# Solitons in the Korteweg-de Vries Equation (KdV Equation)

## Introduction

The Korteweg-de Vries Equation (KdV equation) describes the theory of water waves in shallow channels, such as a canal. It is a non-linear equation which exhibits special solutions, known as *solitons,* which are stable and do not disperse with time. Furthermore there as solutions with more than one soliton which can move towards each other, interact and then emerge at the same speed with no change in shape (but with a time "lag" or "speed up").

The KdV equation is

$$\frac{\partial u}{\partial t} = 6\,u\,\frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3}$$

Because of the $u\,\partial u/\partial x$ term the equation is non-linear (this term increases four times if u is doubled).

## One soliton solution

The simplest soliton solution is

$$u(x, t) = -2\,\mathrm{sech}^2(x - 4\,t),$$

which is a trough of depth 2 traveling to the right with speed 4 and not changing its shape.

Let us verify that it does satisfy the equation:

```
In[4]:= uexact[x_, t_] = -2 Sech[x - 4 t]^2

Out[4]= -2 Sech[4 t - x]^2

In[5]:= D[uexact[x, t], t] == 6 uexact[x, t] D[uexact[x, t], x] - D[uexact[x, t], {x, 3}] // Simplify

Out[5]= True
```

*Mathematica* returns **True**, indicating that equation is satisfied.

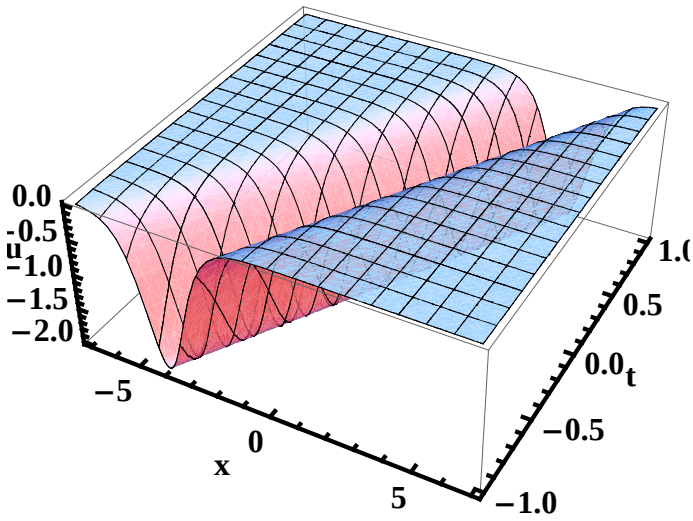
*Mathematica* function **NDSolve** can solve partial differential equations in two (but not more than two) variables, such as x and t. However, it tends to be very slow and require a lot of memory. Nonetheless, if we put in the soliton at the initial time, it correctly propagates the soliton in time:

```
In[6]:= xmin = -8; xmax = 8;
        sol = NDSolve[{D[u[x, t], t] == 6 u[x, t] D[u[x, t], x] - D[u[x, t], {x, 3}],
           u[x, 0] == -2 Sech[x]^2, u[xmin, t] == u[xmax, t]}, u, {x, xmin, xmax}, {t, -1, 1}]

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

Out[6]= {{u → InterpolatingFunction[{{-8., 8.}, {-1., 1.}}, <>]}}
```

Plotting the solution shows the trough propagating to the right.

In[7]:=
```
Plot3D[u[x, t] /. Flatten[sol], {x, -7, 7}, {t, -1, 1},
 PlotPoints → 50, PlotRange → All, AxesLabel → {"x", "t", "u"}]
```

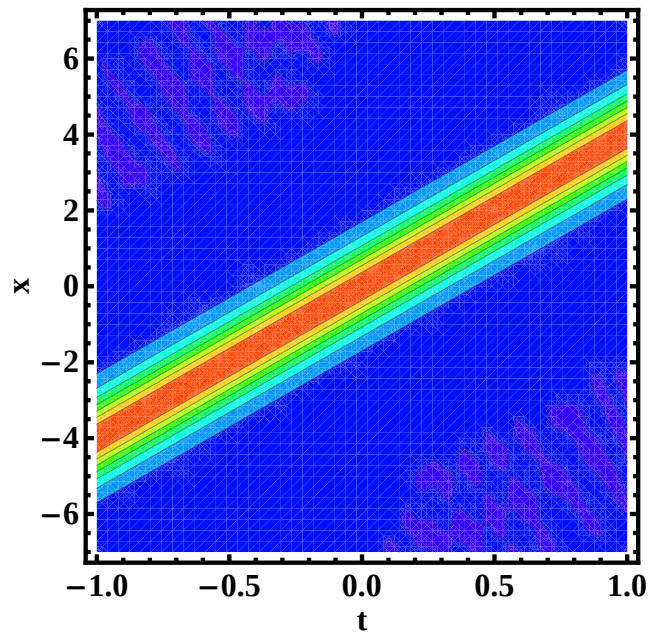
Out[7]=




A contour plot can also be useful:

In[8]:=
```
ContourPlot[u[x, t] /. Flatten[sol], {t, -1, 1}, {x, -7, 7},
 ColorFunction → (Hue[0.7 # 1] &), PlotPoints → 50, PlotRange → All, FrameLabel → {"t", "x"}]
```

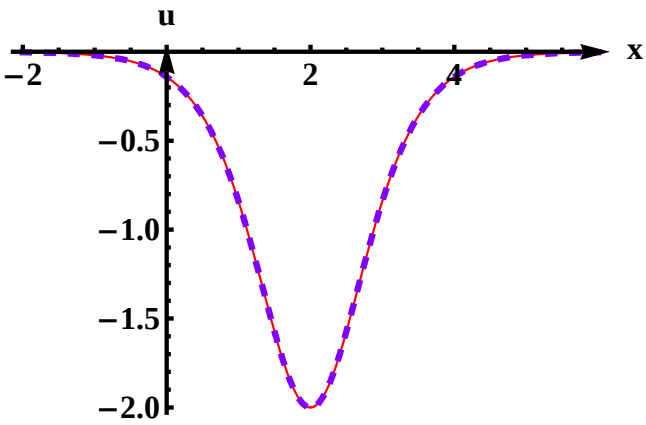
Out[8]=




To verify that the numerical solution is the soliton, we plot both for a particular value of t (t = 0.5 here):

In[9]:=
```
Plot[{u[x, 0.5] /. Flatten[sol], -2 Sech[x - 2]^2},
 {x, -2, 6}, PlotStyle → {{Hue[0], AbsoluteThickness[1]},
   {Hue[0.75], Dashing[{0.01, 0.03}], AbsoluteThickness[3]}}, AxesLabel → {"x", "u"}]
```

Out[9]=




We see that the two agree very well.

In fact there is a whole family of 1-soliton solutions parametrized by the depth of the trough. These are

$$u(x, t) = -x_{max}\,\mathrm{sech}^2\left[\sqrt{\frac{x_{max}}{2}}\,(x - 2\,x_{max}\,t)\right],$$

so the deeper the trough the faster the soliton moves and the narrower it is. We verfiy that this does satisfy the KdV equation:

In[10]:=
```
Clear[xmax]
```

In[11]:=
```
uexact[x_, t_] = -xmax Sech[Sqrt[xmax / 2] (x - 2 xmax t)]^2
```

Out[11]=
$$-\text{xmax}\,\text{Sech}\left[\frac{\sqrt{\text{xmax}}\,(x - 2\,t\,\text{xmax})}{\sqrt{2}}\right]^2$$

In[12]:=
```
D[uexact[x, t], t] == 6 uexact[x, t] D[uexact[x, t], x] - D[uexact[x, t], {x, 3}] // Simplify
```

Out[12]=
```
True
```

## ■ Two soliton solution

The theory for solutions with more than one soliton is complicated and we will not discuss it, but rather just display a two-soliton solution, verify that it is indeed a solution, and look at its properties. Specifying adequate resolution and number of time steps, my computer ran out of memory.

The theory states that an initial state

$$u(x, 0) = -n(n+1)\,\mathrm{sech}^2(x),$$

results in n solitons that propagate with different velocities. The solution for n = 2 is

$$u(x, t) = -12\,\frac{3 + 4\cosh(2x - 8t) + \cosh(4x - 64t)}{[3\cosh(x - 28t) + \cosh(3x - 36t)]^2}$$

(You may want to verify that this reduces to -6sech$^2 x$ for t = 0.)

It is not immediately evident that the above expression for u(x, t) satisfies the KdV equation, but *Mathematica* confirms that it does:

In[13]:=
```
uexact[x_, t_] = -12 (3 + 4 Cosh[2 x - 8 t] + Cosh[4 x - 64 t]) / (3 Cosh[x - 28 t] + Cosh[3 x - 36 t])^2
```

Out[13]=
$$-\frac{12\,(3+\text{Cosh}[64\,t-4\,x]+4\,\text{Cosh}[8\,t-2\,x])}{(\text{Cosh}[36\,t-3\,x]+3\,\text{Cosh}[28\,t-x])^2}$$

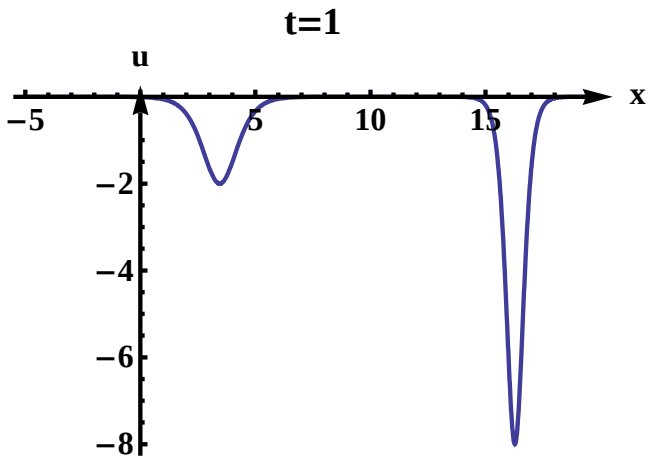
In[14]:=
```
D[uexact[x, t], t] == 6 uexact[x, t] D[uexact[x, t], x] - D[uexact[x, t], {x, 3}] // Simplify
```

Out[14]= True

Next we plot the solution at time t = 1:

In[15]:=
```
Plot[uexact[x, 1], {x, -5, 20}, PlotRange → All, PlotLabel → "t=1", AxesLabel → {"x", "u"}]
```

Out[15]=




We see a trough of depth 8 and a trough of depth 2. To determine the speeds of these troughs we locate the minima of the function at two different times, t=2 and 3,

In[16]:=
```
FindMinimum[uexact[x, 2], {x, 10}]
```

Out[16]= {-2., {x → 7.45069}}

In[17]:=
```
FindMinimum[uexact[x, 3], {x, 10}]
```

Out[17]= {-2., {x → 11.4507}}

The last two results show that trough of depth 2 travels with speed 2.

In[18]:=
```
FindMinimum[uexact[x, 2], {x, 30}]
```

```
FindMinimum::lstol :
 The line search decreased the step size to within tolerance specified
   by AccuracyGoal and PrecisionGoal but was unable to find a sufficient
   decrease in the function.  You may need more than MachinePrecision
   digits of working precision to meet these tolerances. >>
```

Out[18]= {-8., {x → 32.2747}}

In[19]:=
```
FindMinimum[uexact[x, 3], {x, 50}]
```

Out[19]= {-8., {x → 48.2747}}

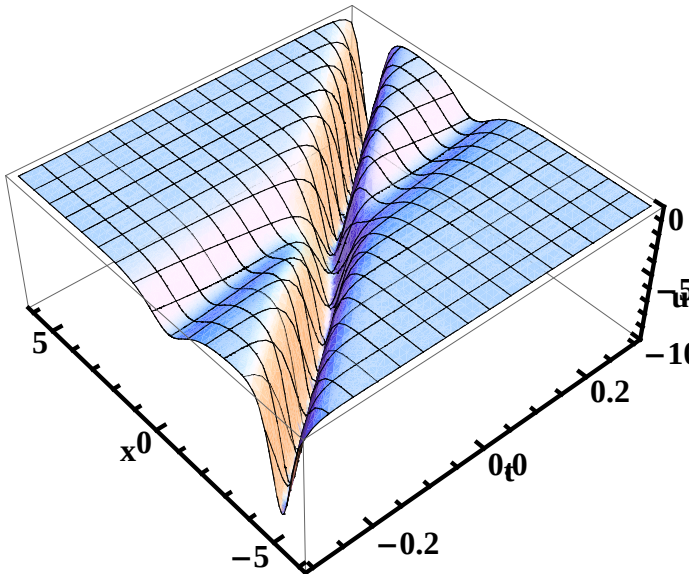
The last two results show that trough of depth 8 travels with speed 16. Thus we have created two solitons of the type that we discussed in the previous section. Note, however, that there is no linear superposition (because the equation is non-linear), so the 2-soliton solution is *not* the sum of the two individual solitons in the region

where they overlap, as one can see from the explicit solutions.

Let’s now see these two solitons interact in the vicninity of t = 0. We do a 3D plot,

In[20]:=
```
Plot3D[uexact[x, t], {t, -0.3, 0.3}, {x, -6, 6}, PlotPoints → 50,
 PlotRange → {-10, 0}, AxesLabel → {"t", "x", "u"}, ViewPoint → {-1.78, -2.06, 2.5}]
```

Out[20]=

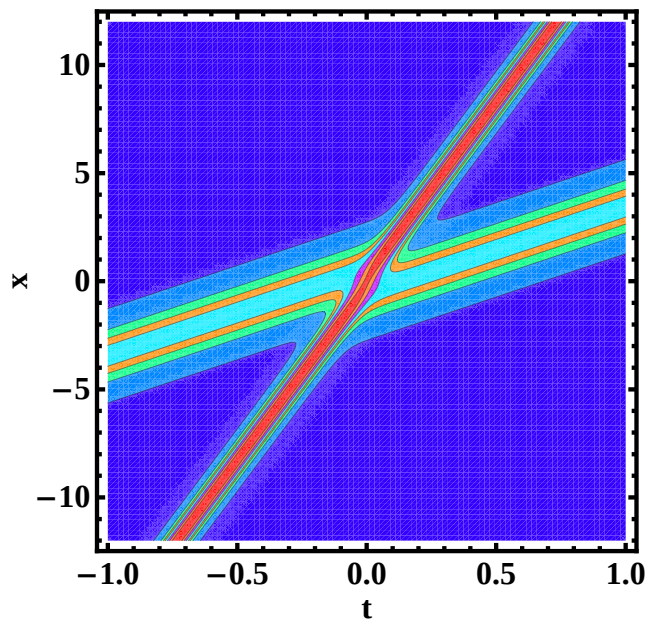



At negative times, the deeper soliton, which moves faster, approaches the shallower one. At t = 0 they combine to give u(x, 0) = $-6\,\mathrm{sech}^2(x)$, (a single trough of depth 6) and, after the encounter, the deeper soliton has overtaken the shallower one and both resume their original shape and speed. However, as a result of the interaction, the shallower soliton experiences a delay and the deeper soliton is speeded up.

This is also easily seen in a contour plot

In[21]:=
```
ContourPlot[uexact[x, t], {t, -1, 1}, {x, -12, 12},
 PlotPoints → 100, FrameLabel → {"t", "x"}, ColorFunction → (Hue[0.7 #1] &),
 PlotRange → All, Contours → {-0.1, -0.6, -1.1, -1.6, -2.6, -4., -5.6}]
```

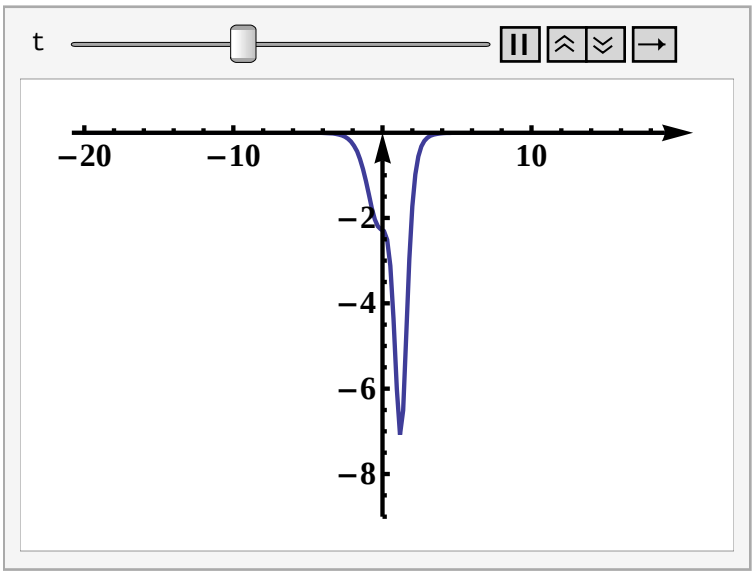
Out[21]=



Finally we show an animation of the two solitons crossing each other.

In[22]:=
```
Animate[Plot[uexact[x, t], {x, -20, 20}, PlotRange → {-9, 0}], {t, -1, 1, 0.02}]
```

Out[22]=




## ■ Other solutions

Now suppose that the initial condition is such that is does not just produce one or more solitons. We will take

$$u(x, 0) = -4\,\mathrm{sech}^2(x),$$

In[23]:=
```
xmin = -12; xmax = 12;
sol = NDSolve[{D[u[x, t], t] == 6 u[x, t] D[u[x, t], x] - D[u[x, t], {x, 3}],
   u[x, 0] == -4 Sech[x]^2, u[xmin, t] == u[xmax, t]}, u, {x, xmin, xmax}, {t, 0, 1}]
```

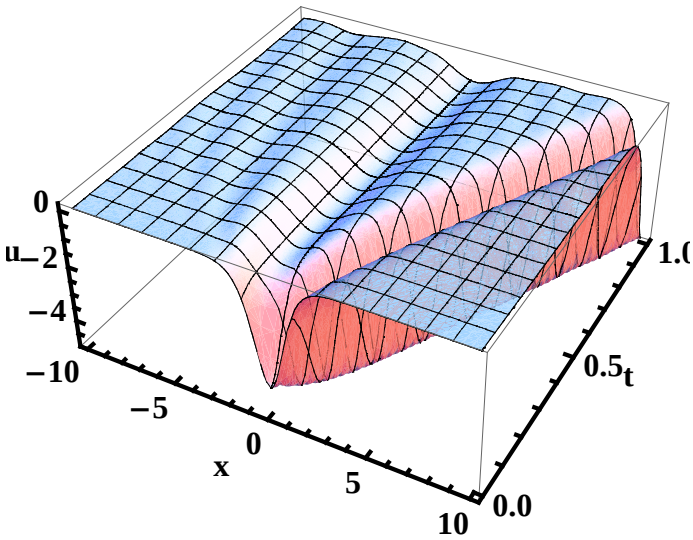
```
NDSolve::eerr :
 Warning: Scaled local spatial error estimate of 19.795807677713334‘ at t = 1.‘ in the
   direction of independent variable x is much greater than prescribed error
   tolerance. Grid spacing with 451 points may be too large to achieve the desired
   accuracy or precision.  A singularity may have formed or you may want to specify
   a smaller grid spacing using the MaxStepSize or MinPoints method options. >>
```

Out[23]=
```
{{u → InterpolatingFunction[{{-12., 12.}, {0., 1.}}, <>]}}
```

Ignoring the warning messages, we plot the solution:

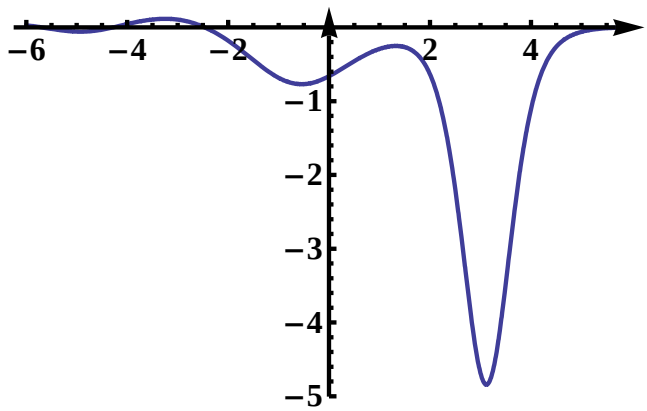
In[24]:=
```
Plot3D[u[x, t] /. Flatten[sol], {x, -10, 10}, {t, 0, 1},
 PlotPoints → 50, PlotRange → All, AxesLabel → {"x", "t", "u"}]
```

Out[24]=




In[25]:=
```
Plot[u[x, 0.3] /. Flatten[sol], {x, -6, 6}, PlotRange → All]
```

Out[25]=




In[26]:=
```
FindMinimum[u[x, 0.3] /. Flatten[sol], {x, 2.5, 3}]
```

Out[26]= {−4.84161, {x → 3.11717}}

In[27]:=
```
FindMinimum[u[x, 0.2] /. Flatten[sol], {x, 1.7, 2.2}]
```

Out[27]= {−4.76433, {x → 2.13993}}

The peak moving to the right has a depth of about 5 and a speed of about 10, and is a soliton of the family discussed in the first section. In addition, there are waves moving to the left. These will disperse and lose their form with time.

Finally we show an animation of this solution, starting from t = 0 and going up to t = 1.

In[28]:=
```
Animate[Plot[u[x, t] /. Flatten[sol], {x, -5, 10}, PlotRange → {-6, 0}], {t, 0, 0.8, 0.01}]
```

Out[28]=