichlet and Neumann boundary conditions at the boundaries.

We now all of the spatial derivative approximations required to convert eq. (1) into a system of ODEs (using the approach of the MOL)

$$\frac{du_i}{dt} = (\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta r^2}) + \frac{1}{r_i}(\frac{u_{i+1} - u_{i-1}}{2\Delta r}) \tag{7}$$

where $i = 2, 3, \cdots, N-1$. For $i = 1$ and $i = N$, we use noncentered approximations so that fictitious points are avoided. Also, we have to provide two BCs in the programming of the ODEs, one at $i = 1$ and one at $i = N$. The preprogramming of the FD approximations for first and second derivatives in library routines (such as dss002.m and dss042.m) makes the implementation of BCs straightforward.

We can use more grid points in the approximations, buy starting with more Taylor series for neighboring grid points, to arrive at higher order FD approximations. For example, if we use five points (rather than three) in the FD approximations, the differentiation matrix is (note: the spatial variable is now z rather than r)

$$\frac{d\mathbf{u}}{dz} = \begin{bmatrix} -50 & 96 & -72 & 32 & -6 \\ -6 & -20 & 36 & -12 & 2 \\ 2 & -16 & 0 & 16 & -2 \\ -2 & 12 & -36 & 20 & 6 \\ 6 & -32 & 72 & -96 & 50 \end{bmatrix} \mathbf{u} + O(\Delta z^4)$$

We can now assign various combinations of grid points to the rows. For example, if we use

Grid point(s)	Weighting coefficients
1	$-50, 96, -72, 32, -6$
2	$-6, -20, 36, -12, 2$
$3, \cdots N-2$	$2, -16, 0, 16, -2$
$N-1$	$-2, 12, -36, 20, 6$
N	$6, -32, 72, -96, 50$

then we have used primarily centered approximations, which would be appropriate for diffusion problems, such as for the first derivative in eq. (1).

If we use

Grid point(s)	Weighting coefficients
1	$-50, 96, -72, 32, -6$
2	$-6, -20, 36, -12, 2$
3	$2, -16, 0, 16, -2$
$4, \cdots, N-1$	$-2, 12, -36, 20, 6$
N	$6, -32, 72, -96, 50$

these are the biased upwind approximations in dss020.m discussed in the heat exchanger example. In other words, the differentiation matrix is the basis for a spectrum of FD approximations (and their coding follows directly from the differentiation matrix).

Finally, all of the preceding discussion has been for equally spaced grids. If the grid is unequally spaced, for example, to concentrate the grid points in regions of large spatial variation of the solution, analogous approximations can be derived which have the same mathematical form, i.e., weighted sums of the dependent variable at neighboring points. The derivation of these nonuniform approximations is a bit more involved (the Taylor series is not as easy to use for this case, and the starting point is therefore usually the Lagrange interpolation polynomial).

Fortunately, all of this has been done by Bengt Fornberg in a very compact algorithm that is programmed in a variety of languages, e.g., Fortran, Mathematica, Maple: Fornberg, B., *Calculation of Weights in Finite Difference Formulas*, SIAM Review, v. 40, no. 3, pp 685-691, Sept., 1998. The small Fortran subroutine, weights.for, that accompanies this algorithm is widely used in PDE numerical analysis. It can produce the FD weighting coefficients for

- Derivatives of any order m ($\frac{d^m}{dz^m}$)
- Approximations of any order p ($O(\Delta z^p)$)
- Approximations at any grid point, $i = 1, ..., n$ (centered and noncentered to avoid fictitious points)
- Uniform and nonuniform grids (Δz does not have to be constant), so that the grid points can be concentrated where the solution changes rapidly in z; further if the code does this automatically, we have adaptive mesh refinement (*r refinement* or *amr*).

To conclude, we select approximations based on the general characteristics of the PDEs (hyperbolic, parabolic, elliptic). Here are some general guidelines

- Centered approximations are appropriate for elliptic and parabolic PDEs since there is no preferred direction

- Parabolic PDEs will generally “diffuse” away discontinuities so that, for example, incompatibilites between initial and boundary conditions will not cause numerical difficulties

- Elliptic problems, which have no initial value derivatives, can be accommodated by converting them to parabolic PDEs (by adding pseudo time derivatives, then integrating until these derivatives essentially go to zero)

- Hyperbolic PDEs are the most difficult to solve numerically because they propagate discontinuities

- Centered approximations for hyperbolic problem generally don’t work (they usually produce unrealistic oscillations)

- Noncentered approximations must be used for hyperbolic PDEs that take into account the preferred direction (the direction of flow); in general, hyperbolic PDEs require some form of upwinding

- Thus, we have to know the direction of flow so we can then select the direction for the upwinding

- Using higher order approximations does not guarantee better accuracy; for example, higher order centered approximations applied to hyperbolic problems actually produces more oscillation

- The choice of an approximation(s), particularly for a high order, nonlinear PDE problem, is as much art as science; thus we should be aware of and consider the use of a variety of approximations, e.g., FDs, FVs, FEs, least squares, flux limiters, ENO (essentially nonoscillatory), TVD (total variation diminishing), etc. approximations

- These approximations originated in the CFD field, primarily to minimize numerical distortion and error such as numerical diffusion and oscillation; typically, CFD applicatiuons require the solution of strongly hyperbolic PDEs systems, e.g., the Euler and Navier Stokes equations

- Numerical methods for PDEs is a very active field of research, with new methods and associated software continuously being announced
- Much of the impetus for new, high speed (supercomputers) comes from applications requiring the solution of systems of nonlinear, multidimensional, time dependent PDEs, e.g., CFD, aircraft design, geophysics, atmospheric modeling, numerical general relativity.

We now consider briefly some other forms of spatial discretization.

Weighted Residuals

The method of weighted residuals requires more analytical work than the FD approach to PDEs, and the analytical work is rather specific for the problem at hand (and thus, generally has to be repeated for each new problem). To give an idea of this approach, if we consider Fourier's second law in Cartesian coordinates

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{1}$$

$$u(x_0, t) = 0, \frac{\partial u(x_L, t)}{\partial x} = 0 \qquad (2)(3)$$

$$u(x, 0) = 1 \tag{4}$$

we assume a series solution of the form

$$u(x, t) \cong \sum_{i=1}^{N} c_i(t)\phi_i(x) \tag{5}$$

Note that the RHS of eq. (5) is a *product* or *separated,* which makes substituion into the problem equations relatively easy.

For example, if we compute the derivatives in eq. (1),

$$\frac{\partial u}{\partial t} \cong \sum_{i=1}^{N} c_i'(t)\phi_i(x)$$

$$\frac{\partial^2 u}{\partial x^2} \cong \sum_{i=1}^{N} c_i(t)\phi_i''(x)$$

and then substitute these derivatives in eq. (1), we have

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x) = \sum_{i=1}^{N} c_i(t)\phi_i''(x) \tag{6}$$

Since eq. (5) is only an approximate solution, it generally will have an error or a *residual*, $R(x,t)$. Eq. (6) can be written in residual form as

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x) - \sum_{i=1}^{N} c_i(t)\phi_i''(x) = R(x,t) \tag{7}$$

If we could make the residual zero everywhere, we would have the exact solution. Generally, this is not possible, so we try to approximate this condition by applying a weighting function, $w(x)$, to the residual according to the integral

$$\int_{x_l}^{x_u} w(x)R(x,t)dx = 0 \tag{8}$$

This is the basis of the *method of weighted residuals.* Several well known methods then follow from eq. (7), depending on the form of the weighting function.

- Setting the residual to zero at a series of values of x is the *collocation method.* The expectation, then, is that the residual will be small at points in between those points where it is set to zero. Fort example, if we select points $x_l < x_1 < x_2 < x_3 < x_u$ as collocation points, then from eq. (7), with $N = 3$

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x_1) - \sum_{i=1}^{N} c_i(t)\phi_i''(x_1) = 0$$

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x_2) - \sum_{i=1}^{N} c_i(t)\phi_i''(x_2) = 0$$

$$\sum_{i=1}^{N} c_i'(t)\phi_i(x_3) - \sum_{i=1}^{N} c_i(t)\phi_i''(x_3) = 0$$

The first equation is

$$c_1'(t)\phi_1(x_1) + c_2'(t)\phi_2(x_1) + c_3'(t)\phi_3(x_1) - c_1(t)\phi_1''(x_1) - c_2(t)\phi_2''(x_1) - c_3(t)\phi_3''(x_1) = 0$$

This is a linearly implicit ODE with coupling coefficients $\phi_1(x_1), \phi_2(x_1), \phi_3(x_1)$. This, we can use the Cash code discussed previously for the 3×3 problem.

The BCs are accommodated according to the choice of the *basis functions*, $\phi_i(x)$. Typically, these basis functions are chosen to identically satisfy the boundary conditions. If the collocation points are chosen as the roots of an orthogonal polynomial, we then have *orthogonal collocation* (OC). This choice of the collocation points gives some error minimizing qualities to the solution.

The weighting function for collocation in eq. (8) can be considered a delta function, i.e.,

$$\int_{x_l}^{x_u} \delta(x - x_1)R(x,t)dx = R(x_1,t) = 0$$

so the residual is set to zero at $x = x_1$ as above.

- If the weighting function is taken as a basis function,

$$\int_{xl}^{x_u} \phi_i(x) R(x,t) dx = 0 \tag{10}$$

  This is the *Galerkin method.* The essential idea is that the *basis functions are orthogonal to the residual.* Again, this has certain error minimizing qualities for the solution of the PDE through eq. (5) (it does not make the residual zero for all x but rather, makes it small, on average over the interval $x_l \leq x \leq x_u$). If the Galerkin criterion (10) is applied to the N weighting functions, we again arrive at linearly implicit ODEs (or DAEs depending on how the BCs are handled).

Finite Elements

Finite elements (FE) is a vast subject, and we can only mention here a few basic ideas. If in the method of weighted residuals as summarized by eq. (8), we choice the basis functions, $\phi_i(x)$, to have *compact support*, that is, to have nonzero values only over a finite interval in x, these "elements" then serve as the basis of the assumed solution such as eq. (5). The elements can have any mathematical form (and many have been proposed and used); a common choice is the so-called "hat function" or "chapeau function", which makes the evaluation of the integrals in the FE method relatively easy. Thus, if we substitute the residual for the preceding PDE problem in eq. (10)

$$\int_{x_l}^{x_u} \phi_j(x) R(x,t) dx = \int_{x_l}^{x_u} \phi_j(x) \left\{ \sum_{i=1}^{N} c_i'(t)\phi_i(x) - \sum_{i=1}^{N} c_i(t)\phi_i''(x) \right\} dx = 0$$

Note now that when the integration in x is performed (e.g., by interchanging the orders of integration and summation, and using integration by parts), the result is a function of t only, and we again arrive at linearly implicit ODEs. For example, using hat functions as the FEs, we obtain the implicit ODE

$$(1/6)c'_{i+1} + (4/6)c'_i + (1/6)c'_{i-1} = D\left\{\frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta x^2}\right\} \tag{11}$$

Eq. (11) is of the same form as the FD approximation of eq. (1) using second order, central FDs. Note, however, that in using the FE method, we

- Had to do a substantial amount of analytical work, even for the modest problem of eqs. (1) to (4).
- Used FEs which are only linear (rather than the higher order polynomials that are the basis of the higher order FDs)

Thus, we might expect we would have to use a relatively large number of FEs (and therefore grid points and ODEs) to achieve accuracy in the PDE solution comparable to a higher order FD. These conclusions also apply to orthgonal collocation (OC is relatively difficult to apply, and its performance is probably not any better than when using carefully selected FDs).

The principal utility of the FE method is the spatial discretization on irregular geometries, for which the FD method becomes diffiicult to apply. Thus, body fitted coordinates are a very active area of research, and the grid that results from this analysis (based on triangles, for example), is then used within the FE framework. Computer codes that accommodate irregular geometries are therefore generally based on the FE method, and a key consideration in designing and using these codes is the choice of the basic finite element shapes. A spectrum of finite elements has been developed and analyzed, particularly for 2- and 3-D PDE systems. This is the basis of the Matlab PDE toolbox, for example (it is a 2-D FE code).

Finite Volumes

Another approach to spatial discretization (in addition to FDs and FEs) is the finite volume (FV) formulation. Briefly, this method is based on the use of conservation principles such as the conservation of mass and conservation of energy, as applied to a finite or incremental volume of the system to be analyzed. To illustrate this approach, we again consider the analysis of a system modeled by the one-dimensional heat conduction equation, and we then subdivide the spatial coordinate into a grid, corresponding to a series of contiguous volumes. An energy balance wtitten on one of the finite volumes is then

$$V_i \rho_i C_{pi} \frac{du_i}{dt} = A_{i-1} q_{i-1} - A_i q_i \tag{12}$$

which is interpreted to conserve eneregy exactly by the proper choice of the fluxes, q_i and q_{i-1}.

For example, the fluxes could be computed as

$$q_i = k_{i,i+1} \left(\frac{u_i - u_{i+1}}{\Delta x_i/2 + \Delta x_{i+1}/2} \right)$$

where u_i is interpreted as the mid-cell (or mid-volume) temperature. Thus, we end up again with a system of ODEs, so that the FV method can be considered as a variant of the MOL, with a different spatial discretization (than FDs or FEs). The challenge, in general, in using the FV approach is to calculate the intercell fluxes in such a way that conservation is preserved. This becomes more difficult for multidimensions, irregular geometries and coordinate systems other than Cartesian coordinates. Thus, a substantial effort has recently been devoted to the FV method in cylindrical and spherical coordinates so that conservation is maintained, e.g., in the solution of Maxwell's equations for calculating EM fields in communications systems with arrays (antennas) with non-Cartesian shapes.

In summary, the use of the FV method is a very active field, with the attractive feature of maintaining conservation of basic physical quantities, so that the discretizied (approximated) PDEs in fact maintain the conservation implied by the PDEs themselves.

Adaptive Grids

The equidistribution principle uses a monitor function based on the curvature of the solution.

At the time level t_{k+1}, the nodes $x_i^{k+1}, i = 1, \ldots, N$ are located such that

$$\int_{x_{i-1}^{k+1}}^{x_i^{k+1}} m(u(x, t_{k+1}))dx = \int_{x_{i-1}^{k+1}}^{x_i^{k+1}} \left\{ \alpha + \left\| \frac{\partial^2 u(x, t_{k+1})}{\partial x^2} \right\|_\infty \right\}^{1/2} dx = \text{constant}$$

where $u(x, t_{k+1})$ is the PDE solution which has been advanced to the time t_{k+1} using the fixed grid $x_i^k, i = 1, \ldots, N$

MOL Formulation:

- Finite difference approximations (Fornberg)
- Time integration with the implicit BDF solver LSODI (Hindmarsh) or the implicit RK solver RADAU5 (Hairer, Wanner)
- Parameters
  - Scaling factor α
  - Limiting factor β
  - Updating frequency N_{adapt}
- Adaptive grids require parameter tuning
- There is no universal method ... yet!
- Our Fortran codes are generally available, e.g, subroutine AGE
- The contributions of Dr. Alain Vande Wouwer and Dr. Philippe Saucez, Polytechnique Mons (Belgium) are gratefully acknowledged

References and Software

Sources of mathematical software include:

Name: Transactions on Mathematical Software (TOMS), Association for Computing Machinery (ACM)

- Features: An extensive library of high quality, public domain mathematical software
- Reference: http://www.acm.org/dl/search.html

Name: Netlib (Inter**net** **Lib**rary) - accessed 74,280,550 times by 22APR00

- Features: An extensive library of high quality, public domain PDE mathematical software
- Reference:
  http://www.netlib.org/liblist.html
  Select pde

  netlib@ornl.gov (whois ...., send index, send index from toms)

Name: NA-Digest (**N**umerical **A**nalysis Digest; an electronic newsletter)

- Features: A free, weekly newsletter edited by Dr. Cleve Moler, the author of Matlab; to subscribe and get information:
  na.digest@na-net.ornl.gov
  na.whois@na-net.ornl.gov
  na.help@na-net.ornl.gov
- Reference: All past issues of NA-Digest are in Netlib (and they can be searched by subject and author)
- Example:

  From: DEAL <deal@hermes.iwr.uni-heidelberg.de>
  Date: Tue, 18 Apr 2000 10:42:32 +0200 (MET DST)
  Subject: DEAL, C++ Finite Element Library

  Version 3.0 of the deal.II object-oriented finite element library is available on the deal.II home-page at

  http://gaia.iwr.uni-heidelberg.de

deal.II is a C++ program library targeted at adaptive finite elements and error estimation. It uses state-of-the-art programming techniques of the C++ programming language to offer you a modern interface to the complex data structures and algorithms required for adaptivity and enables you to use a variety of finite elements in one, two, and three space dimensions, as well as support for time-dependent problems.

The library is written for research purposes and offers many features:

- Support for one, two, and three space dimensions, using a unified interface that enables writing programs almost dimension independent.

- Handling of locally refined grids, including different adaptive refinement strategies based on local error indicators and error estimators.

- Support for a variety of finite elements, including Lagrange elements of order one through four, and discontinuous elements.

- Extensive documentation: all documentation is available online in a logical tree structure to allow fast access to the information you need. If printed it comprises about 200 pages of tutorials, several reports, and far more than 1,000 pages of programming interface documentation with explanations of all classes, functions, and variables.

- Modern software techniques that make access to the complex data structures and algorithms as transparent as possible. The use of object oriented programming allows for program structures similar to the structures in mathematical analysis.

- Fast algorithms that enable you to solve problems with up to several millions of degrees of freedom quickly. As opposed to programming symbolic algebra packages the penalty for readability is low.

- Support for several output formats, including some common formats for visualization of scientific data.

- Support for a variety of computer platforms, including multi- processor machines.

- Free source code under an Open Source license, and the invitation to contribute to further development of the library.

Wolfgang Bangerth, Guido Kanschat, the deal.II team

From: Tom Goodale <goodale@aei-potsdam.mpg.de>
Date: Thu, 20 Apr 2000 16:13:55 +0200 (CEST)
Subject: Cactus, Parallel Code for PDEs

This is to announce the release of beta 7 of Cactus 4.0.

Cactus is a general, modular, parallel code for solving systems of partial differential equations. The code has been developmented over many years by a large international collaboration of numerical relativity and computational science research groups and can be used to provide a portable platform for solving any system of partial differential equations. The code compiles and runs on a variety of different platforms, from laptops running Linux or NT to clusters of workstations, to large T3Es, with no modifications needing to be made to the application codes.

As an example, one set of modules (thorns) can be used to solve problems in numerical relativity, for example black hole or neutron star collisions. The design of the code and the thorns means that general routines can be written, for example to locate the position of the event horizon, which can then be made generally available and used for other problems.

The code is available from

http://www.cactuscode.org

Notes for this release can be found at

http://www.cactuscode.org/Development/beta7.txt

For further information, please email

cactusmaint@cactuscode.org

The NAG library has a spectrum of PDE software, much of it developed by Martin Berzins, University of Leeds

Chapter D03 - Partial Differential Equations
D03EAF - Elliptic PDE, Laplace's equation, 2-D arbitrary domain
D03EBF - Elliptic PDE, solution of finite difference equations by SIP, five-point 2-D molecule, iterate to convergence
D03ECF - Elliptic PDE, solution of finite difference equations by SIP for seven- point 3-D molecule, iterate to convergence
D03EDF - Elliptic PDE, solution of finite difference equations by a multigrid technique
D03EEF - Discretize a 2nd order elliptic PDE on a rectangle
D03FAF - Elliptic PDE, Helmholtz equation, 3-D Cartesian co-ordinates
D03MAF - Triangulation of a plane region
D03PCF - General system of parabolic PDEs, method of lines, finite differences, one space variable
D03PDF - General system of parabolic PDEs, method of lines, Chebyshev C0 collocation, one space variable
D03PEF - General system of first order PDEs, method of lines, Keller box discretisation, one space variable
D03PFF - General system of PDEs, convection-diffusion in conservative form, method of lines, one space variable
D03PHF - General system of parabolic PDEs, coupled DAEs, method of lines, finite differences, one space variable
D03PLF - PDEs, general system, one space variable, method of lines, compute interpolant in spatial direction
D03PJF - General system of parabolic PDEs, coupled DAEs, method of lines, Chebyshev C0 collocation, one space variable

D03PKF - General system of first order PDEs, coupled DAEs, method of lines, Keller box discretisation, one space variable
D03PPF - General system of parabolic PDEs, coupled DAEs, method of lines, finite differences, remeshing, one space variable
D03PRF - General system of first order PDEs, coupled DAEs, method of lines, Keller box discretisation, remeshing, one space variable
D03PSF - General system of PDEs, convection-diffusion in conservative form, method of lines, coupled DAEs, remeshing, comprehensive one space variable
D03PUF - General system of PDEs, appropriate Riemann solver for Euler equations, method of lines, Roe's scheme, one space variable
D03PVF - General system of PDEs, appropriate Riemann solver for Euler equations, method of lines, Osher's scheme, one space variable
D03PWF - Modified HLL Riemann solver for Euler equations in conservative form, for use with D03PFF, D03PLF and D03PSF
D03PXF - Exact Riemann Solver for Euler equations in conservative form, for use with D03PFF, D03PLF and D03PSF
D03PYF - PDEs, spatial interpolation with D03PDF or D03PJF
D03PZF - PDEs, spatial interpolation with D03PCF or D03PHF
D03RAF - General system of second order PDEs, method of lines, finite differences, remeshing, two space variables, rectangular region
D03RBF - General system of second order PDEs, method of lines, finite differences, remeshing, two space variables, rectilinear region
D03RYF - Check initial grid data in D03RBF
D03RZF - Extract grid data from D03RBF
D03UAF - Elliptic PDE, solution of finite difference equations by SIP, five-point 2-D molecule, one iteration
D03UBF - Elliptic PDE, solution of finite difference equations by SIP, seven-point 3-D molecule, one iteration Chapter D04 - Numerical Differentiation
D04AAF - Numerical differentiation, derivatives up to order 14, function of one real variable