Solution of a PDE With No Spatial Derivatives Using the `MATLAB pdepe` Function

Bill Greene

Version 0.2, September 24, 2018

A partial differential equation (PDE) is a differential equation with more than one independent variable. Usually a PDE contains derivatives of the dependent variables with respect to all of these independent variables. In this example, however, we show how the `MATLAB pdepe` function can be used to solve a PDE that has a derivative of its dependent variable with respect to time but not the spatial variable. The `pdepe` documentation implies that this is not possible but a simple boundary condition "trick" enables `pdepe` to handle this case.

A PDE of this form can be written

$$\frac{\partial u}{\partial t} = A(x) \tag{1}$$

with the accompanying initial condition

$$u(0, x) = B(x) \tag{2}$$

The solution to this equation is particularly simple

$$u(t, x) = A(x)t + B(x) \tag{3}$$

It is important to note that no boundary conditions are required for this solution because equation (1) has no derivatives with respect to x.

For this example we somewhat arbitrarily choose

$$A(x) = cos(3\pi x) \tag{4}$$
$$B(x) = sin(\pi x) \tag{5}$$

Many other smooth functions of x could also have been chosen for A and B.

1 `MATLAB` Code

1.1 PDE Definition

`pdepe` requires that the equations to be solved be defined in the following form

$$c(x, t, u, \frac{\partial u}{\partial x})\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f(x, t, u, \frac{\partial u}{\partial x})\frac{\partial u}{\partial x}\right) + s(x, t, u, \frac{\partial u}{\partial x}) \tag{6}$$

and that the user write a function with the following signature to return the coefficients in this equation

```
[c,f,s] = pdefun(x,t,u,dudx)
```

The details of the input and output arguments to this function are defined in the `pdepe` documentation. A `pdefun` function defining equation (1) is simply

```
c = 1;
f = 0;
s = A(x);
```

1.2 Boundary Conditions

`pdepe` requires that the boundary conditions be defined in the following form

$$p(x,t,u) + q(x,t)f(x,t,u,du/dx) = 0 \tag{7}$$

As mentioned above a trick in the definition of the boundary conditions is needed. Because the flux, f is always zero as specified in the PDE definition, a "do-nothing" boundary condition can be defined by setting $q = 1$ and $p = 0$.

2 MATLAB Code

The complete listing of the `MATLAB` code for this example is shown below.

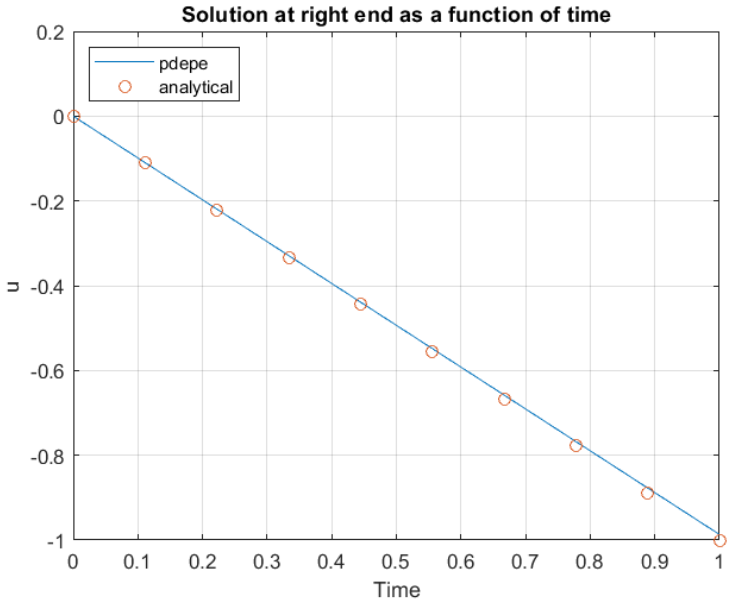
```
x = linspace(0,1,30);
t = linspace(0,1,10);

% define s coefficient in PDE
A = @(x) cos(3*pi*x);
% define initial condition
B = @(x) sin(pi*x);

% calculate analytical solution
ua=uAnal(x,t,A, B);

pdeFunc = @(x,t,u,DuDx) pde(x,t,u,DuDx,A);
icFunc = @(x) ic(x,B);
bcFunc = @(xl,ul,xr,ur,t) bc(xl,ul,xr,ur,t);
m=0;
u = pdepe(m, pdeFunc,icFunc,bcFunc,x,t);

err=ua(:)-u(:);
fprintf('Solution␣error:␣max=%12.3e,␣average=%12.3e\n', ...
  max(abs(err)), norm(err)/length(err));
figure; plot(t, u(:,end), t, ua(:,end), 'o'); grid on;
xlabel('Time'); ylabel('u');
legend('pdepe', 'analytical', 'Location','northwest' );
title('Solution␣at␣right␣end␣as␣a␣function␣of␣time');

figure; plot(t, u(:,1), t, ua(:,1), 'o'); grid on;
xlabel('Time'); ylabel('u');
legend('pdepe', 'analytical', 'Location','northwest' );
title('Solution␣at␣left␣end␣as␣a␣function␣of␣time');
```

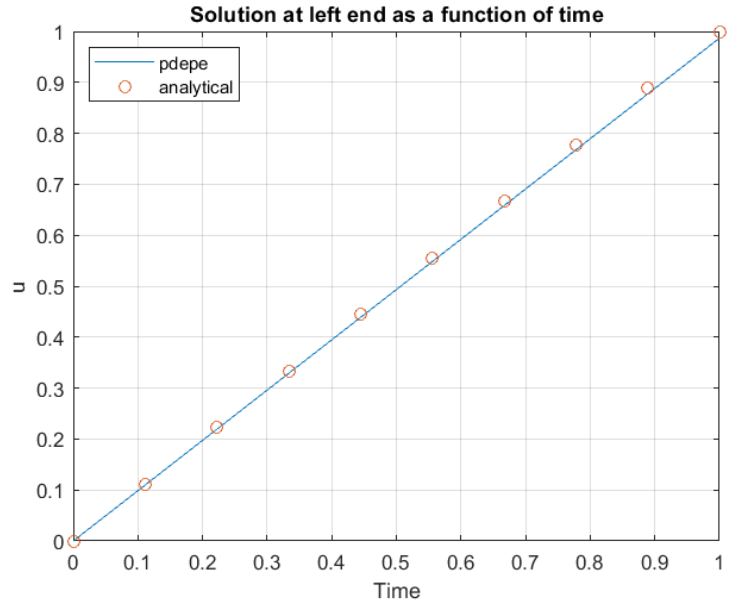
```
figure; plot(x, u(end,:), x, ua(end,:), 'o'); grid on;
xlabel('x'); ylabel('u');
legend('pdepe', 'analytical', 'Location','northwest' );
title('Solution along the length at final time');

function [c,f,s] = pde(x,t,u,DuDx,A)
nx=length(x);
c = ones(1,nx);
f = zeros(1,nx);
s = A(x);
end

function u0 = ic(x,B)
u0 = B(x);
end

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

function u=uAnal(x,t,A, B)
nt = length(t);
u=t'*A(x) + repmat(B(x),nt,1);
end

%% Copyright (C) 2018 William H. Greene
```

When this code is run in `MATLAB` , the following output is produced.

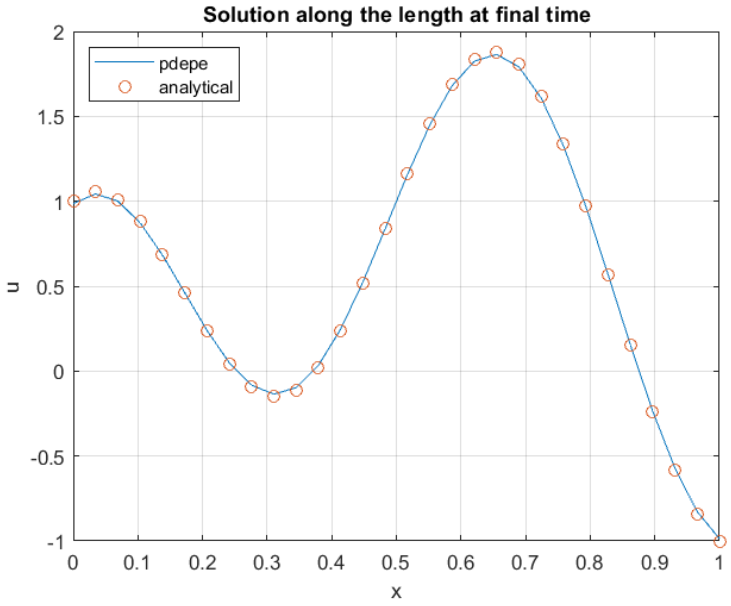
```
Solution error: max= 1.317e-02, average= 3.243e-04
```

The two figures below show the solution at the two ends as a function of time. The agreement between `pdepe` and analytical solutions is good indicating that the boundary conditions are working as expected.





The figure below shows the solution as a function of x at the final time. Again, the `pdepe` and analytical solutions agree well at all points along the length. In particular, the solution errors at the boundaries are roughly the same as those at the interior points.



As expected, the error can be reduced by choosing a finer spatial mesh (more points in the x-vector). The error also depends on the particular A and B functions chosen. In particular, if these functions are chosen simply as constants, there is no discretization in the spatial dimension and the error is on the order of machine precision.