AA 2021/2022 Computational Methods
Lesson 7 - Partial Differential Equations

Almaz Khabibrakhmanov*, Mario Galante†, Alexandre Tkatchenko

Theoretical Chemical Physics group, FSTM,
Campus Limpertsberg, University of Luxembourg

Numerical Solution of the 1D Wave Equation

Today, we are going to solve the 1D wave equation that represents the motion of a string vibrating. The general formulation of our example problem is the following:

$$u_{tt} = c^2 u_{xx},\ 0 \le x \le a,\ 0 \le t \le T \tag{1}$$

$$\begin{cases} u(0,x) = f(x) \\ u_t(0,x) = g(x) \end{cases}, \quad \begin{cases} u(t,0) = \phi(t) \\ u(t,a) = \psi(t) \end{cases} \tag{2}$$

We are solving for the function $u(t,x)$ describing the vertical shift of the point of the string with the coordinate x at the moment of time t. We will use the explicit Euler method when the second derivatives both in space and in time are approximated by the central differences:

$$u_{tt} = \frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\Delta t^2},\ u_{xx} = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{\Delta x^2} \tag{3}$$

The lower index j refers to the space axis, and the upper index k refers to the time axis.

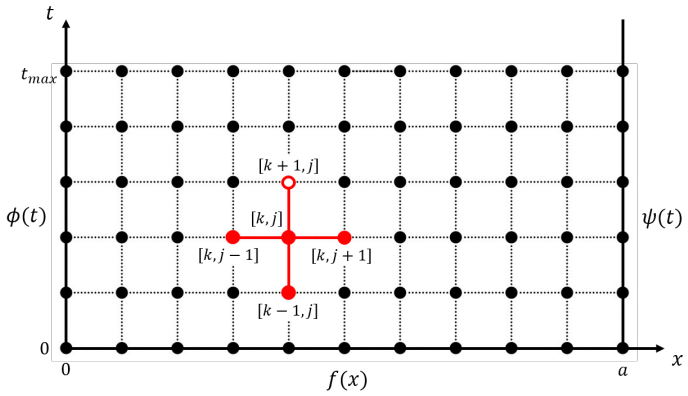

Figure 1: The numerical grid with the stencil of the computational scheme shown in red.

Solving then Eq.(1) for u_j^{k+1} at the timestep $k+1$, we obtain:

$$u_j^{k+1} = -u_j^{k-1} + 2u_j^k(1 - s^2) + s^2(u_{j+1}^k + u_{j-1}^k)\ , \tag{4}$$

where $s = \frac{c\Delta t}{\Delta x}$ is the so-called Courant number, key dimensionless parameter of the problem determining the stability of the numerical scheme. To account for the Neumann initial condition (on the derivative u_t), the first timestep requires the special treatment. As it was explained during the lecture, if $g(x) = 0$, for the first timestep we will have:

$$u_j^1 = u_j^0(1 - s^2) + \frac{1}{2}s^2(u_{j+1}^0 + u_{j-1}^0)\ , \tag{5}$$

*almaz.khabibrakhmanov@uni.lu
†mario.galante@uni.lu

Part 1 - Setting Up The Scene

1. Write the function `InitCond` implementing the following initial conditions:

$$f(x) = \begin{cases} 1.25\frac{x}{a}, & x \leq 0.8a \\ 5(1 - \frac{x}{a}), & x > 0.8a \end{cases} \tag{6}$$

   Physically this corresponds to the situation when the string is "plucked" 1 mm at $x = 0.8a$.

2. Write two functions, `LeftEnd` and `RightEnd`, implementing the following boundary conditions on the left and right ends of the string, respectively:

$$\phi(t) = 0, \qquad \psi(t) = 0 \tag{7}$$

   Physically this means that the string is fixed at its ends.

3. Initialize the variables which define the parameters of the problem: $c = 0.5$, $a = 1$, $t_{max} = 20$, $\Delta t = 0.02$, $\Delta x = 0.1$.

4. Set up the variables, `x` and `t`, which store the grid points in space and time, respectively. The steps of the grid are defined by Δx and Δt.

5. Initialize the two-dimensional zero `numpy` array, `u_num`, which will store the numerical solution $u_{num}(t, x)$ at the grid points so that `u_num[k,j]` $\equiv u_{num}(t_k, x_j)$.

6. Download the script `wave.py` from the Moodle. It contains two functions, `Analytical` and `Animator`. The first one provides the analytical solution, and the second one is needed for the visualisation of the results. Copy them into your code. Read the code for `Analytical` together with comments and try to understand it. Ask a question if something is not clear for you.

7. Calculate the analytical solution by calling the function `Analytical` and save the solution to the 2D array `u_an`. Plot the solution by calling `Animator` to check that everything works correctly.

Part 2 - The Solution

Now you have to write the function `WaveSolver` which will solve the 1D wave equation numerically. Below is the pseudocode that should help you. Read through it and then try to think about the arguments that your function may require to be provided. Only then start coding.

Algorithm 1 1D wave equation solver

1: Assign the values to u_j^0 for all $j = 1, \ldots, n_x$ using the initial conditions.
2: Assign the values to u_0^k and $u_{n_x}^k$ for all $k = 1, \ldots, n_t$ using the boundary conditions.
3: Make the first timestep: assign values to all u_j^k with $k = 1$ which are *unknown*. Use Eq. (5) for that.
4: Evaluate all *unknown* u_j^k for the remaining timesteps using Eq. (4) (write a double loop).
5: Return the calculated 2D array with the u_j^k values.

Part 3 - Analysis

1. Uncomment the lines in `Animator` which are responsible for adding the second line to the graph. Add `u_num` as an additional argument to the `Animator`.

2. Call your function `WaveSolver` and save the results to the variable `u_num`.

3. Call the function `Animator`. Wait for the animation to be created and saved (this may take some time). Analyze the result.

4. Now try to play with the different values of space- and timesteps. We suggest you to try some combinations of the following values: $\Delta t = 0.2, 0.1, 0.05, 0.02$; $\Delta x = 0.1, 0.05, 0.02, 0.01$. For every combination, don't forget to rename the file with the animation, e. g. `solution_dt=0.1_dx=0.1.gif`.

5. Analyze the obtained results. Recall what was discussed regarding the stability of the used numerical scheme during the lecture. Estimate the Courant number $s = \frac{c\Delta t}{\Delta x}$ for all the considered cases and make your conclusion.