# SIG in Mathematical Challenges of Nonlinear Waves and Interfacial Dynamics

MATLAB Tutorial by E. Renzi and D. N. Sibley (Loughborough University)

The purpose of this Tutorial is for you to familiarise with the use of MATLAB to solve mathematical problems and to gain experience of presenting your results in a professional manner.

In this tutorial, you will learn:

- To define variables and arrays
- To perform simple algebraic operations
- To plot functions in 2D and 3D
- To find the numerical solution of differential equations

## Collision of two solitary waves

Wave phenomena are often encountered in science. They may be first introduced as waves on a string, or perhaps in the stretched membrane of a drum, or indeed via electromagnetic waves. In this question you will explore a curious wave phenomenon that occurs in shallow water, first observed by J. Scott Russell on the Edinburgh-Glasgow canal in 1834. He was observing the motion of a boat being drawn rapidly along a channel by two horses, when the boat suddenly stopped, creating a large solitary wave of water, which continued along the channel apparently without change of form and at constant speed. In fact, Russell managed to follow the wave for two whole miles on horseback.
Much later – during the second half of the 20th century – it was shown that the collision of two such solitary waves can be described by the function

$$u(x,t) = 12\frac{3 + 4\cosh(2x - 8t) + \cosh(4x - 64t)}{\big(3\cosh(x - 28t) + \cosh(3x - 36t)\big)^2}, \tag{1}$$

where $u(x,t)$ is the amplitude of the system of two waves at point $x$ and time $t$.

**Part 1 - Basic operations**

(a) Let us first study the system at time $t = 0$. Define the MATLAB variable "`t`" accordingly.

(b) Define an **array** " `X` " of spatial coordinates $x \in (-20, 20)$, with spacing 0.01. Then define "`u0`" to calculate $u(x,0)$ with the equation above by using **element by element** operations.

(c) Since $t = 0$ is fixed, $u(x,0)$ is a function of one variable, i.e. $x$. **Plot** $u(x,0)$ against $x$. Add a title, labels and a grid to your plot.

(d) Now define two other instants, $t_1 = -1$, $t_2 = 1$. Define two arrays, "`u1`" and "`u2`" to calculate, respectively, $u(x,-1)$, $u(x,1)$ by using the function $u(x,t)$ of equation (1), for $x \in (-20, 20)$.

(e) On a new figure, plot $u(x,-1)$, $u(x,0)$ and $u(x,1)$ versus $x$ on the **same** plot. Your graph should be **fully** labelled and titled, with a **legend** to distinguish between the three functions.

(f) On a new figure, plot $u(x,-1)$, $u(x,0)$ and $u(x,1)$ versus $x$ on **three inline** plots.

**Part 2 - "For" loops, advanced input and output**

(a) Let us now introduce a smarter way to input variables. Create an **input dialogue window** to enter the variables "`t0`", "`t1`" and "`t2`". Run the program again to see the changes.

(b) Now let us study the behaviour of the system with time. Create an input dialogue window to insert an initial time "`ti`", a time increment "`dt`" and a final time "`tf`". Create an array " `T` " with the values from $t_i$ to $t_f$ with increment $dt$.

(c) Use a **for** loop to calculate Eq. (1) at all points $X$ and instants $T$. On a new figure, plot a space-time **contour plot** of the function $u(x,t)$.

**Part 3 - Numerical methods - Solution of differential equations**

In this part, we wish to numerically solve the Korteweg-de Vries equation (KdV) equation which is able to model certain types of waves. The equation form we consider for the function $u(x,t)$ is given as

$$\frac{\partial u}{\partial t} + 6u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0, \tag{2}$$

and in fact a particular 'solitary wave' solution of this has already been seen above in equation (1). Note that equation (2) is a partial differential equation (PDE) which is first order in time and third order in space. Here we employ a **method of lines** technique for solving this PDE, which means we will first discretise the spatial derivatives (with $N$ discretisation points, using **finite differences**), and then evolve the resulting system of $N$ ordinary differential equations (ODEs) with a time-stepping routine of our choice (we will use an inbuilt MATLAB solver). The PDE will be solved with periodic boundary conditions at $x = -a$ and $x = a$, and solved from time $t = 0$ to $t = t_{\text{final}}$.

To understand the code `kdvFD_periodic.m`, consider the following steps:

(a) Write down Taylor series expansions of a continuous and sufficiently differentiable function $f(x)$ for the points $f(x+h)$, $f(x+2h)$, $f(x-h)$, and $f(x-2h)$ up until (and including) terms $O(h^3)$. Then find approximations for $f'$ and $f'''$ by considering $f(x+h)-f(x-h)$ and similar combinations. Convince yourself that these approximations have an error of $O(h^2)$.

(b) Place a breakpoint on line 15 by clicking on the "–" next to the line number. Change the value of $N$ to use 10 discretisation points, and run the code. Now in the main MATLAB window look at the output for $D1$. Do you understand the relationship between this and your approximation for $f'$? Do similarly for $D3$ and $f'''$ by placing a breakpoint on line 21.

(c) Look again at $D1$ from your breakpoint on line 15, and then add another breakpoint on line 16 and re-evaluate $D1$. Can you see how the values now "wrap-around" and will effectively apply periodic boundary conditions? [See $D3$ with a breakpoint on line 24 similarly]. Finally for the construction of the differentiation matrices, place a breakpoint on line 27 and look at $D1$ and $D3$ to see how they have been scaled appropriately with the spacing between discretisation points.

(d) Consider equation (1). We would like to exploit the fact that we have an analytic solution for a particular case to check our code. Hence we wish to start our simulation with the same initial condition (i.e. the solution at $t = 0$). Verify that by substituting $t = 0$ into (1) that you obtain $u(x,0) = 6\ \text{sech}^2 x$, and see that this is implemented on line 27 of the MATLAB code.

(e) Remove all breakpoints, set $N$ back to $N = 300$, and run the code. You should at first see a "waterfall" plot of the evolution of the numerical solution. Press any key and another figure should display showing a 'video' of the simulation with a comparison between your numerical solution and the analytical one from equation (1).

(f) The movie ended at $t = t_{\text{final}} = 1$, with a small discrepancy between numerical and analytical solutions. Compare this to a run with $N = 200$ and $N = 500$ to see the importance of checking numerical convergence. [Note here that we are checking that the spatial discretisation has converged. MATLAB is automatically making sure of our time-stepping as we are using `ode15s` to solve the resulting ODE system, which adaptively varies the time steps to stay within a small tolerance].

(g) Try changing $t_{\text{final}}$ to allow for a longer simulation. For instance, try $t_{\text{final}} = 5$ to see the periodic boundary conditions in action, and also the physics of the collision of solitary waves. Note that the analytic solution is for an infinite domain, so disappears from view on the right hand side after $x = 20$.

(h) Try using `help` followed by a space, and then any MATLAB command you don't recognise in the code to read about what it does. In particular, see that `ode15s` would give a solution at timepoints based on its own adaptive timestepping, and hence to evaluate the solution at regular intervals we make use of `deval`.

(i) Adapt the code to solve $\partial_t u + \beta u \partial_x u + \partial_x^3 u = 0$ and solve with $N = 400$, $a = 5$, $t_{\text{final}} = 2 \times 10^{-2}$, and $\beta = 1$. Use initial condition $1200\ \text{sech}^2(10x + 20) + 300\ \text{sech}^2(5x + 5)$, remove the analytic solution, and set the axes in the 'video' to use `axis([-a a -500 1500])`.