Section 8.5: Partial Differential Equations

Created by Tomas de-Camino-Beck

Diffusion

This is the simple diffusion equation:

\[ \text{deqn} = \frac{\partial u}{\partial t} = d \frac{\partial^2 u}{\partial x^2} , \]

Let generate a simple pulse function using picewise equations,

\[ g[x_\text{, } \varepsilon_] := \begin{cases} 
0 & x < -1 \\
10^{-10} & -\varepsilon \leq x \leq \varepsilon \\
0 & x > 1 
\end{cases} \]

Now, let's run some numerical simulations using the following initial and boundary conditions:
1. Initial condition: \( u(x, 0) = g(x) \) (our picewise function)
2. Boundary conditions (absorbing): \( u(0, t) = 0 \) and \( u(10, t) = 0 \) note that our spatial domain \( L = \{-10, 10\} \)
3. Let's choose a value for the diffusion coefficient \( d \)

\[ d = 0.08; \]
\[ \text{sol} = \text{NDSolve}\{\{\text{deqn}, u[x, 0] = g[x, 1], u[-10, t] = 0, u[10, t] = 0\}, u, \{x, -10, 10\}, \{t, 0, 100\}\} \]

\text{NDSolve::mxst : Using maximum number of grid points 10000 allowed by the MaxPoints or MinStepSize options for independent variable x. More...}
let's look at a plot of different times:

\begin{verbatim}
Plot[Evaluate[{u[x, 10] /. sol[[1]], u[x, 50] /. sol[[1]], u[x, 100] /. sol[[1]]}], {x, -10, 10}, PlotRange -> All, Frame -> True]
\end{verbatim}

It can be seen from here that the numerical solution is a gaussian, just as the analytical solution. In the stochastic model chapter, a stochastic process, simulating random walk, produces the same numerical result. However, deterministic walks can also generate diffusion (Wolfram 2002)
Fisher's Equation

Fisher's equations models a population with density dependent growth and diffusion in a one dimensional space. Nota the the equation is the same as diffusion, but now we add a term \( \mu u(1 - u) \) which is logistic growth (non-dimensional), with growth rate \( \mu \). See Murray (1993)

\[
deqn = D[u[x, t], t] = D D[u[x, t], x, x] + \mu u[x, t] (1 - u[x, t]);
\]

\[
TraditionalForm[deqn]
\]

\[
u^{(0,1)}(x, t) = \mu(1 - u(x,t))u(x, t) + d u^{(2,0)}(x, t)
\]

Let's use a similar pulse function:

\[
Clear[g];
g[x_, \varepsilon_] := \begin{cases} 
0 & x \leq -1 \\
0.001 & -\varepsilon \leq x \leq \varepsilon \\
0 & x \geq 1 
\end{cases}
\]

The initial and boundary conditions are the same:

\[
d = 0.08; \mu = 2;
sol = NDSolve[{deqn, u[x, 0] = g[x, 1], u[-10, t] = 0, u[10, t] = 0, u, {x, -10, 10}, {t, 0, 100}]
\]

NDSolve::mxst : Using maximum number of grid points 10000
allowed by the MaxPoints or MinStepSize options for independent variable x. More...

\[
[[u \rightarrow InterpolatingFunction][{-10., 10.}, {0., 100.}], <>]]
\]

\[
Plot3D[Evaluate[u[x, t] /. sol[[1]]], {x, -10, 10}, {t, 0, 10}, PlotPoints -> 40, Mesh -> False, PlotRange -> Automatic]
\]

Let look at different plot for different times, so we can see the front of the travelling wave:
\begin{verbatim}
Plot[Evaluate[
  {u[x, 8] /. sol[[1]], u[x, 9] /. sol[[1]], u[x, 10] /. sol[[1]], u[x, 11] /. sol[[1]]},
  {x, -10, 10}, PlotRange -> All, Frame -> True]
\end{verbatim}

--- Graphics ---

References
