How to | Solve a Partial Differential Equation

Mathematica's differential equation solving functions can be applied to many different classes of differential equations, automatically selecting the appropriate algorithms without the need for preprocessing by the user. One such class is partial differential equations (PDEs).

Using D to take derivatives, this sets up the $c = 2$ transport equation, $2\frac{\partial y(x,t)}{\partial x} + \frac{\partial y(x,t)}{\partial t} = 0$, and stores it as pde:

```
In[14]:= pde = D[y[x, t], t] + 2 D[y[x, t], x] == 0
Out[14]= y^(0,1)[x, t] + 2 y^(1,0)[x, t] == 0
```

Use DSolve to solve the equation and store the solution as soln. The first argument to DSolve is an equation, the second argument is the function to solve for, and the third argument is a list of the independent variables:

```
In[15]:= soln = DSolve[pde, y[x, t], {x, t}]
Out[15]= {{y[x, t] → C[1][1/2 (2 t - x)]}}
```

The answer is given as a rule and C[1] is an arbitrary function.

To use the solution as a function, say f[x, t], use /. (the short form of ReplaceAll) and [[...]] (the short form of Part):

```
In[16]:= f[x_, t_] = y[x, t] /. soln[[1]]
Out[16]= C[1][1/2 (2 t - x)]
```

You can then evaluate f[x, t] like any other function:

```
In[17]:= f[x, 0]
Out[17]= C[1][-x/2]
```

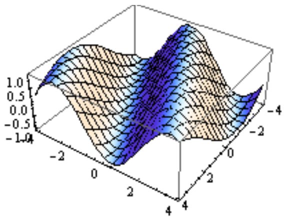
You can also add an initial condition like $y(0, t) = \sin(t)$ by making the first argument to DSolve a list. The solution is stored as sol:

```
In[18]:= sol = DSolve[{pde, y[0, t] == Sin[t]}, y[x, t], {x, t}]
Out[18]= {{y[x, t] → Sin[1/2 (2 t - x)]}}
```

Use Plot3D to plot the solution:

```
In[20]:= Plot3D[y[x, t] /. sol, {x, -4, 4}, {t, -4, 4}]
```

Out[20]=



Use DSolve with the inhomogeneous PDE $2\frac{\partial y(x,t)}{\partial x}+\frac{\partial y(x,t)}{\partial t}=\sin(x)$ with the initial condition $y(0,t)=\cos(t)$:

```
In[1]:= sol1 = DSolve[
          {D[y[x, t], t] + 2 D[y[x, t], x] == Sin[x], y[0, t] == Cos[t]}, y[x, t], {x, t}]
```

Out[1]= $\left\{\left\{y[x,t]\to\frac{1}{2}\left(1+2\,\mathrm{Cos}\left[\frac{1}{2}(2t-x)\right]-\mathrm{Cos}[x]\right)\right\}\right\}$

Get just the solution from the nested list:

```
In[2]:= sol2 = sol1[[1, 1, 2]]
```

Out[2]= $\frac{1}{2}\left(1+2\,\mathrm{Cos}\left[\frac{1}{2}(2t-x)\right]-\mathrm{Cos}[x]\right)$

Evaluate the solution for given values of the parameters:

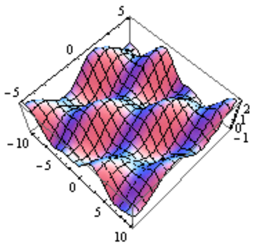
```
In[3]:= sol2 /. {t → 1, x → 2}
```

Out[3]= $\frac{1}{2}(3-\mathrm{Cos}[2])$

Now, use Plot3D to plot the solution:

```
In[5]:= Plot3D[sol2, {x, -10, 10}, {t, -5, 5}]
```

Out[5]=



You can also work with PDEs that have non-numeric coefficients.

Use DSolve to solve a inhomogeneous PDE, for example, $a\frac{\partial y(x,t)}{\partial x}+b\frac{\partial y(x,t)}{\partial t}=c\sin(x)$ with the initial condition $y(0,t)=a\cos(t)$. The solution is stored as pdesol:

```
In[6]:= pdesol = DSolve[{b D[y[x, t], t] + a D[y[x, t], x] == c Sin[x], y[0, t] == a Cos[t]},
          y[x, t], {x, t}]
```

Out[6]= $\left\{\left\{y[x,t]\to\frac{c-c\,\mathrm{Cos}[x]+a^2\,\mathrm{Cos}\left[\frac{a\,t-b\,x}{a}\right]}{a}\right\}\right\}$

Define a function Fsol, corresponding to the solution pdesol:

```
In[7]:= Fsol[x_, t_] = (y[x, t] /. pdesol)[[1]]
```

Out[7]= $\frac{c-c\,\mathrm{Cos}[x]+a^2\,\mathrm{Cos}\left[\frac{a\,t-b\,x}{a}\right]}{a}$

Evaluate the solution function for given values of the parameters:

```
In[8]:= Fsol[2, 1]
```

Out[8]= $\frac{c-c\,\mathrm{Cos}[2]+a^2\,\mathrm{Cos}\left[\frac{a-2\,b}{a}\right]}{a}$

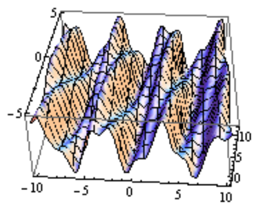
Substitute values for the parameters:

```
In[9]:= Fsol[2, 1] /. {a → 2, b → 5, c → 10}
```

```
Out[9]= 1/2 (10 - 10 Cos[2] + 4 Cos[4])
```

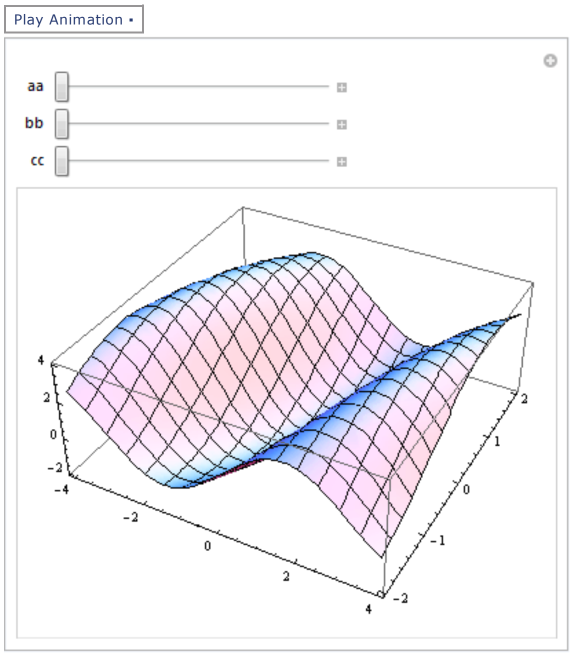
Plot the solution `Fsol` for a given set of values of parameters:

```
In[10]:= Plot3D[Fsol[x, t] /. {a → 2, b → 5, c → 10}, {x, -10, 10}, {t, -5, 5}]
Out[10]=
```



Use `Manipulate` to show how the solution `Fsol` changes with respect to the parameters `a`, `b`, and `c`:

```
In[11]:= Manipulate[Plot3D[Fsol[x, t] /. {a → aa, b → bb, c → cc}, {x, -4, 4}, {t, -2, 2}],
          {aa, -2, 2}, {bb, -2, 2}, {cc, -2, 2}, SaveDefinitions → True]
Out[11]=
```



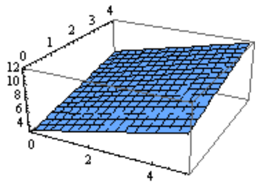
The examples so far use `DSolve` to obtain symbolic solutions to PDEs. When a given PDE does not contain parameters, `NDSolve` can be used to obtain numerical solutions. The results of `NDSolve` are given as `InterpolatingFunction` objects.

Here, the solution produced by `NDSolve` is stored as `nsol1`:

```
In[12]:= nsol1 =
          NDSolve[{D[y[x, t], t] + 2 D[y[x, t], x] == 3 , y[x, 0] == x + 3 , y[5, t] == t + 8},
           y[x, t], {x, 0, 5}, {t, 0, 4}]
Out[12]= {{y[x, t] → InterpolatingFunction[{{0., 5.}, {0., 4.}}, <>][x, t]}}
```

Plot the solution with `Plot3D`:

```
In[15]:= Plot3D[nsol1[[1, 1, 2]], {x, 0, 5}, {t, 0, 4}]
Out[15]=
```



The `InterpolatingFunction` object can be evaluated, plotted, and used in other operations.

Get just the InterpolatingFunction solution from nsol1 and assign it to the new symbol nsol2:

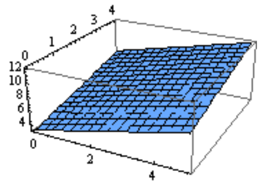
```
In[13]:= nsol2 = nsol1[[1, 1, 2]]
Out[13]= InterpolatingFunction[{{0., 5.}, {0., 4.}}, <>][x, t]
```

Evaluate the solution with values specified for x and t:

```
In[14]:= nsol2 /. {x → 1/2, t → 1}
Out[14]= 4.5
```

Plot the solution nsol2 with Plot3D:

```
In[24]:= Plot3D[nsol2, {x, 0, 5}, {t, 0, 4}]
Out[24]=
```



When the PDE contains parameters, NDSolve can be used for each specific value of the parameters. In addition, you can set up a function that uses NDSolve and takes parameter values.

Use ?NumericQ to prevent the function fsol from evaluating for non-numeric values of the parameter:

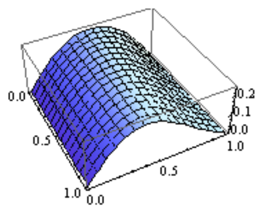
```
In[7]:= fsol[k_?NumericQ] := NDSolve[{Derivative[2, 0][u][x, t] x^4 ==
            x^3 t Cos[x Sin[t]] / (1/10 + Sin[t]) + k Derivative[0, 1][u][x, t],
          u[0, t] == 0, u[1, t] == 0, u[x, 0] == x (1 - x)}, u[x, t], {x, 0, 1}, {t, 0, 1}]
```

Find the solution corresponding to a specific value of the parameter, 5 in this case:

```
In[9]:= fsol[5]
Out[9]= {{u[x, t] → InterpolatingFunction[{{0., 1.}, {0., 1.}}, <>][x, t]}}
```

Plot the solution using Plot3D. Evaluate is necessary so that the evaluations occur in the correct order:

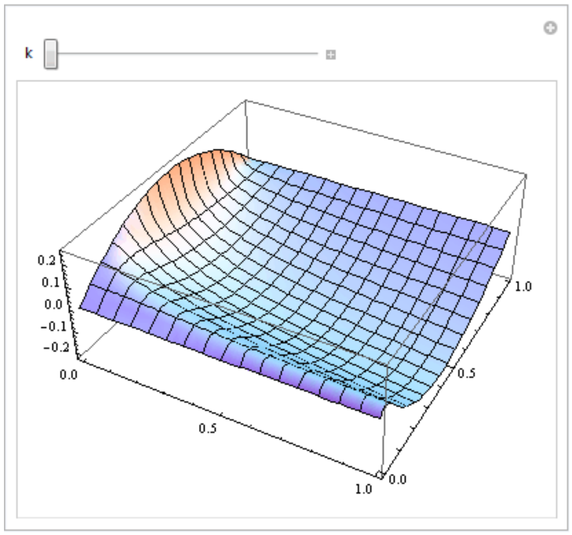
```
In[10]:= Plot3D[Evaluate[u[x, t] /. fsol[5][[1]]], {t, 0, 1}, {x, 0, 1}, PlotRange → All]
Out[10]=
```



NDSolve can also be used with other *Mathematica* functions, like Manipulate.

Solve the PDE corresponding to a given value of the parameter k, and then plot the resulting solution:

```
In[11]:= Manipulate[Plot3D[Evaluate[u[x, t] /. fsol[k][[1]]], {t, 0, 1},
           {x, 0, 1}, PlotRange -> All], {k, 1/100, 5}, SaveDefinitions → True]
Out[11]= Play Animation ▸
```



Tutorials

Differentiation

Differential Equations

Differential Equation Solving with DSolve

Advanced Numerical Differential Equation Solving in *Mathematica*

Related Demonstrations

A Passive Cochlear Model

Brownian Motion in 2D and the Fokker-Planck Equation

Heat Conduction in Some Standard Solids

Convection-Diffusion in a Semi-Infinite Region

Related Links

How to: Solve an Equation
How to: Plot the Results of NDSolve
How to: Work with Differential Equations

See Also

DSolve ▪ **NDSolve** ▪ **InterpolatingFunction** ▪ **Plot3D** ▪ **Manipulate** ▪ **Part** ▪ **ReplaceAll** ▪ **NumericQ**

Related Guides

Differential Equations
"How to" Topics

Give Feedback