Section 8.5: Partial Differential Equations

Created by Tomas de-Camino-Beck

Diffusion

This is the simple diffusion equation:

```
deqn = D[u[x, t], t] == d D[u[x, t], x, x];
```

```
TraditionalForm[deqn]
```

$u^{(0,1)}(x, t) = d\, u^{(2,0)}(x, t)$

Let generate a simple pulse function using picewise equations,

$$g[x\_, \epsilon\_] := \begin{cases} 0 & x < -1 \\ 10^{10} & -\epsilon \le x \le \epsilon \\ 0 & x > 1 \end{cases}$$

Now, lets run some numerical simulations using the folowing initial and boundary conditions:

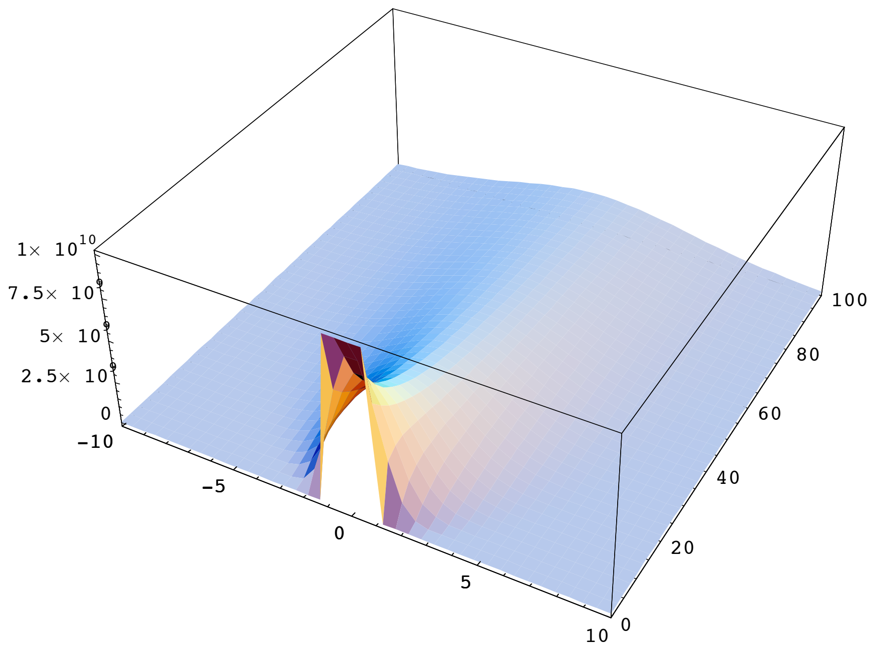
1. Initial condition: $u(x, 0) = g(x)$ (our picewise function)
2. Boundary conditions (absorbing): $u(0, t) = 0$ and $u(10, t) = 0$ note that our spatial domain $L = \{-10, 10\}$
3. lets choose a value for the diffusion coefficient d

```
d = 0.08;
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::mxsst : Using maximum number of grid points 10000
   allowed by the MaxPoints or MinStepSize options for independent variable x. More…
```

```
{{u → InterpolatingFunction[{{-10., 10.}, {0., 100.}}, <>]}}
```

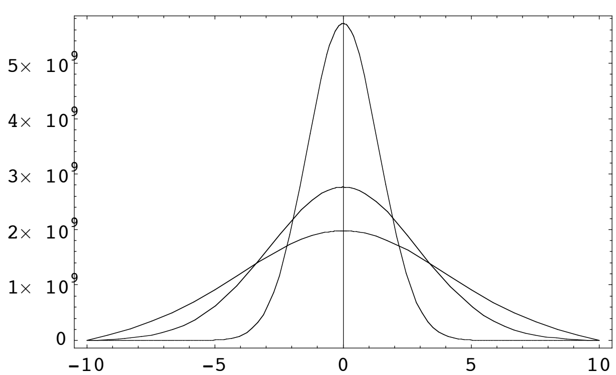
```
Plot3D[Evaluate[u[x, t] /. sol[[1]]], {x, -10, 10},
 {t, 0, 100}, PlotPoints → 40, Mesh → False, PlotRange → All]
```




```
- SurfaceGraphics -
```

Lets loos at a plot of different times:

```
Plot[Evaluate[{u[x, 10] /. sol[[1]], u[x, 50] /. sol[[1]], u[x, 100] /. sol[[1]]}],
 {x, -10, 10}, PlotRange → All, Frame → True]
```




```
- Graphics -
```

It can be seen form 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 $\mu\, u(1-1)$ which is logistic growth (non-dimensional), with growth rate μ. See Murray (1993)

```
deqn = D[u[x, t], t] == d D[u[x, t], x, x] + μ u[x, t] (1 - u[x, t]);
```

```
TraditionalForm[deqn]
```

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

Lets use a similar pulse function:

```
Clear[g];
g[x_, ε_] := ⎧ 0       x < -1
             ⎨ 0.001   -ε ≤ x ≤ ε
             ⎩ 0       x > 1
```

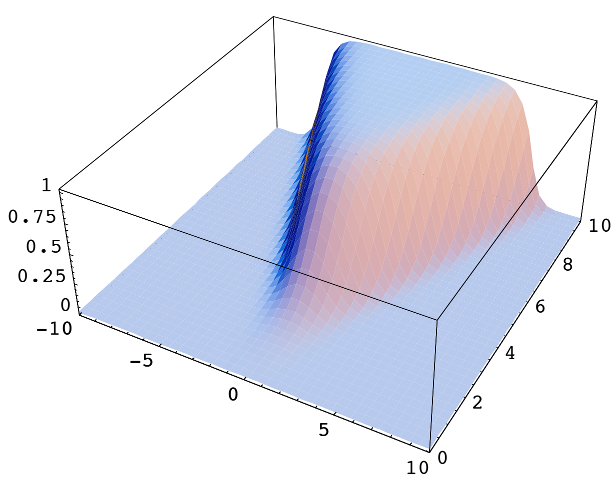
The initial and boundary conditions are the same:

```
d = 0.08; μ = 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::mxsst : Using maximum number of grid points 10000
   allowed by the MaxPoints or MinStepSize options for independent variable x. More…
```

```
{{u → InterpolatingFunction[{{-10., 10.}, {0., 100.}}, <>]}}
```

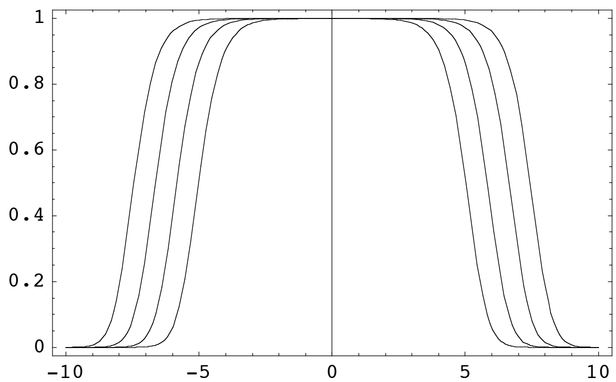
```
Plot3D[Evaluate[u[x, t] /. sol[[1]]], {x, -10, 10},
 {t, 0, 10}, PlotPoints → 40, Mesh → False, PlotRange → Automatic]
```




```
- SurfaceGraphics -
```

Let look at different plot for different times, so we can see the front of the travelling wave:

```
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]
```



- Graphics -

References

Murray J.D. 1993. Mathematical biology, 2nd, corr. ed edn. Springer-Verlag, Berlin, New York .

Wolfram, S. 2002. A new kind of science. : Wolfram Media, Champaign, IL.