Solving Partial Differential Equations

In a *partial differential equation* (PDE), the function being solved for depends on several variables, and the differential equation can include partial derivatives taken with respect to each of the variables. Partial differential equations are useful for modelling waves, heat flow, fluid dispersion, and other phenomena with spatial behavior that changes over time.

What Types of PDEs Can You Solve with MATLAB?

The MATLAB® PDE solver `pdepe` solves initial-boundary value problems for systems of PDEs in one spatial variable x and time t. You can think of these as ODEs of one variable that also change with respect to time.

`pdepe` uses an informal classification for the 1-D equations it solves:

- Equations with a time derivative are *parabolic*. An example is the heat equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$.
- Equations without a time derivative are *elliptic*. An example is the Laplace equation $\frac{\partial^2 u}{\partial x^2} = 0$.

`pdepe` requires at least one parabolic equation in the system. In other words, at least one equation in the system must include a time derivative.

`pdepe` also solves certain 2-D and 3-D problems that reduce to 1-D problems due to angular symmetry (see the argument description for the symmetry constant `m` for more information).

Partial Differential Equation Toolbox™ extends this functionality to generalized problems in 2-D and 3-D with Dirichlet and Neumann boundary conditions.

Solving 1-D PDEs

A 1-D PDE includes a function $u(x,t)$ that depends on time t and one spatial variable x. The MATLAB PDE solver `pdepe` solves systems of 1-D parabolic and elliptic PDEs of the form

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right) + s\left(x,t,u,\frac{\partial u}{\partial x}\right).$$

The equation has the properties:

- The PDEs hold for $t_0 \le t \le t_f$ and $a \le x \le b$.
- The spatial interval $[a, b]$ must be finite.
- `m` can be 0, 1, or 2, corresponding to *slab*, *cylindrical*, or *spherical* symmetry, respectively. If $m > 0$, then $a \ge 0$ must also hold.
- The coefficient $f\left(x,t,u,\frac{\partial u}{\partial x}\right)$ is a flux term and $s\left(x,t,u,\frac{\partial u}{\partial x}\right)$ is a source term.
- The flux term must depend on the partial derivative $\partial u/\partial x$.

The coupling of the partial derivatives with respect to time is restricted to multiplication by a diagonal matrix $c\left(x,t,u,\frac{\partial u}{\partial x}\right)$.

The diagonal elements of this matrix are either zero or positive. An element that is zero corresponds to an elliptic equation, and any other element corresponds to a parabolic equation. There must be at least one parabolic equation. An element of c that corresponds to a parabolic equation can vanish at isolated values of x if they are mesh points (points where the solution is evaluated). Discontinuities in c and s due to material interfaces are permitted provided that a mesh point is placed at each interface.

Solution Process

To solve PDEs with `pdepe`, you must define the equation coefficients for c, f, and s, the initial conditions, the behavior of the solution at the boundaries, and a mesh of points to evaluate the solution on. The function call `sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)` uses this information to calculate a solution on the specified mesh:

- `m` is the symmetry constant.
- `pdefun` defines the equations being solved.
- `icfun` defines the initial conditions.
- `bcfun` defines the boundary conditions.
- `xmesh` is a vector of spatial values for x.
- `tspan` is a vector of time values for t.

Together, the `xmesh` and `tspan` vectors form a 2-D grid that `pdepe` evaluates the solution on.

Equations

You must express the PDEs in the standard form expected by `pdepe`. Written in this form, you can read off the values of the coefficients c, f, and s.

In MATLAB you can code the equations with a function of the form

```
function [c,f,s] = pdefun(x,t,u,dudx)
c = 1;
f = dudx;
s = 0;
end
```

In this case `pdefun` defines the equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$. If there are multiple equations, then c, f, and s are vectors with each element corresponding to one equation.

Initial Conditions

At the initial time $t = t_0$, for all x, the solution components satisfy initial conditions of the form

$$u(x, t_0) = u_0(x).$$

In MATLAB you can code the initial conditions with a function of the form

```
function u0 = icfun(x)
u0 = 1;
end
```

In this case `u0 = 1` defines an initial condition of $u_0(x,t_0) = 1$. If there are multiple equations, then `u0` is a vector with each element defining the initial condition of one equation.

Boundary Conditions

At the boundary $x = a$ or $x = b$, for all t, the solution components satisfy boundary conditions of the form

$$p(x, t, u) + q(x, t) f\left(x, t, u, \frac{\partial u}{\partial x}\right) = 0.$$

$q(x,t)$ is a diagonal matrix with elements that are either zero or never zero. Note that the boundary conditions are expressed in terms of the flux f, rather than the partial derivative of u with respect to x. Also, of the two coefficients $p(x,t,u)$ and $q(x,t)$, only p can depend on u.

In MATLAB you can code the boundary conditions with a function of the form

```
function [pL,qL,pR,qR] = bcfun(xL,uL,xR,uR,t)
pL = uL;
qL = 0;
pR = uR - 1;
```

```
qR = 0;
end
```

`pL` and `qL` are the coefficients for the left boundary, while `pR` and `qR` are the coefficients for the right boundary. In this case `bcfun` defines the boundary conditions

$$u_L(x_L, t) = 0$$
$$u_R(x_R, t) = 1$$

If there are multiple equations, then the outputs `pL`, `qL`, `pR`, and `qR` are vectors with each element defining the boundary condition of one equation.

Integration Options

The default integration properties in the MATLAB PDE solver are selected to handle common problems. In some cases, you can improve solver performance by overriding these default values. To do this, use `odeset` to create an `options` structure. Then, pass the structure to `pdepe` as the last input argument:

```
sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,options)
```

Of the options for the underlying ODE solver `ode15s`, only those shown in the following table are available for `pdepe`.

Category	Option Name
Error control	`RelTol`, `AbsTol`, `NormControl`
Step-size	`InitialStep`, `MaxStep`
Event logging	`Events`

Evaluating the Solution

After you solve an equation with `pdepe`, MATLAB returns the solution as a 3-D array `sol`, where `sol(i,j,k)` contains the kth component of the solution evaluated at `t(i)` and `x(j)`. In general, you can extract the kth solution component with the command `u = sol(:,:,k)`.

The time mesh you specify is used purely for output purposes, and does not affect the internal time steps taken by the solver. However, the spatial mesh you specify can affect the quality and speed of the solution. After solving an equation, you can use `pdeval` to evaluate the solution structure returned by `pdepe` with a different spatial mesh.

Example: The Heat Equation

Open in MATLAB Online

View MATLAB Command

An example of a parabolic PDE is the heat equation in one dimension:

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

This equation describes the dissipation of heat for $0 \le x \le L$ and $t \ge 0$. The goal is to solve for the temperature $u(x,t)$. The temperature is initially a nonzero constant, so the initial condition is

$$u(x,0) = T_0.$$

Also, the temperature is zero at the left boundary, and nonzero at the right boundary, so the boundary conditions are

$$u(0,t) = 0,$$
$$u(L,t) = 1.$$

To solve this equation in MATLAB, you need to code the equation, initial conditions, and boundary conditions, then select a suitable solution mesh before calling the solver `pdepe`. You either can include the required functions as local functions at the end of a file (as in this example), or save them as separate, named files in a directory on the MATLAB path.

Code Equation

Before you can code the equation, you need to make sure that it is in the form that the `pdepe` solver expects:

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right) + s\left(x,t,u,\frac{\partial u}{\partial x}\right).$$

In this form, the heat equation is

$$1\cdot\frac{\partial u}{\partial t} = x^0\frac{\partial}{\partial x}\left(x^0\frac{\partial u}{\partial x}\right) + 0.$$

So the values of the coefficients are as follows:

- $m = 0$
- $c = 1$
- $f = \dfrac{\partial u}{\partial x}$
- $s = 0$

The value of m is passed as an argument to `pdepe`, while the other coefficients are encoded in a function for the equation, which is

```
function [c,f,s] = heatpde(x,t,u,dudx)
c = 1;
f = dudx;
s = 0;
end
```

(Note: All functions are included as local functions at the end of the example.)

Code Initial Condition

The initial condition function for the heat equation assigns a constant value for u_0. This function must accept an input for x, even if it is unused.

```
function u0 = heatic(x)
u0 = 0.5;
end
```

Code Boundary Conditions

The standard form for the boundary conditions expected by the `pdepe` solver is

$$p(x,t,u) + q(x,t)f\left(x,t,u,\frac{\partial u}{\partial x}\right) = 0.$$

Written in this form, the boundary conditions for this problem are

$$u(0,t) + (0\cdot f) = 0,$$

$$(u(L,t) - 1) + (0\cdot f) = 0.$$

So the values for p and q are

- $p_L = u_L, \qquad q_L = 0.$
- $p_R = u_R - 1, \quad q_R = 0.$

The corresponding function is then

```
function [pl,ql,pr,qr] = heatbc(xl,ul,xr,ur,t)
pl = ul;
ql = 0;
pr = ur - 1;
qr = 0;
end
```

Select Solution Mesh

Use a spatial mesh of 20 points and a time mesh of 30 points. Since the solution rapidly reaches a steady state, the time points near $t = 0$ are more closely spaced together to capture this behavior in the output.

```
L = 1;
x = linspace(0,L,20);
t = [linspace(0,0.05,20), linspace(0.5,5,10)];
```

Solve Equation

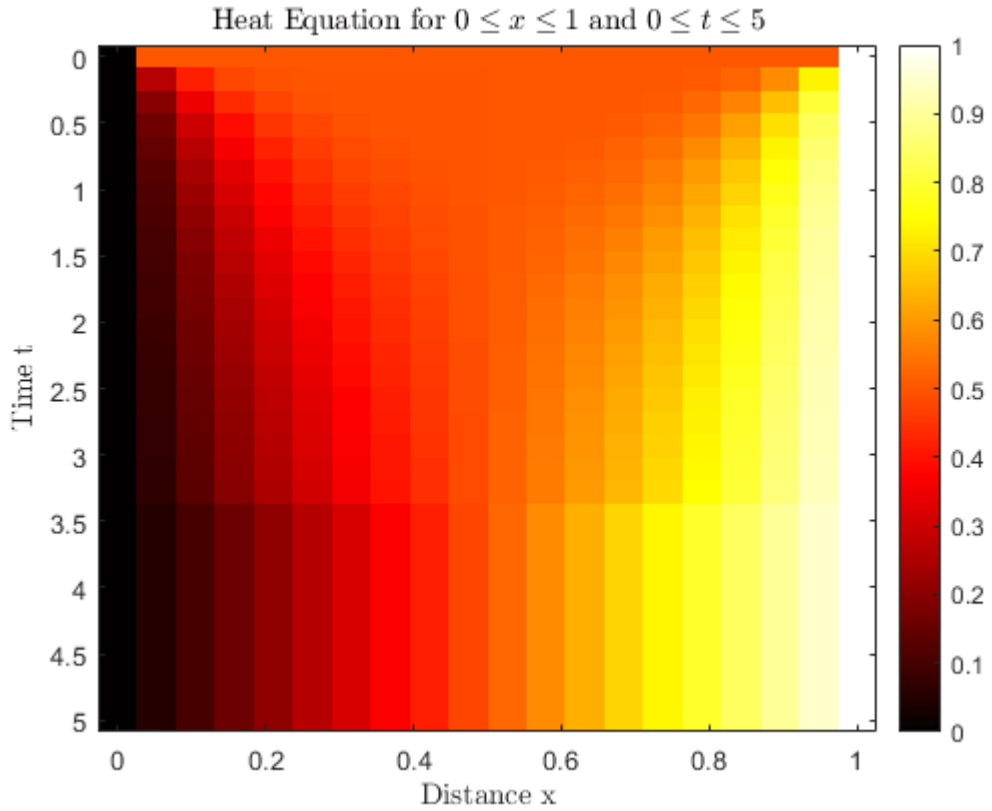
Finally, solve the equation using the symmetry m, the PDE equation, the initial condition, the boundary conditions, and the meshes for x and t.

```
m = 0;
sol = pdepe(m,@heatpde,@heatic,@heatbc,x,t);
```

Plot Solution

Use `imagesc` to visualize the solution matrix.

```
colormap hot
imagesc(x,t,sol)
colorbar
xlabel('Distance x','interpreter','latex')
ylabel('Time t','interpreter','latex')
title('Heat Equation for $0 \le x \le 1$ and $0 \le t \le 5$','interpreter','latex')
```



Local Functions

```
function [c,f,s] = heatpde(x,t,u,dudx)
c = 1;
```

```
f = dudx;
s = 0;
end
function u0 = heatic(x)
u0 = 0.5;
end
function [pl,ql,pr,qr] = heatbc(xl,ul,xr,ur,t)
pl = ul;
ql = 0;
pr = ur - 1;
qr = 0;
end
```

PDE Examples and Files

Several available example files serve as excellent starting points for most common 1-D PDE problems. To explore and run examples, use the Differential Equations Examples app. To run this app, type

```
odeexamples
```

To open an individual file for editing, type

```
edit exampleFileName.m
```

To run an example, type

```
exampleFileName
```

This table contains a list of the available PDE example files.

Example File	Description	Example Link
`pdex1`	Simple PDE that illustrates the formulation, computation, and plotting of the solution.	Solve Single PDE
`pdex2`	Problem that involves discontinuities.	Solve PDE with Discontinuity
`pdex3`	Problem that requires computing values of the partial derivative.	Solve PDE and Compute Partial Derivatives
`pdex4`	System of two PDEs whose solution has boundary layers at both ends of the interval and changes rapidly for small t.	Solve System of PDEs
`pdex5`	System of PDEs with step functions as initial conditions.	Solve System of PDEs with Initial Condition Step Functions

References

[1] Skeel, R. D. and M. Berzins, "A Method for the Spatial Discretization of Parabolic Equations in One Space Variable," *SIAM Journal on Scientific and Statistical Computing*, Vol. 11, 1990, pp. 1–32.

See Also

`bvp4c` | `ode45` | `odeset` | `pdepe` | `pdeval`

Related Topics

- Solve Single PDE
- Solve System of PDEs