# Numerical Methods: Lab III
Simulating Partial Differential Equations (I)

Guillermo Jiménez Pérez
Universitat Pompeu Fabra

2019-2020

## Introduction

In this third session, the simulation of systems described with partial differential equations (PDEs) is explored. Particularly, the Bateman-Burger's equation, the simplest PDE employed for simulating the movement of a wave that attempts to take into account non-linear phenomena such as fluid turbulence. The one-dimensional formulation of the Bateman-Burger's equation is:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{1}$$

For exploring this equation, we will simulate a tank full of water, checking how the fluid level evens out in the different points of the simulation. Numerically, the **explicit formulation** of the system will be solved through the usage of Euler and Runge-Kutta 2 methods. The main difference with the previous lab sessions is that the system contains spatial information as well as temporal; that is, the height of the water in the tank will be dependant on the time and on the position within the tank.

For illustration purposes, the system is composed of a "one-dimensional" container, full of the first liquid, lying in the $x$ axis. The depth of the container is considered to be small enough to be negligible, thus holding the 1D formulation. A representation of the container, viewed from above, could be:

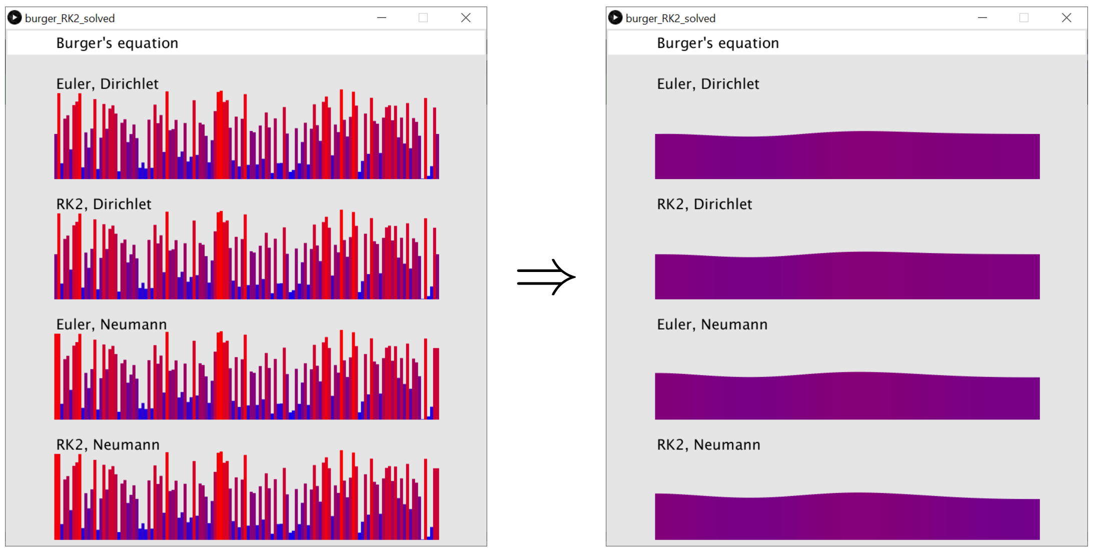

Figure 1: Initial randomized condition (left). System at some point in the simulation (right).

The main difference with respect to the previous lab session is the existence of spatial derivatives, which are discretized and take shape as different values for the quantity of interest (in this case, the water level) at different spatial locations. In this lab session, this will be represented by the different vertical bars in the left image, where each bar encodes a different height of the water at a specific location.

# 1 Theoretical Work

## Background Theory Concepts

**TW1 [5%]** What is the state composed of? Which variables should be stored?

**TW2** [5%] Obtain the spatial discretization expressions for $\frac{\partial u}{\partial x}$ and $\frac{\partial^2 u}{\partial x^2}$. These expressions are discretized using a Taylor series of the appropriate order.

**TW3**[$2 \cdot 5\%$] In accordance with the previous question, obtain the expressions according to which the system is updated using Euler and Runge-Kutta 2.

## Processing

**TW4[5%]** The system has spatial information, encoded in discretized sections of the water tank. Thus, the state needs to store the values for different positions of the water tank for each time step. How can this spatial information codified in the state? That is, which data structure could be used to store these values?

**TW5** [5%] The obtained expression contains the terms $u^k_{j+1}$ and $u^k_{j-1}$. Can we compute these values at positions $j = 0$ and $j = N - 1$? If not, how can we update the values of these positions?

**TW6[5%]** In this session, Dirichlet and Neumann boundary conditions will be applied. These boundary conditions apply an update rule at the boundary elements of an initial value problem and substitutes its value according to a given expression. In this case, the Dirichlet boundary conditions take the shape of:

$$u|_{\partial\Omega} = c,$$

where $c$ is a constant and $|_{\partial\Omega}$ is read as "evaluated at $\partial\Omega$". The Neumann boundary conditions, on their behalf, are defined as:

$$\left.\frac{\partial u}{\partial t}\right|_{\partial\Omega} = 0,$$

where $\partial\Omega$ is the elements that compose the boundary. According to this, answer the following. 1) Which are the elements that compose the boundary for this problem? 2) How can we translate these expressions into code?

**TW7** [5%] **Given your responses for TW3**, write the equations for Euler and Runge-Kutta 2 that must be coded into processing and rearrange/order them so that the information for computing a specific equation is obtained from the previous equations. That is:

(1) Write the equations for updating the temporal evolution for each of the methods, according to TW3.

(2) List the equations and check any dependencies between the equations. If necessary, use a few values to check these dependencies.

(3) Rearrange the equations for respecting the dependencies.

# 2 Practical Work

Constant values are given in the code. Given the problem statement described in the introduction, code the following:

**PW1 [5%]** Define the states for Euler and Runge-Kutta, calling them **StateEDirichlet** and **StateRK2Dirichlet**, respectively. Define also the states **StateENeumann** and **StateRK2Neumann** for question PW5. The array size has 128 elements (variable ArraySize in the provided code). Fill all the elements of the state with zeros.

**PW2 [5%]** Codify the computation of the approximation of $u_x$ and $u_{xx}$ in the functions "*approximate_ux*" and "*approximate_uxx*", respectively. These functions must take an array and return another array, where each position is updated according to the rules written in TW2.

**PW3 [10%]** Fill the *SetInitialState* function. For doing so:

(a) Set the current simulation time to zero.

(b) Fill the rest of the state elements with random values. Beware! All states (**StateEDirichlet**, **StateRK2Dirichlet**, **StateENeumann** and **StateRK2Neumann**) must have exactly the same values at every position after the function call.

**PW4 [10%]** Fill the *EnforceBoundaryConditions* function. This function takes an array, the index (specific element of the state, e.g. the height of water in the tank) whose values will be modified, and a strategy, which can only be either "Dirichlet" or "Neumann", and modifies the boundary positions of given array according to the given strategy. The formulation must follow your response in question TW6. Use a constant value of 0.5 for the Dirichlet implementation.

**PW5 [2·15%]** For each of the methods (Euler and RK2), fill the corresponding *TimeStep* function, according to the expressions defined in TW7. This function takes an array over which to apply the specific method and a boundary condition strategy (a string) and performs the update according to the rule (Euler or RK2) over the given array.

**PW6 [0%]** For the visualization, fill the *SimY* line in the *DrawState* function with the name of the index you used to retrieve the system's concentration. Also, replace the 0 in the line with the *for* loop with the array size.

## Submission

You have to deliver all the code for both preliminary and practical work as pde files. Your code should be commented and clear, easy to understand and easy to follow. You have to submit as well a written report with the answers of the questions, any observations relevant to the development of the lab, and conclusions of your work. All the files should be submitted in a zip file though the Aula Global.

**IMPORTANT NOTE: A blind revision scheme will followed, so USE ONLY YOUR NIAS at the code and the document. DO NOT PUT YOUR NAMES ANYWHERE**