Version: December 3, 2017

Practical Course Reader for Control of distributed systems

made at the

University of California at Berkeley

BY **Arnaud de LA FORTELLE**

visiting professor from MINES ParisTech

---

for the course CE-C291F — EE-C291 — ME-C236

With the help of:

Bradley Cage	student
Hongbei Chen	student
Yue Hu	student
Carlin Liao	student
Xin Peng	student
Robert Ruigrok	student
Lin Yang	student
Qingan Zhao	student
Ruitong Zhu	student

Contents

Chapter 1

Introduction

1.1 Objectives

This practical course reader aims at giving good advice to students realizing a project. Starting with the theoretical knowledge contained in the reader of Prof. Alexandre Bayen, students have to realize a project based on a distributed parameter system that they should:

1. describe
2. model
3. explain

The first part is a descriptive (i.e. qualitative) description of the system to be studied, its interest and the goal of the study. The second part is an elaboration, under simplifications, of a quantitative model (meaning here some kind of Partial Differential Equation and some control or optimization). The third part is an analysis of this model, either by solving the equations, by simulation or by a combination of both, using the tools learned during the course. It should lead to a better understanding of how the system works and how it can be optimized or controlled to fulfill the goals set in the description.

Therefore this practical course reader shows examples of modeling and various analysis methods that can support the creativity of the students realizing a project.

1.2 System description

A system is, according to Cambridge dictionary "a set of connected things or devices that operate together". There is no recipe to describe a system.

However, it is often useful, instead of having an *analytical* description (i.e. tear the system into pieces), to have a *synthetic* one (i.e. seeking for a global view). Without any generalization, again, one can look after the following 4 views of a system:

1. objective
2. environment
3. actions
4. transformations

One can very arguably contest these words, but remember they are only guidelines. In most systems, since they are man-made, there is a goal. This *objective* is important to describe since it contains the meaning: what is it for? Note this is quite often the opposite of the analytical view: how does it work? However, in our case, this goal contains the information about the *objective function* for optimization or for control. The *environment* dictates what the system has to do: a system is never isolated and its goal is often closely related to its environment. At least, the *actions* a system perform are partially internal and partially external, so that taking into account the environment allow understanding of disturbances but also the objective. Since a system is usually complex, it may evolve (i.e. experience *transformations*) according to actions and in line with the objective. One can also think of transformation as internal states of a finite automaton.

In any case, a description is intended at telling the whole story to other people and this is why it is highly recommended to use only words, no formula, but drawings can well replace a thousand words.

1.3 System modeling

While a system description is intended to be purely qualitative, modeling aims at estimating some quantities. Very clearly, the choice of the quantities is decisive. Important quantities can be derived from the system description, such as the objective function (what to maximize or minimize?), the transformations (internal states) or environment variables.

There are many ways to measure quantities, e.g. numbers of vehicles in a microscopic view may be equivalent to measuring densities of vehicle at a macroscopic scale. Unfortunately there is no algorithm to produce significant model and this is a knowledge that needs practice. However, in our present case, namely partial differential equations modeling, there is very often a

kind of scaling that is necessary in order to produce continuous-space and continuous time variables. Since this reader aims at being practical, the reader is referred to Chapter 2 for modeling examples.

One can (too?) quickly summarize some usual quantities of interest:

- macroscopic quantities (temperature, density, deformation...);
- variables (state space, time...) and parameters including the control;
- optimization criteria;
- invariant quantities (mass, energy, momentum...) and constraints;
- initial and boundary conditions.

1.4 System explanation

What is called here explanation has been also called "solution" during the course. It consists mainly in studying the model under various views and to produce new knowledge. This usually implies simulations, analytical analysis, optimization and visualization; all previous methods can be compared or combined in order to give more insight of the model.

1.4.1 Analysis

Analysis is important for the concepts it carries. We have learned a few methods to exactly solve or to transform the PDEs:

1. Dimensional analysis allow to build interesting quantities that should remain constant;
2. Product-form solutions leads to decomposition of solutions (often in the type of Fourrier transform) with knowledge of the modes and their frequencies (spatial and temporal).
3. the method of characteristics is a powerful methods to build solutions, even discontinuous like shockwaves in LWR equations. The characteristic curves are interesting features.

In the present course, students should perform some initial analytical analysis, even if it does not lead to a complete solution, in order to exhibits interesting quantities as mentioned above. These quantities should be used as guidelines as far as possible for further study.

1.4.2 Simulation

Simulation is interesting for all the data it brings. However there are many ways to simulate a system. At least we can mention:

- Microscopic simulation — often agent-based — where a lot of microscopic entities evolves with their dynamics; This method is relevant when the PDE is a macroscopic view of the system, i.e. when it is derived from a scaling;
- Finite differences schemes, that is often a direct implementation of the dynamics underlying the PDE (differences apply on a discretization of space and time);
- Finite elements is usually associated to computation of eigenvalues and eigenvectors, i.e. to product-form solutions and the computation of modes; in this case, discretization is made through a projection of the functions onto a finite (functional) basis.

In this course, students have to produce simulations. The goal is to exploit the data to visualize the system evolution (see also Section 1.4.4) and get a better understanding. One of the main problem of simulation is to have a clear idea of the quantities to be displayed, be it statistics, metrics or functions. This is why a good prior analysis (see also Section 1.4.1) is a good idea. A must in simulation is to compare quantities computed from the output of the simulation with similar quantities computed from theoretical analysis: e.g. number of vehicles per km in micro-simulation with densities in LWR equations.

1.4.3 Optimization or control

The goal of this course is to learn how a system can be controlled or optimized. This requires to have a good description of the system in order to have clear ideas of what is to be optimized or controlled. Usually, the design of the control is free and this is where students can be creative.

There is a very common distinction between 2 close concepts: planning and control. In most real systems, the planning phase is responsible for building suitable a path, according to the dynamics of the system and to the environment; then, a control loop ensures the systems follows this path by using actuators (the controls) and taking care of disturbances (in sensors and in actuators). Planning is open-loop, usually with a simple model (deterministic...) but with a complex cost function that expresses the desirable

behavior. Control is closed-loop, designed to be robust, taking into account noises and disturbances, with a simple cost function: follow the plan.

Techniques are very diverse and really depend on the system and the kind of control. Examples are given in Chapter 4.

1.4.4 Visualization

The visualization aspect of a study may be the most important yet the most challenging. Indeed, a drawing is extremely instructive, provided it is well explained. But the beauty can be misleading and the students have to take great care of the quantities they illustrate: What should be seen in the drawing? What does it tell us? Does it compare well with existing knowledge or not?

Therefore it is advised to start quickly (very early with simulation data) to try to see something and to continuously improve the visualization aspect. It is important for explaining the results as well as for enhancing the students' own understanding (and debugging the codes, very often).

Chapter 2

Modeling

2.1 Objectives

The modeling of a system starts with a good description (see Section 1.2). Very often, the system can is made of numerous elements (e.g. cars, particles...) or can be decomposed into small chunks (string, heated body...). In this course we want to obtain PDEs: this means that the dynamics is local. Small elements influence only their neighbors through quantities that are very important: tension and curvature for a string, density and flow for traffic... In order to obtain PDEs, it is often necessary to assume such quantities are slowly varying and that one can *scale* the system. Moreover, one should keep only a very limited number of quantities and simplifying assumptions are key to that objective (neglect gravity, torsion, viscosity...). Finally, an equation (even after scaling) is always obtain as limit of another equation: it is important to explain which principles (physical or other) lead to these equations, be it a conservation equation or dynamical equations.

Now, a model is not complete with only a PDE: initial and boundary conditions must be explicit, the control parameter (or the space of choice) must be given as well as the optimization criteria. The following examples illustrate this method.

2.2 Derivation of partial differential equations

This section shows how one can describe some simple systems and derive from physical principles the PDEs modeling their behavior.

2.2.1 2D wave equation

Prepared by subgroup 1 (Bradley Cage and Lin Yang)

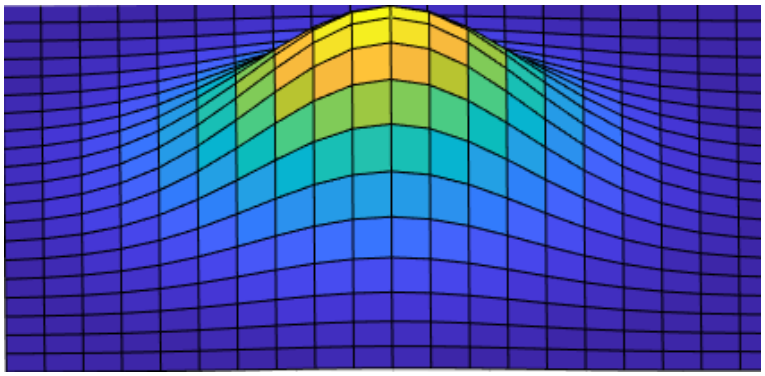


Fig. 2.1: The membrane system

Description Our system is comprised of a flexible membrane stretched to some shape, with all of its edges fixed in place. The desired goal is to understand the vertical position of the various points on the membrane over time. The membrane in this system has vertical deflections which are small compared to its overall size, and deflections happen only in the vertical direction.

This 2D system is a continuation of the 1D wave equation, and is a natural precursor to the 3D wave case.

Model Assumptions:

- Membrane has uniform planar density ρ
- The tension per unit length, F_t, caused by stretching the membrane is the same at all points and in all directions and does not change during the motion
- Vertical position is given by some function $u(x, y, t)$

We begin from basic principles.

$$\Sigma F = m\vec{a}$$

Taking some small section of the membrane $\mathrm{d}x$ by $\mathrm{d}y$, we can replace mass and acceleration and since we know density and that $\vec{a}$ is the second derivative of position with respect to time, thus enabling us to rewrite the equation.

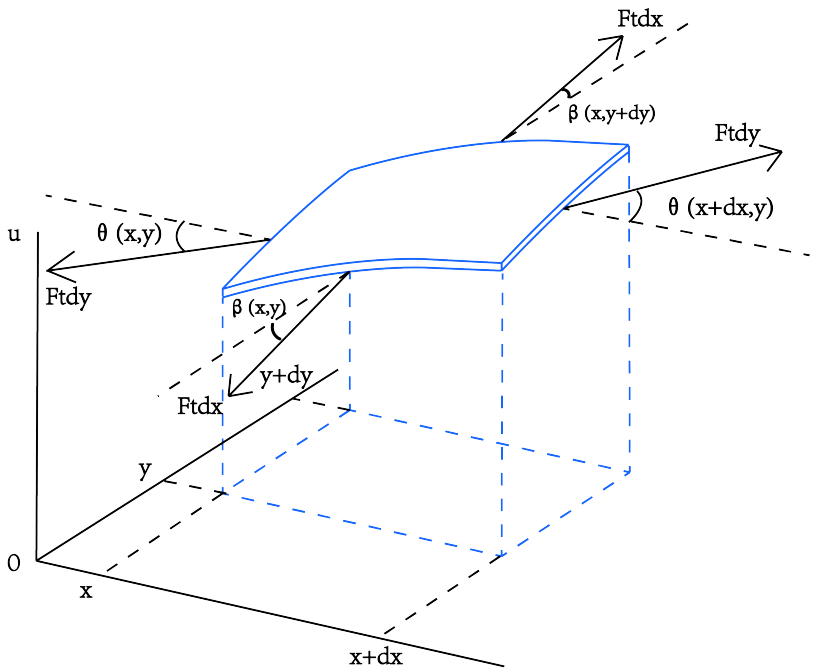


Fig. 2.2: The force analysis of a small section of the membrane system

$$\Sigma F = \rho \mathrm{d}x\mathrm{d}y\frac{\partial^2 u}{\partial t^2} \tag{1}$$

Performing a force balance on the section of membrane in the x and y directions gives us tensions at each on each side, then resolved to their vertical components. Remember that since tension is constant per unit length, we must multiply the force acting on each side by the length of that side. Thus, the force acting on this balance lets us rewrite ΣF (that is we only consider vertical forces, forces acting in the $x-u$ and $y-u$ planes):

$$\Sigma F = F_x + F_y$$

$$F_x = F_t \mathrm{d}y\Big[\sin\big(\theta(x+\mathrm{d}x,y,t)\big) - \sin\big((\theta(x,y,t)\big)\Big]$$

$$F_y = F_t \mathrm{d}x\Big[\sin\big(\beta(x,y+\mathrm{d}y,t)\big) - \sin\big((\beta(x,y,t)\big)\Big]$$

We can confidently use the small angle approximation sin in the x direction

$$\sin(\theta) \approx \tan(\theta) = \frac{\partial u}{\partial x} = u_x$$

and likewise in the y direction

$$\sin(\beta) \approx \tan(\beta) = \frac{\partial u}{\partial y} = u_y$$

to get our equations into the form

$$F_x = F_t \mathrm{d}y \Big[u_x(x + \mathrm{d}x, y, t) - u_x(x, y, t) \Big]$$

$$F_y = F_t \mathrm{d}x \Big[u_y(x, y + \mathrm{d}y, t) - u_y(x, y, t) \Big]$$

From there we can sum these forces and plug them back in to equation 1

$$\rho \mathrm{d}x\mathrm{d}y \frac{\partial^2 u}{\partial t^2} = F_t \Big[dy \Big[u_x(x+\mathrm{d}x, y, t) - u_x(x, y, t) \Big] + dx \Big[u_y(x, y+\mathrm{d}y, t) - u_y(x, y, t) \Big] \Big]$$

We then divide by $\mathrm{d}x$ and $\mathrm{d}y$ and take the limit as $\mathrm{d}x, \mathrm{d}y \to 0$:

$$\rho \frac{\partial^2 u}{\partial t^2} = \lim_{\mathrm{d}x, \mathrm{d}y \to 0} F_t \left[\frac{u_x(x + \mathrm{d}x, y, t) - u_x(x, y, t)}{\mathrm{d}x} + \frac{u_y(x, y + \mathrm{d}y, t) - u_y(x, y, t)}{\mathrm{d}y} \right]$$

We recognize that we now have derivatives in the form of difference quotients, and can take the partial derivative of each one (since u is a function of multiple variables)

$$\rho \frac{\partial^2 u}{\partial t^2} = F_t \left[\frac{\partial}{\partial x} u_x + \frac{\partial}{\partial y} u_y \right] = F_t \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] \tag{2}$$

Dividing over the uniform tension, we reach our final form.

$$\frac{\rho}{F_t} \frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{3}$$

We can adhere to standard conventions and write our final 2D wave equation as

$$a^2 \frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad a = \sqrt{\frac{\rho}{F_t}} \tag{4}$$

Equation 4 is also commonly written using the Laplace operator:

$$a^2 \frac{\partial^2 u}{\partial t^2} = \nabla^2 u \tag{5}$$

Initial and boundary conditions The initial state (i.e. at $t = 0$) of the system is given by two functions:

- the vertical position all over the membrane $u(x, y, 0) = u_0(x, y)$;.
- the membrane speed $\frac{\partial u(x,y,0)}{\partial t} = u_1(x, y)$. When — as usually assumed — the membrane is at rest, then $u_1 = 0$.

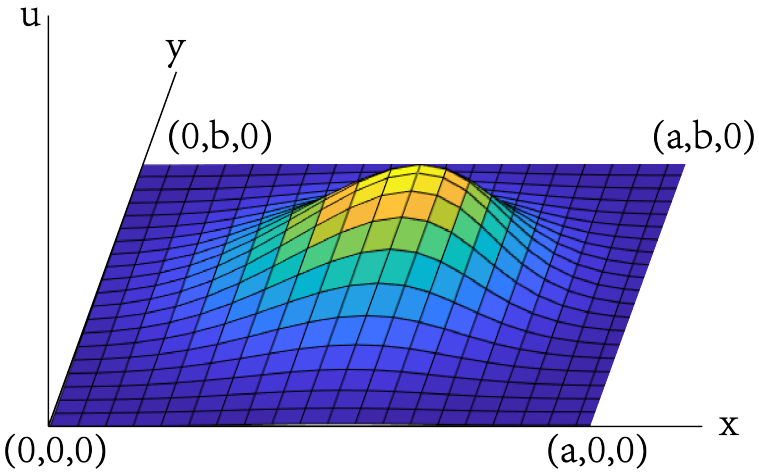


Fig. 2.3: The membrane system's boundaries

The membrane's area is ab (with the edge length being a,b respectively).

For membrane the boundary conditions can be of the two types (or a combination of):

- The vertical positions of the 4 edges of the membrane remain fixed all the time, and usually at 0, i.e., $u(x, 0, t) = 0$, $u(0, y, t) = 0$, $u(a, y, t) = 0$, $u(x, b, t) = 0$.
- Another type of boundary condition is the reflecting or no-flux boundary condition: this implies a derivative condition along the normal at the boundary and it is more technical to write (though not less meaningful in terms of physics).

Some hints for the control and the cost function of the 2D wave equation:

- Control: Maintain the highest vertical deflection of the membrane to be some constant height H
- The cost function then could be finding the smallest force exerted on the membrane to maintain the highest height H in the membrane system

2.2.2 2D heat diffusion

Prepared by subgroup 2 (Qingan Zhao and Ruitong Zhu)

Description The aim of this part is to describe and model a PDE that describes temperature dynamics in a two-dimensional body via heat conduction. Basically, heat conduction is the exchange of heat from regions of higher temperatures into regions with lower temperatures, which varies in the transfer rate for different materials.

Consider a thin flat body with a constant thickness h and uniform density ρ'. Assume that the faces of the thin body are in perfect insulation, which means there is no heat flow travel in the out-of-plane direction of the body. Hence, heat can only flow in the direction within the plane of the body, which turns into a two-dimensional problem. Then a two-dimensional coordinate system is established such that each point of the body can be described with a coordinate (x, y). Then the (2D-uniform) density of the body is $\rho = \rho' h$. Denote the temperature function of each point by T so that the temperature of the body at position (x, y) and time t are described as $T(x, y, t)$, as shown in Figure 2.4. The goal is to derive $T(x, y, t)$ when there is no internal heat source.

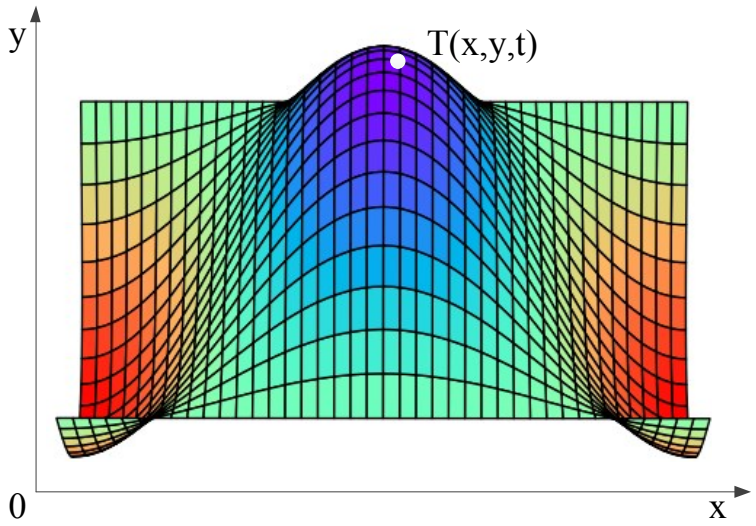


Fig. 2.4: System description in 2 dimensions

Model Consider a small rectangular element of the body with vertices (x, y), $(x + \mathrm{d}x, y)$, $(x, y + \mathrm{d}y)$, and $(x + \mathrm{d}x, y + \mathrm{d}y)$. The heat flows are shown

in Figure 2.5.

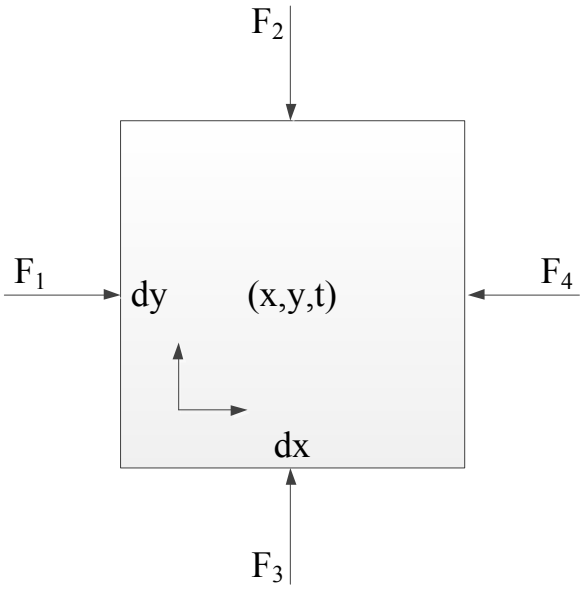


Fig. 2.5: Heat flows in a small rectangular element of the body

The heat amount Q (i.e the thermal energy) of the rectangular element at time t is:

$$Q(x, y, t) = CmT(x, y, t) \tag{6}$$

where C is called *heat capacity*, which is a supposed to be constant (assuming the material is uniform and temperature do not vary too much); $m = \rho A$ is the mass of the rectangular element where A its surface.

The rate of thermal energy change with respect to time is therefore:

$$\frac{\partial Q}{\partial t} = C\rho \mathrm{d}x\mathrm{d}y\frac{\partial T}{\partial t} \tag{7}$$

As shown in Figure 2.5, the incoming flow is $F_1 + F_2 + F_3 + F_4$. Denote the heat flux $\vec{q}$ in horizontal and vertical directions by q_x and q_y, then we have:

$$F_1 = q_x(x, y, t)\mathrm{d}y \tag{8}$$
$$F_2 = -q_y(x, y + \mathrm{d}y, t)\mathrm{d}x \tag{9}$$
$$F_3 = q_y(x, y, t)\mathrm{d}x \tag{10}$$
$$F_4 = -q_x(x + \mathrm{d}x, y, t)\mathrm{d}y \tag{11}$$

Now, we know that according to energy conservation, the thermal energy variation of any small element (as in Equation (7)) is equal to the total

incoming heat flow. By putting the partial flows as in Equations (8)-(11), this conservation principle yields:

$$C\rho \mathrm{d}x\mathrm{d}y\frac{\partial T}{\partial t} = \mathrm{d}y[q_x(x,y,t) - q_x(x+\mathrm{d}x,y,t)] + \mathrm{d}xh[q_y(x,y,t) - q_y(x,y+\mathrm{d}y,t)] \tag{12}$$

Now, another physical principle, *Fourier's Law*, states that the heat flow is (negatively) proportional to the gradient of temperature:

$$\vec{q} = -k\nabla T \tag{13}$$

where k is known as the thermal conductivity of the material (also considered as a constant). Then q_x and q_y are expressed as:

$$\begin{aligned} q_x &= -k\frac{\partial T}{\partial x} \\ q_y &= -k\frac{\partial T}{\partial y} \end{aligned} \tag{14}$$

Hence, Equation (12) can be written as:

$$\frac{\partial Q}{\partial t} = k\mathrm{d}yh\left(\frac{\partial T(x+\mathrm{d}x,y,t)}{\partial x} - \frac{\partial T(x,y,t)}{\partial x}\right) + k\mathrm{d}xh\left(\frac{\partial T(x,y+\mathrm{d}y,t)}{\partial y} - \frac{\partial T(x,y,t)}{\partial y}\right) \tag{15}$$

Combine Equation (7) and (15):

$$\frac{\partial T(x,y,t)}{\partial t} = \frac{k}{c\rho}\left(\frac{\partial^2 T(x,y,t)}{\partial x^2} + \frac{\partial^2 T(x,y,t)}{\partial y^2}\right) \tag{16}$$

Denote $k/c\rho$ by a^2, and the two-dimensional heat equation can be drawn:

$$\frac{\partial T}{\partial t} = a^2\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{17}$$

If we would like to solve the PDE in practice, the initial conditions and (or) boundary conditions need to be specified. For initial conditions, assume that in domain Ω:

$$T(x,y,0) = \phi(x,y) \tag{18}$$

As for boundary conditions, we should know either the value or the gradient of the function at some boundaries (i.e., fixed temperature or fixed flow).

For fixed temperature, for example, at the boundary $x = 0$, a boundary condition could be stated as follow:

$$T(0,y,t) = 0 \tag{19}$$

such a boundary condition means the temperature will always be 0 at the line $x = 0$ of the 2D plane.

The boundary condition could aslo be a fixed flow. For example, at the boundary $x = 0$:

$$\nabla(0, y, t) = 0 \tag{20}$$

such a boundary condition means that there is no heat transfer at the line $x = 0$ of the 2D plane.

When the conditions are given, it is the time to solve the PDE. Most simple PDEs can be solved using analytical methods, such as separation of variables and transform techniques. If the analytical methods do not work, numerical methods can be implemented to find the numerical approximations to the solutions of the PDE. A practical example with boundary conditions will be given in Section 2.3, where the analytical solution will be derived.

Once a system has been solved, we could enhance it by implementing the optimization. One thing we could do is to add some controls. For example, we would like the temperature of a plane less as stable as possible. A proper cost function could be a mathematical expression giving $\mathrm{d}T/\mathrm{d}t$ as a function of the external heat sourse. Tools such as Kalman filter can be implemented to do the optimization. A real world example is in aircraft temperature monitoring, for which temperature changes that above a threshold should be warned.

2.2.3 2D random walk

Prepared by subgroup 4 (Carlin Liao)

Description Consider a particle in a discrete two-dimensional grid at location $\vec{x} = [x_1, x_2]$ that moves in discrete time intervals. On each movement, the particle has an equal probability of moving one unit in any of the four cardinal directions $[x_1 + 1, x_2]$, $[x_1 - 1, x_2]$, $[x_1, x_2 + 1]$, and $[x_1, x_2 - 1]$. Our goal is to describe the probability of the point arriving at point $\vec{x}$ at time $t + 1$, $P[\vec{x}, t + 1]$ using Fick's Law.

Model Ultimately, our goal is to find a form of Fick's Law using $p(\vec{x}, t) = P[\vec{x}, t]$, of which the above description is setup for. Note that we can replace t with $t + 1$ or any other t without loss of generality due to the memoryless property of 2D random walk.

We begin by describing $P[\vec{x}, t+1]$ using our knowledge of the probabilities of where it may be at the prior time step t.

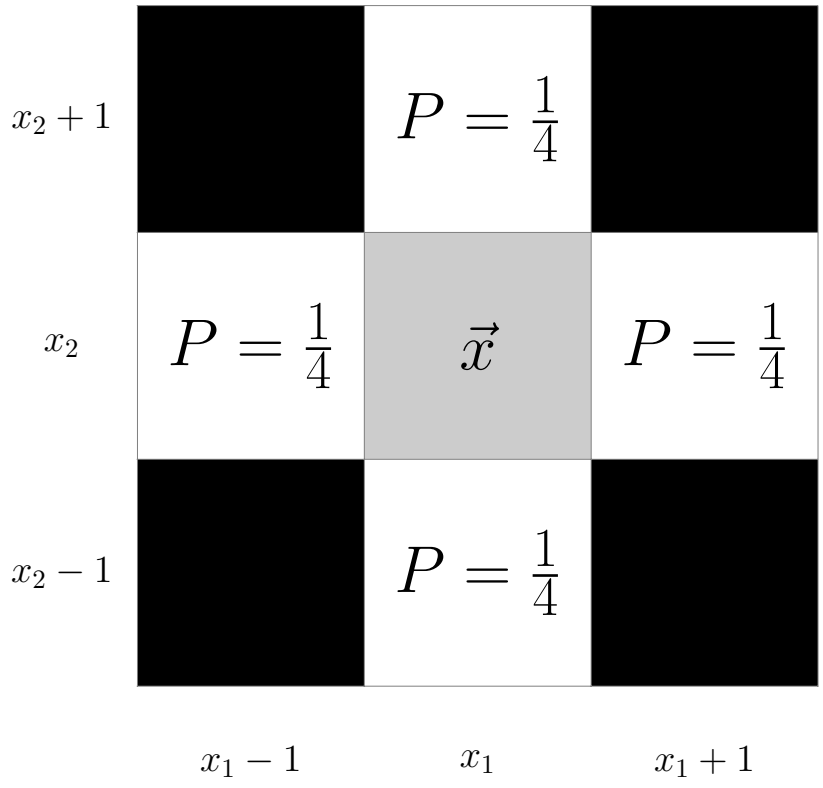


Fig. 2.6: Probability of next step in the 2D random walk

$$\begin{aligned} P\left[\vec{x}, t+1\right] = \quad & \frac{1}{4} P\left[\begin{bmatrix} x_1 + 1 \\ x_2 \end{bmatrix}, t\right] \\ & + \frac{1}{4} P\left[\begin{bmatrix} x_1 - 1 \\ x_2 \end{bmatrix}, t\right] \\ & + \frac{1}{4} P\left[\begin{bmatrix} x_1 \\ x_2 + 1 \end{bmatrix}, t\right] \\ & + \frac{1}{4} P\left[\begin{bmatrix} x_1 \\ x_2 - 1 \end{bmatrix}, t\right] \end{aligned}$$

In order to develop the parallels to Fick's law, we take the Taylor expansions of each of our probabilities with respect to t for $P\left[\vec{x}, t+1\right]$ and with respect to x for each of the probabilities we've used at time t.

First, we take the our target probability and find its first order Taylor

expansion with respect to time, and using our $p(\vec{x}, t)$ as described earlier.

$$\begin{aligned} P\left[\vec{x}, t+1\right] &\approx P\left[\vec{x}, t+1\right] + \nabla_t^T \left(P\left[\vec{x}, t+1\right]\right)(t+1-t) \\ &= p + \frac{\partial p}{\partial t} \\ &= p \end{aligned}$$

Note that we exploit the memoryless property of random walk to reduce $\frac{\partial p}{\partial t}$ to 0, as the probability that a particle performing truly random walk is in a certain location in space is not influenced by time.

Next, we take the second order Taylor expansion of our probabilities of movement in the x_1 direction, using similar substitutions as before. Here, $\mathcal{H}(f)$ describes the Hessian of the function f.

$$\begin{aligned} P\left[\begin{bmatrix} x_1 \pm 1 \\ x_2 \end{bmatrix}, t\right] &\approx p + \nabla_{\vec{x}}^T(p)\left(\begin{bmatrix} x_1 \pm 1 \\ x_2 \end{bmatrix} - \vec{x}\right) + \left(\begin{bmatrix} x_1 \pm 1 \\ x_2 \end{bmatrix} - \vec{x}\right)^T \mathcal{H}(p)\left(\begin{bmatrix} x_1 \pm 1 \\ x_2 \end{bmatrix} - \vec{x}\right) \\ &= p + \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix} \pm 1 \\ 0 \end{bmatrix} + \begin{bmatrix} \pm 1 \\ 0 \end{bmatrix}^T \begin{bmatrix} \frac{\partial^2 p}{\partial x_1^2} & \frac{\partial p}{\partial x_2 x_1} \\ \frac{\partial p}{\partial x_1 x_2} & \frac{\partial^2 p}{\partial x_2^2} \end{bmatrix} \begin{bmatrix} \pm 1 \\ 0 \end{bmatrix} \\ &= p \pm \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{\partial^2 p}{\partial x_1^2} \end{aligned}$$

Similarly, we take the second order Taylor expansion for probabilities in the x_2 direction.

$$\begin{aligned} P\left[\begin{bmatrix} x_1 \\ x_2 \pm 1 \end{bmatrix}, t\right] &\approx p + \nabla_{\vec{x}}^T(p)\left(\begin{bmatrix} x_1 \\ x_2 \pm 1 \end{bmatrix} - \vec{x}\right) + \left(\begin{bmatrix} x_1 \\ x_2 \pm 1 \end{bmatrix} - \vec{x}\right)^T \mathcal{H}(p)\left(\begin{bmatrix} x_1 \\ x_2 \pm 1 \end{bmatrix} - \vec{x}\right) \\ &= p \pm \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} \frac{\partial^2 p}{\partial x_1^2} & \frac{\partial p}{\partial x_2 x_1} \\ \frac{\partial p}{\partial x_1 x_2} & \frac{\partial^2 p}{\partial x_2^2} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= p \pm \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \frac{\partial^2 p}{\partial x_2^2} \end{aligned}$$

Now, we combine all of the above into our original expression

$$\begin{aligned} p = \quad & \frac{1}{4}\left(p + \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix}1\\0\end{bmatrix} + \frac{\partial^2 p}{\partial x_1^2}\right) \\ & + \frac{1}{4}\left(p - \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix}1\\0\end{bmatrix} + \frac{\partial^2 p}{\partial x_1^2}\right) \\ & + \frac{1}{4}\left(p + \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix}0\\1\end{bmatrix} + \frac{\partial^2 p}{\partial x_2^2}\right) \\ & + \frac{1}{4}\left(p - \left(\frac{\partial p}{\partial x}\right)^T \begin{bmatrix}0\\1\end{bmatrix} + \frac{\partial^2 p}{\partial x_2^2}\right) \end{aligned}$$

We note how the first order terms cancel out and proceed to further simplify. Take note of the use of the Laplacian operator ∇^2.

$$\begin{aligned} p &= p + \frac{1}{4}\left(\frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2} + \frac{\partial^2 p}{\partial x_2^2}\right) \\ 0 &= \frac{1}{2}\left(\frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2}\right) \\ \nabla^2 p &= \frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2} \\ \nabla^2 p &= 0 \end{aligned}$$

We note that this final result to be (as expected) the solution to Fick's second law in the special case of the diffusion (or random walk, in our case) being independent of time, also known as Laplace's equation. For a more general result (i.e. not assuming that our particle's position is independent of time), using these same methods we would arrive at the more general statement of Fick's second law

$$\frac{\partial p}{\partial t} = \nabla^2 p$$

To close, while this result was derived for the two-dimensional random walk case, the derivation for n-dimensions should be readily apparent from this case.

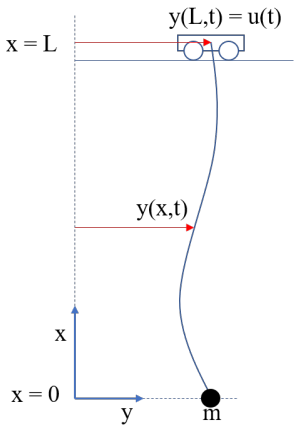
Adapted in part from http://engineering.dartmouth.edu/~d30345d/courses/engs43/chapter2.pdf

2.2.4 Heavy chain

Prepared by subgroup 4 (Rob Ruigrok)

Description This chapter will cover the derivation of the dynamics of a heavy chain with uniform density ρ, length L and a point mass on the end with mass m. The top of the chain will be suspended on a rolling trolley, where the lateral speed can be controlled. First, we will derive the dynamics, and later go into more detail on the "flatness" of the system and several control inputs.

Model An overview of the system is provided in figure 2.2.4. Let x be coordinate along the length of the chain, where the chain is attached to the trolley at $x = L$. The lateral displacement of the chain is denoted by $y(x, t)$. The problem can be made more complex by adding a point mass m to the end of the chain.

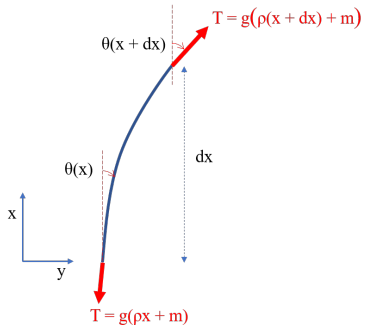


The derivation of the dynamics goes largely the same as the derivation of the vibrating spring. Here, however, the tension in the chain will be depending on x, and there will now be different boundary conditions linked to the trolley at the top and the attached mass at the bottom. Figure 2.2.4 illustrates the forces acting on a finite chain element of length dx, which is the basis in the derivation of the PDE.

We start with defining the tension in the chain as a function of x:

$$T = (m + x\rho)g \tag{21}$$

To analyze the dynamics, we are interested of the lateral component of this tension. When examining an infinitesimal chain element of length Δx, we are interested in the net lateral force. Instead of defining the force on both sides of element Δx, we here determining the difference by using a first order Taylor approximation, together with a small angle approximation:



$$sin(\theta)T \approx tan(\theta)T = \frac{\partial y}{\partial x}T \tag{22}$$

$$\Delta F \approx F_x \Delta x = \left[\frac{\partial y}{\partial x}T\right]_x \Delta x \tag{23}$$

The lateral motion of an infinitesimal chain element can now be described with Newtons second law of motion, $F = ma$, for which the net force is defined in equation 23. The mass of the chain element is $\rho\Delta x$, and the acceleration is $\frac{\partial^2 y}{\partial t^2}$

$$\begin{aligned} F &= ma \\ \left[\frac{\partial y}{\partial x}T\right]_x \Delta x &= \rho \Delta x \frac{\partial^2 y}{\partial t^2} \\ \left[\frac{\partial y}{\partial x}(m + x\rho)g\right]_x &= \rho \frac{\partial^2 y}{\partial t^2} \end{aligned} \tag{24}$$

When there is no mass suspended to end of the chain, the dynamics can be further written out:

$$\left[\frac{\partial y}{\partial x}x\rho g\right]_x = \rho \frac{\partial^2 y}{\partial t^2} \tag{25}$$

$$g\left[\frac{\partial y}{\partial x}x\right]_x = \frac{\partial^2 y}{\partial t^2} \tag{26}$$

$$g\left[x\frac{\partial^2 y}{\partial x^2} + \frac{\partial y}{\partial x}\right] = \frac{\partial^2 y}{\partial t^2} \tag{27}$$

with boundary condition: $y(L, t) = u(t)$.

Solving the PDE ***TODO: move part of it to control section***

It is difficult to directly find a solution for the partial differential equation in equation 525). For this particular problem, we assume that we know that we can rewrite to a Bessel function. Bessel functions are mainly used to describe wave propagation, and have the following format, where α denotes the order of the Bessel function:

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - \alpha^2)y = 0 \tag{28}$$

In order to make the formulation in equation 25 fit this a zero-order Bessel function, we have to do certain changes of variables and convert the problem to the Laplace domain. In the Laplace domain, we can find solutions for our type of Bessel function, after which we need to transform it back to the time domain and reverse the variable changes.

Step 1: substitution First step, do a substitution with $p = 2\sqrt{\frac{x}{g}}$. This will change the dependency from y(x,t) to y(p,t). Use the chain rule and find fill out the new values:

$$\begin{aligned}\frac{\partial p}{\partial x} &= \frac{1}{g}\left(\frac{x}{g}\right)^{-\frac{1}{2}} = \frac{1}{g\sqrt{\frac{x}{g}}} = \frac{2}{gp} \\ \frac{\partial y}{\partial x} &= \frac{\partial y}{\partial p}\frac{\partial p}{\partial x} + \frac{\partial y}{\partial t}\cancelto{0}{\frac{\partial t}{\partial x}} = \frac{\partial y}{\partial p}\frac{2}{gp}\end{aligned} \tag{29}$$

When we fill everything out in the left part of equation 25, we get the following expression:

$$\begin{aligned} g\frac{\partial}{\partial x}\left[\frac{\partial y}{\partial x}x\right] &= g\frac{\partial}{\partial x}\left[\frac{\partial y}{\partial p}\frac{2}{gp}\frac{1}{4}gp^2\right] = g\frac{\partial}{\partial x}\left[\frac{\partial y}{\partial p}\frac{p}{2}\right] \\ &= g\frac{\partial}{\partial p}\left[\frac{\partial y}{\partial p}\frac{p}{2}\right]\frac{\partial p}{\partial x} = g\left[\frac{\partial^2 y}{\partial p^2}\frac{p}{2} + \frac{1}{2}\frac{\partial y}{\partial p}\right]\frac{2}{gp} \\ &= \frac{\partial^2 y}{\partial p^2} + \frac{1}{p}\frac{\partial y}{\partial p} \end{aligned} \tag{30}$$

So the total equation can be written as:

$$g\big[\frac{\partial y}{\partial x}x\big]_x - \frac{\partial^2 y}{\partial t^2} = 0 \tag{31}$$

$$\frac{\partial^2 y}{\partial p^2} + \frac{1}{p}\frac{\partial y}{\partial p} - \frac{\partial^2 y}{\partial t^2} = 0 \tag{32}$$

$$p\frac{\partial^2 y}{\partial p^2} + \frac{\partial y}{\partial p} - p\frac{\partial^2 y}{\partial t^2} = 0 \tag{33}$$

Step 2: transform to Laplace domain When converting the from the time domain, we use the following notation:

$$y(x,t) \xrightarrow{\mathcal{L}} Y(x,s)$$

Further, to get the Laplace transform in the desired format, we require that the systems is initially *at rest*, which means that $y(x,0) = 0$ and $\dot{y}(x,0) = 0$.

$$\begin{aligned} p\frac{\partial^2 y(x,t)}{\partial p^2} + \frac{\partial(x,t)}{\partial p} - p\frac{\partial^2 y(x,t)}{\partial t^2} &= 0 \\ \xrightarrow{\mathcal{L}} p\frac{\partial^2 Y(x,s)}{\partial p^2} + \frac{\partial Y(x,s)}{\partial p} - ps^2 Y(x,s) &= 0 \end{aligned} \tag{34}$$

NEW CHANGE OF VARIABLES (did it myself on paper, still need to isert)
Bessel function!

2.3 Manipulation of partial differential equations

2.3.1 Weak formulation

Prepared by subgroup 3 (Xin Peng and Hongbei Chen)

Objective Weak formulations are important tools for the analysis of mathematical equations that permit the transfer of concepts of linear algebra to solve problems in other fields such as partial differential equations. In a weak formulation, an equation is no longer required to hold absolutely and has instead weak solutions only with respect to certain "test vectors" or "test functions". This is equivalent to formulating the problem to require a solution in the sense of a distribution.

Take Poisson's equation as a example. Our aim is to solve this Poisson's equation

$$-\Delta u = f \tag{1.1}$$

on a domain $\Omega \subset R^d$,$u = 0$ on its boundary.
Suppose u is continuously differentiable in continental space R^2, test it with differentiable functions v and integral, we get

$$\int_\Omega \left(\nabla^2 u\right) v \mathrm{d}x = \int_\Omega f v \mathrm{d}x \tag{1.2}$$

We can make the left side of this equation more symmetric by integration by parts using Green's identity and assuming that $v = 0$ on $\partial\Omega$

$$-\int_\Omega \left(\nabla^2 u\right) v \mathrm{d}x = -\int_{\partial\Omega} (\nabla u)\, v \mathrm{d}s + \int_\Omega \nabla u \nabla v \mathrm{d}x \tag{1.3}$$

$$-\int_\Omega \left(\nabla^2 u\right) v \mathrm{d}x = \int_\Omega \nabla u \nabla v \mathrm{d}x \tag{1.4}$$

The equation 1.4 is what is usually called the weak formulation of Poisson's equation. As we can see, weak formulations are partial differential equations testing with "test vectors" or "test functions" and then integral both side of equations. This transformation sacrifices the smoothness of solution. Since a large number of differential equations used to describe the phenomena in the real world do not have enough smooth solutions to solve such equations can only use weak formulation.

Demonstration A Dirichlet problem is the problem of finding a function which solves a specified partial differential equation in the interior of a given region that takes prescribed values on the boundary of the region. We take a PDE for example.

We consider f a continuous function on Ω of sumtable square and u the solution of the following partial derivative equation on Ω

$$-\Delta u + k^2 u = f \tag{2.1}$$

With the condition at the edge $u = 0$ on $\partial\Omega$. This can also be rewritten $u \subset V_0$. This condition at the edge is called the Dirichlet condition. We prove that there exists a unique solution to this PDE problem using the Lax-Milgram theorem.

Let $u \subset V_0$ be arbitrary. Multiply the two parts of the previous equation by v then sum to the domain Ω, since v and f are both summable square on this domain. Equation:

$$-\int_\Omega \Delta uv\mathrm{d}w + k^2 \int_\Omega uv\mathrm{d}w = \int_\Omega vf\mathrm{d}w \tag{2.2}$$

We can make the left side of this equation more symmetric by integration by parts using Green's identity:

$$-\int_\Omega v\Delta u\mathrm{d}w = -\int_{\partial\Omega} v\nabla u\mathrm{d}s + \int_\Omega (\nabla u\nabla v)\,\mathrm{d}w \tag{2.3}$$

In this formulation, $v = 0$ on $\partial\Omega$ ($v \subset V_0$).Thus $\int_{\partial\Omega} v\nabla u\mathrm{d}s = 0$.

$$\int_\Omega (\nabla u\nabla v)\,\mathrm{d}w + k^2 \int_\Omega uv\mathrm{d}w = \int_\Omega vf\mathrm{d}w \tag{2.4}$$

If u is twice differentiable, there is equivalence between this formulation and that of the initial problem given in the hypothesis section(Because we used to integral both side of equation). Then the solution of the weak formulation is the same as the initial solution. We can therefore solve the weak formulation instead of solving the initial problem.

2.3.2

2.3.3

2.3.4 Example of solving the 2D heat diffusion

Prepared by subgroup 2 (Qingan Zhao)

Description Now consider a retangular plane (length: a; width: b) with 2 boundaries that have no heat transfer (i.e., $T(x,0) = f(x)$; $T(x,b) = g(x)$). The other two boundaries have fixed temperature ($T(0,y) = T(a,y) = 0$). We would like to solve the PDE under the stationary state (i.e., $\mathrm{d}T/\mathrm{d}t = 0$). The diagram of the system is shown in Figure 2.7.

PDE and boundary conditions From Section 2.2.2 we have already known that the 2D heat equation is:

$$\frac{\partial T}{\partial t} = a^2\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \qquad where\ a = \sqrt{\frac{k}{c\rho}} \tag{35}$$

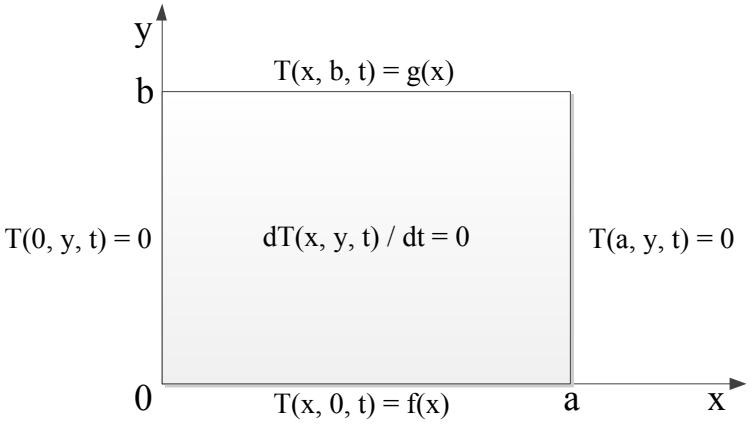


Fig. 2.7: System description of the example

The boundary conditions of this problem are as follows:

$$T(x, 0, t) = f(x) \tag{36}$$

$$T(x, b, t) = g(x) \tag{37}$$

$$T(0, y, t) = 0 \tag{38}$$

$$T(a, y, t) = 0 \tag{39}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = 0 \tag{40}$$

Solution Since the system is under the stationary state ($\mathrm{d}T/\mathrm{d}t = 0$), $T(x, y, t)$ can be expressed as $T(x, y)$, and the PDE can be written as:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{41}$$

We could sperate the variables by express $T(x, y)$ as $X(x)Y(y)$. Then the PDE can be written as:

$$X''(x)Y(y) + X(x)Y''(y) = 0 \tag{42}$$

Since $X(x)Y(y)$ is not identically zero, divide both sides by $X(x)Y(y)$ gives:

$$\frac{X"(x)}{X(x)} + \frac{Y"(y)}{Y(y)} = 0 \tag{43}$$

Since the two parts of the left side are independent, the following equations can be drawn (λ is a constant):

$$X"(x) + \lambda X(x) = 0 \tag{44}$$
$$Y"(y) - \lambda Y(y) = 0 \tag{45}$$

Moreover, the boundary conditions Equations (38)-(39) give:

$$X(0)Y(y) = 0 \tag{46}$$
$$X(a)Y(y) = 0 \tag{47}$$

Since $Y(y)$ is not identically zero, we can obtain the following:

$$X(0) = 0 \tag{48}$$
$$X(a) = 0 \tag{49}$$

Hence, the problem becomes solving two indenpendent ordinary differential equations. We know that only when $\lambda = \frac{n^2\pi^2}{a^2}$ $(n = 1, 2, 3, ...)$, the problem has the following solution:

$$X_n(x) = \sqrt{\frac{2}{a}} sin \frac{n\pi x}{a} \tag{50}$$

$$Y_n(y) = a_n e^{\frac{n\pi}{a}y} + b_n^{-\frac{n\pi}{a}y} \tag{51}$$

Hence, the solution can be expressed as:

$$T(x, y) = \sqrt{\frac{2}{a}} \sum_{n=1}^{\infty} (a_n e^{\frac{n\pi}{a}y} + b_n^{-\frac{n\pi}{a}y}) sin \frac{n\pi x}{a} \tag{52}$$

Given the other two boundary conditions Equations (36)-(37), a_n and b_n can be solved using the following 2 equations:

$$a_n + b_n = \sqrt{\frac{2}{a}} \int_0^a f(x) sin \frac{n\pi}{a} x dx \tag{53}$$

$$a_n e^{\frac{n\pi}{a}b} + b_n e^{-\frac{n\pi}{a}b} = \sqrt{\frac{2}{a}} \int_0^a g(x) sin \frac{n\pi}{a} x dx \tag{54}$$

Thus, the analytical solution of this problem is solved.

Chapter 3

Simulation

3.1 Objectives

The objective is to obtain numerical data (often the successive states of the system), to visualize these data, to build meaningful statistics and finally to link the results with theoretical considerations. Best would be to compare simulations with analytical results, but this may prove infeasible and then more analysis is required to compare theory and simulations.

3.2 Simulation examples

3.2.1 Forward Euler simulation of the 1D heat equation

Prepared by subgroup 1 (Lin Yang and Bradley Cage)

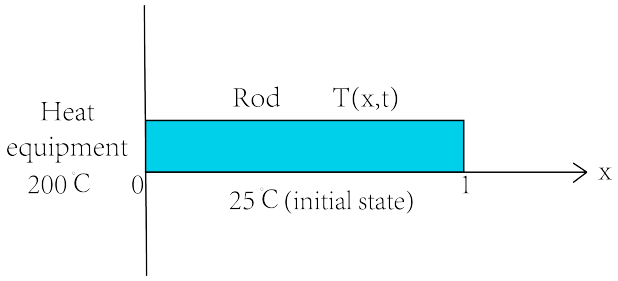


Fig. 3.1: The heated rod system, with the temperature at any point given by $T(x,t)$

Model presentation Our system is comprised of a rod fixed to a heated wall kept at constant temperature. The length of the rod is arbitrarily taken as 1m. The initial temperature of the rod is in equilibrium with the ambient temperature at 25°C and the temperature of the wall is 200°C. We represent temperature at a point on the rod at some time with the function $T(x,t)$. Note that the rod is thin with respect to its length, thus is modeled as a 1D system, that is, temperature is solely a function of rod position and time. The goal of this simulation is to show the variations in temperature of various points on the rod over time.

We take our initial state as:

- $T(0,t) = 200°\text{C}$
- $T(x,t) = 25°\text{C},\ x \neq 0$

The partial differential equation of the 1D heat propagation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1}$$

We explicitly state that $u(\cdot)$ is a function of x and t

$$\frac{\partial}{\partial t} T(x,t) = \alpha \frac{\partial^2}{\partial x^2} T(x,t) \tag{2}$$

We use a finite difference approximation to get compute the derivatives in space and time

$$\frac{T_i^{N+1} - T_i^N}{\Delta t} = \alpha \frac{T_{i+1}^N - 2T_i^N + T_{i-1}^N}{\Delta x^2} \tag{3}$$

Rearranging we reach the final form we need for our Forward Euler approximation. Note that the quantity $\alpha \frac{\Delta t}{\Delta x^2}$ is Fourier's number, with α being the thermal diffusivity of a material. In the following simulations, we arbitrarily choose Aluminium with $\alpha = 9.7e-5$.

$$T_i^{N+1} = T_i^N + \alpha \frac{\Delta t}{\Delta x^2} \left[T_{i+1}^N - 2T_i^N + T_{i-1}^N \right] \tag{4}$$

Implementation We provide an implementation in Matlab.

Code Listing 3.1: Forward Euler and Plotting in MATLAB

```
% Values arbitrarily chosen. It's a useful exercise to vary these
T_final = 300;
N_t = 2000;
```

```
X_final = 1;
N_x = 100;

% Calculate time and x steps based on sampling size and # of samples
T = linspace(0, T_final, N_t+1);
X = linspace(0, X_final, N_x+1);
dt = T(2) - T(1);  % Calculate delta t
dx = X(2) - X(1);  % Calculate delta x

alpha = 9.7e-5; % Thermal diffusivity of Aluminium in m^2/s
Fo = (alpha*dt)/(dx^2); % Fouriers number = diffusive transport rate/st

% Define your initial condition here. This could be some function IC(x)
%  however for simplicity's sake we take a rod with a uniform temperatu
%  and in contact with a hot plate at one end
wall_temp = 200;
init_temp = 25;

% Initialize the N state and the N-1 state
u_old = zeros(1, N_x+1);
u_old(:) = init_temp;
u_old(1) = wall_temp;
u_cur = u_old;
u_plot = u_old;

for t = 1:N_t
    for i = 2:N_x
        % Forward Euler solution to heat equation
        u_cur(i) = Fo*(u_old(i+1) - 2*u_old(i) + u_old(i-1)) + u_old(i);
    end
    u_cur(1) = wall_temp; % Set the left boundary to be our high of 200
    u_old(:) = u_cur; % We move to the next time step, reset N-1 state
    u_plot = [u_plot; u_cur];
end

[X_plot, T_plot] = meshgrid(X,T); % Create 2D meshgrid to create surface

surf(X_plot, T_plot, u_plot, 'EdgeColor', 'none') % Create surface plot
c = colorbar; % Attach colour bar and create scale
c.Label.String = 'Temperature [C]';
```

```
xlim([0 1]) % Add axis limits
xlabel('Distance along rod [m]'); % Add descriptive axis labels
ylabel('Time [s]');
zlabel('Temperature [C]')
```

Results We can create plots of the rods temperature as a function of time and position. This gives us insight into how the system evolves as we maintain our constant temperature.

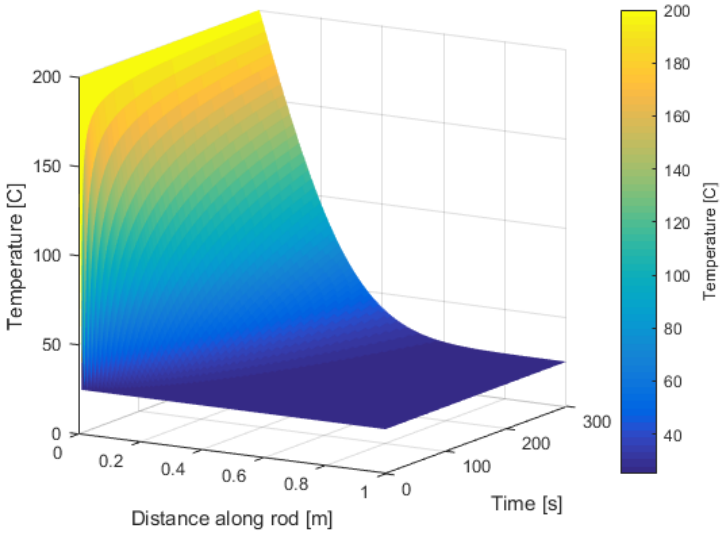


Fig. 3.2: Temperature variation for the whole rod over 300 seconds

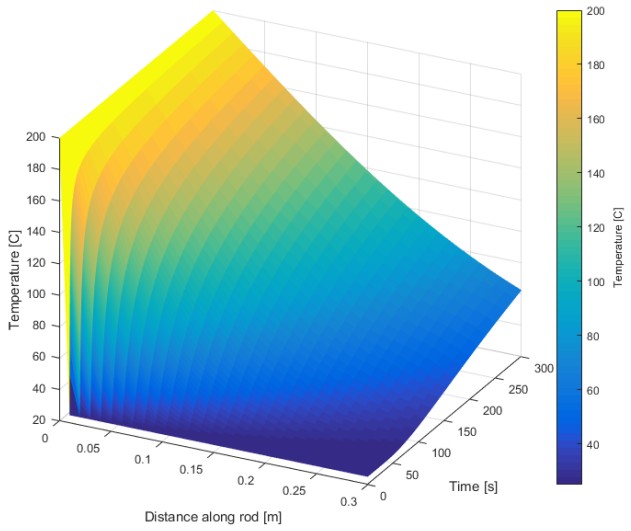


Fig. 3.3: Temperature variation for the rod segment with x varies from 0 to 0.3m over 300 seconds

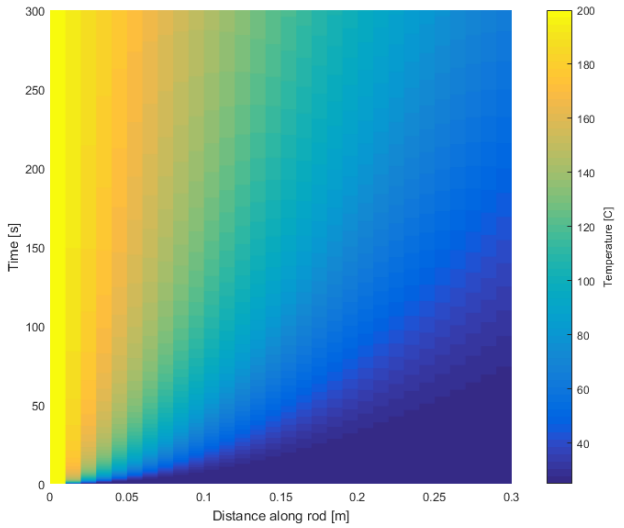


Fig. 3.4: Top view of the temperature variation for the whole rod over 300 seconds

Interpretation The results we obtain are consistent with our model and the physical characteristics. We can see that from the visualization the temperature varies in a logarithmic manner, which is consitent with the fact that the Forward Euler scheme approximates the exact mathematical solution to the 2nd order PDE, which takes the form of an exponential. The exact rate at which heat moves through the rod is dependent on the thermal conductivity of the material, which makes up part of the exponent (i.e. e^{-m}), thus governing the rate. We can see pathlines of the heat front evolve over time, especially when looking at the system from above. Given enough time, the rod will reach a steady state, which in this example would have the rod rest at a completely uniform temperature, since we have not incorporated any form of heat losses or sinks in the system.

Conclusion Here we have learned that in examining systems such as the heated rod, we can safely analyze it as a 1D system. We were able to discretize the partial differential equations and apply a forward Euler simluation to yield phyiscally relevant and meaningful simulations. Through the Matlab visualization, we gain a better understanding of how heat is moving through the rod and the system. From the resources and analysis provided, it is trivial to create other initial/boundary conditions and repeat the simulations to

garner a deeper insight into the heat behaviour. This is left as an exercise to the reader.

3.2.2 1D vibrating string simulation

Prepared by subgroup 2 (Ruitong Zhu and Qingan Zhao)

Model presentation This part is to simulate the vibrating string (i.e., 1D wave equation) and present its displacement with a time-varying image. The string held stationary at both ends and free to vibrate transversely subject only to the restoring forces due to tension in the string. *Figure*1 shows coordinates and definies symbols for the transverse vibrating string.

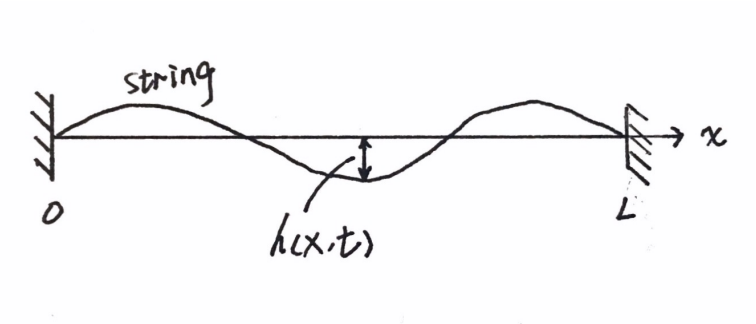


Fig. 3.5: Vibrating String

The partial differential equation (PDE) of this problem is given as follow:

$$\frac{\partial^2 h}{\partial t^2} = a^2 \left(\frac{\partial^2 h}{\partial x^2} \right) \tag{5}$$

where h is the wave function $h(x,t)$, representing the displacement of the string at position x and time t; a is the wave speed which equals to $\sqrt{E/\rho}$.

Implementation The simluation is based on Finite Difference Method (FDM). Using second-order central difference at time t_n and position x_i, we can get the recurrence equation as follow:

$$\frac{u_i^{N+1} - 2u_i^N + u_i^{N-1}}{\Delta t^2} = a^2 \frac{u_{i+1}^N - 2u_i^N + u_{i-1}^N}{\Delta x^2} \tag{6}$$

Assume the wave speed $a = 1$; the length of the string L is 2; the maxium time for this simulation is 4; stepsize Δx and Δt are both equal to 0.01. The

initial condition and the boundary condition are described as follows:

$$h(x,0) = sin(\pi x) \tag{7}$$

$$\left.\frac{\partial h}{\partial t}\right|_{(x,0)} = 0 \tag{8}$$

$$h(0,t) = h(2,t) = 0 \tag{9}$$

Here we offer the implementation in Python:

```
## parameter
a = 1  ## a coefficient of stiffness
L = 2  ## The string is constrained at x=0 and x=L.
T = 4  ## maxium time for this simulation.
dx = 0.01  ## time step
dt = 0.01  ## distance step
N = int(L/dx);
M = int(T/dt);
r = (a*dt/dx)**2  ## a parameter

## initial shape of the string
def initial(x):
tmp = math.sin(math.pi*x)
return tmp

## initial speed of the string
def speed(x):
tmp = 0
return tmp

## Define an array and a blank matrix for later use.
x = [0]
h = np.zeros((M+1, N+1))

## t=0, initial condition
for i in range(N):
x.append(x[i] + dx)  ## x axis
h[0,i+1] = initial(x[i+1])  ## displacement of the string

## t=dt, the first itertaion
for i in range(N-1):
h[1,i+1] = h[0,i+1] + r* (h[0,i] + h[0,i+2] - 2*h[0,i+1])/2
                                + dx*speed(x[i+1])
## displacement of the string

## t=n*dt where n>1
for j in range(1, M):
for i in range(N-1):
h[j+1,i+1] = (h[j,i+2]+h[j,i]-2*h[j,i+1])*r-h[j-1,i+1]+2*h[
                                j,i+1]
```

```
## displacement of the string

t = [0]
for j in range(M):
t.append(t[j] + dt)  ## t axis

## Plot the 3D figure.
fig = plt.figure()
ax = Axes3D(fig)
X, T= np.meshgrid(x,t)
ax.plot_surface(X, T, h, cmap='rainbow')
ax.set_xlabel('X')
ax.set_ylabel('T')
ax.set_zlabel('h')
plt.show()

## Plot the shape of string at different time
for i in range(4):
plt.xlabel(u'x',fontsize=14)
plt.ylabel(u'h',fontsize=14)
plt.show()
```

Results The dynamic change of the string is shown in *Figure*2, ranging from $0 \sim 4$.

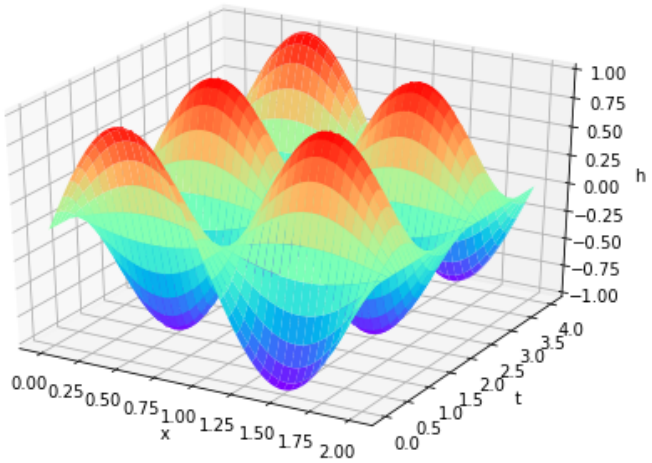


Fig. 3.6: Time-varying Image of the string

More specifically, the shapes of the string at different times are shown below:

Interpretation The results are consistent with our expectation. The shape of the string depends on the initial and boundary conditions. In this case, the string is in the shape of a sine wave that changes periodically.

The initial and boundary conditions can be changed easily at the code by modifying the function *initial*() and *speed*(). Readers can plot more complicated image with different initial displacement and speed function.

Conclusion Finite Difference Method is a practical method to obtain the numerical solution of a Partial Differential Equation. We can learn that it is applicable for one dimentional problems like 1D wave equation and 1D heat equation. It's not difficult to understand but there are some places worth special attention while writing the code. For example, the first iteration needs to be distinguished from other iterations since t_{N-1} is not defined.

3.2.3 1D traffic simulation

Prepared by subgroup 3 (Yue Hu and Carlin Yao)

Model presentation What is the model we want to simulate? What do we want to observe? Which is the state space and the dynamics?

Implementation Explain the structure of the code. Do not put necessarily all the code (not more than 100 lines) since some routines (functions) can hide efficiently some unnecessary complexity. Provide a code that run (and explicit librairies and dependencies). Ensure your file name is aligned with this part.

Results Explain the quantities you are studying (i.e. metrics and statistics). Provide good visualization.

Interpretation Relate these quantities to the model and to theoretical knowledge of the course.

Conclusion What have we learned? Is everything aligned (theory and practice)? What was difficult? Provide perspectives.

3.2.4 Random walk in 2D simulation

Prepared by subgroup 4 (Yue Hu, Carlin Liao and Robert Ruigrok)

Model presentation In this example we simulate the random walk of a particle in a 2D space. A random walk is a mathematical object, known as a stochastic or random process, that describes a path that consists of a succession of random steps. In order to simulate this process, we let a particle move over a discretized grid where its motion is drawn from a set of possible directions. In this simulation we are interested in finding expected distribution of particles after a certain number of time steps as well as the position where they hit the boundaries of the spatial grid.

A particle starts at a specified initial position, from where it begins moving through the grid. In our code, we used a coordinate system to represent the location of a particle as provided in figure 3.2.4. The outer border of the grid is enclosed by a "wall". When the particle hits this wall, its motion stops and the location where it makes contact is registered.

The dynamics of a particle are relatively straightforward and can be described by equation 10. A particle has 5 options for its motion: moving up, right, down, left or no motion (options are depicted in figure 3.2.4). Every motion has a certain probability p to occur. These probabilities can be given as input and must add up to 1.

$$X_{k+1}(x,y) = X_k(x,y) + \begin{cases} (0,1) & \text{with probability } p \uparrow \\ (1,0) & \text{with probability } p \rightarrow \\ (0,-1) & \text{with probability } p \downarrow \\ (-1,0) & \text{with probability } p \leftarrow \\ (0,0) & \text{with probability } p \bullet \end{cases} \tag{10}$$

Implementation At the top of the file, it is possible to set the grid size, starting position of particles, # of particles, simulation horizon and motion direction probabilities. The script will also generate snapshots of the particle distribution at different moments in time during the simulation. By selecting the number of subplots, you can determine how many instances you would like to see. This will provide insights into how the particle distribution develops over time.

Particles are simulated one at a time. The simulation runs until the time horizon T is reached or the particle hits the wall. Their position is saved in a 3-dimensional array (time, x-position, y-position) at specified moments in time only; this way the amount of required memory is kept to a minimum.

Information about where particles hit the wall is included in the same arrays. This data "circumvents" the $n \times n$ data about the particle distribution within the grid. As a result, when we plot the full array we can see the distribution of particles in the grid at their respective location, and the distribution of particles at the wall directly "behind" the wall.

Code Listing 3.2: 2D Ramdom Walk Simulation

```
import numpy as np
import matplotlib.pyplot as plt
from random import *
import pylab

############## START INPUT #################

GridSize = 20 # will create an 20x20 square grid
Pos_init = np.ceil(np.array([0.4,0.4])*GridSize)  # start position
T = 0.3*np.power(GridSizeSquare, 2)  # total # of time steps
n_particles = 10000 # number of particles, 10000+ recommended
# specify subplots for intermediate time snap shots
subplot_row = 2
subplot_column = 2
#define the probabilities of motion: ([up,right,down,left,0])
motion_prob = np.array([0.2,0.2,0.2,0.2,0.2])

############## END INPUT #################

x_grid = GridSizeSquare
y_grid = GridSizeSquare
n_subplot = subplot_row*subplot_column
# calculate time step for snapshots of process
Plot_interval = np.floor((T-1)/(n_subplot-1))
# Normalize probabilities in case they don't add up to 1
motion_prob = motion_prob/np.sum(motion_prob)
# define the change in coordinates of every motion:
motion_xy = np.array([[0,1],[1,0],[0,-1],[-1,0],[0,0],])
motion_prob_sum = np.cumsum(motion_prob)
# this is used later to draw from with randomizer

# make an empty data grid to sum the particle positions
Data = np.zeros((x_grid+2,y_grid+2))  # +2 for wall data
```

```
# Some resizing here and later are for ploting purposes
Data_resized = np.zeros((Data.shape[0]+1,Data.shape[1]+1))
# Create a new empty data set for the intermediate plots:
Data_Snap=np.zeros((n_subplot,Data_resized.shape[0],Data_resized.shape[1]

# construct some arrays for plotting later on:
xx, yy = pylab.meshgrid(
    pylab.linspace(-1,x_grid+1,x_grid+3),
    pylab.linspace(-1,y_grid+1,y_grid+3))

# start loop over all the particles
for i in range(1, n_particles+1):
    # Initialize simulation
    t = 0
    HitWall = False
    Pos = Pos_init
    Subplot = 1

    # Simulation Process
    while t < T and not HitWall:
        MotionRandom = random()
        IndexMotion = np.argmax(motion_prob_sum>MotionRandom)
        Pos = Pos + motion_xy[IndexMotion,:]

        # Now check for hitting the wall
        if Pos[0] == 0|x_grid+1 or Pos[1] == 0|y_grid+1:
            if Subplot <= n_subplot:
                Data_Snap[Subplot-1,Pos[0],Pos[1]] = \
                Data_Snap[Subplot-1,Pos[0],Pos[1]]+1
            HitWall = True

        # Record the position for each time snap
        if t % Plot_interval==0 and Subplot <= n_subplot and not(HitWall)
            Data_Snap[Subplot-1,Pos[0],Pos[1]] =\
            Data_Snap[Subplot-1,Pos[0],Pos[1]]+1
            Subplot = Subplot+1

        t = t+1

    #Save the final results
    Data[Pos[0],Pos[1]] = Data[Pos[0],Pos[1]] + 1
```

```
    Data_resized[:-1,:-1] = Data

# Add the particles that hit the wall in earlier time steps
# to the later plots
Data_Snap_New = np.cumsum(Data_Snap, axis=0)
Data_Snap_New[:,1:x_grid+1,1:y_grid+1] = Data_Snap[:,1:x_grid+1,1:y

# Visualize the outcomes at each snap of process
plt.figure()
for j in range(1, n_subplot+1):
    pylab.subplot(subplot_row, subplot_column, j)
    pylab.pcolor(xx,yy,np.transpose(Data_Snap_New[j-1,:,:]))
    Str = 'Distribution at t = ' + str((j-1)*Plot_interval+1)
    pylab.title(Str)
    # add a color bar
    pylab.colorbar()
    pylab.hold(True)
    pylab.plot([0, x_grid],[0, 0], 'r',
               [0, x_grid],[y_grid, y_grid], 'r',[0, 0],[0, y_grid]
               [x_grid, x_grid],[0, y_grid], 'r')
    pylab.plot(Pos_init[0]-0.5,Pos_init[1]-0.5,'ro')

pylab.show()

# Plot the final distribution
plt.figure()
pylab.pcolor(xx,yy,np.transpose(Data_resized))
pylab.title('Final distribution at t = %d, including hitting walls'
# add a color bar
pylab.colorbar()
pylab.hold(True)
pylab.plot([0, x_grid],[0, 0], 'r',
           [0, x_grid],[y_grid, y_grid], 'r',[0, 0],[0, y_grid], 'r
           [x_grid, x_grid],[0, y_grid], 'r')
pylab.plot(Pos_init[0]-0.5,Pos_init[1]-0.5,'ro')
pylab.show
```

Results In this section we included two simulation for both a $10x10$ grid and a 20×20 grid, with a different simulation time horizon T. Both simulations use 10,000 particles and have a motion probability of 0.2 in all direc-

tions.

It is clearly visible how the particles spread out over time and make their way to the walls over time. The more particles that are simulated per grid resolution, the smoother the distribution becomes. You can clearly see that 10,000 particles simulated lead to a clean distribution in the smaller plot of figure 3.2.4, while the larger plot of 3.2.4 shows a more grainy distribution.

Interpretation *Relate these quantities to the model and to theoretical knowledge of the course.*

I think this motion is described by Fick's Law in 2 dimension. The concentration on a specific point changes over time, depending on the concentration of its surroundings. Should we derive why the second derivative matters? ϕ is the concentration, D the diffusion coefficient.

$$\frac{d\phi}{dt} = D\nabla\phi = D\big(\frac{d^2\phi}{dx^2} + \frac{d^2\phi}{dy^2}\big) \tag{11}$$

Conclusion *What have we learned? Is everything aligned (theory and practice)? What was difficult? Provide perspectives.*

The dynamics of the random walk were easy to model. The challenge in this simulation was to save the data for the intermediate time steps. Since the particles are simulated one by one for the full time horizon, we had to write some non-intuitive code to save the location of every particle at the relevant intermediate time steps.

From the lecture we recall that diffusion distance scales with the square root of time. Here we tried to simulate that. When doubling the grid size and taking a four times higher simulation horizon, the distribution looks similar. However, we have the idea that the scaling did not work for 100%. At the end of the scaled simulation horizon, it seems as if the smaller grid has relatively more particles in the grid than the larger grid has. What could cause this difference?

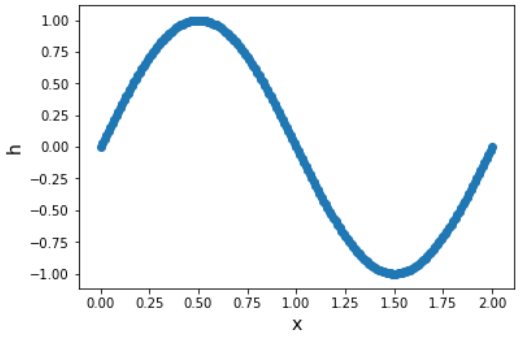


Fig. 3.7: *

t=0

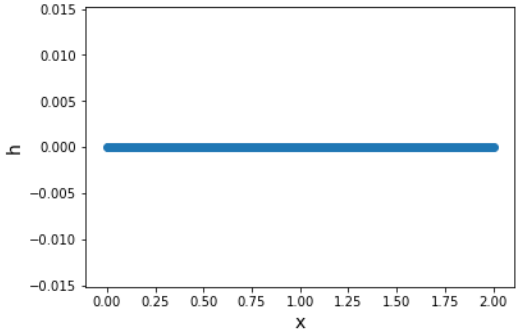


Fig. 3.8: *

t=0.5

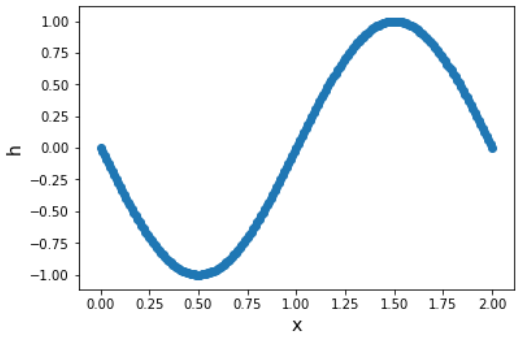


Fig. 3.9: *

t=1

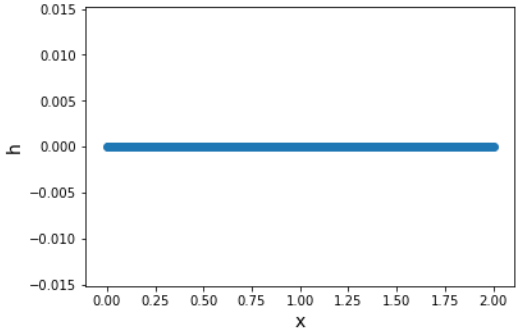


Fig. 3.10: *

t=1.5

Fig. 3.11: Shape of the string at time t

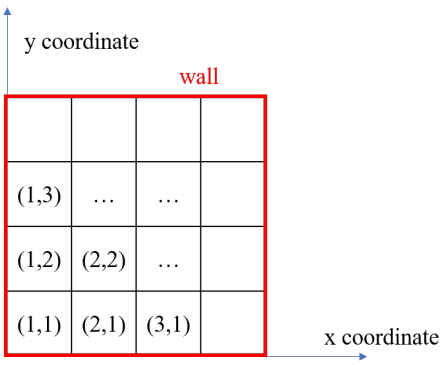


Fig. 3.12: Coordinate representation in spatial grid

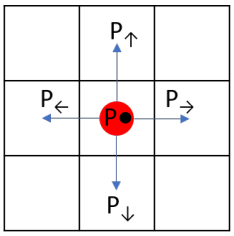


Fig. 3.13: Potential motion per time step

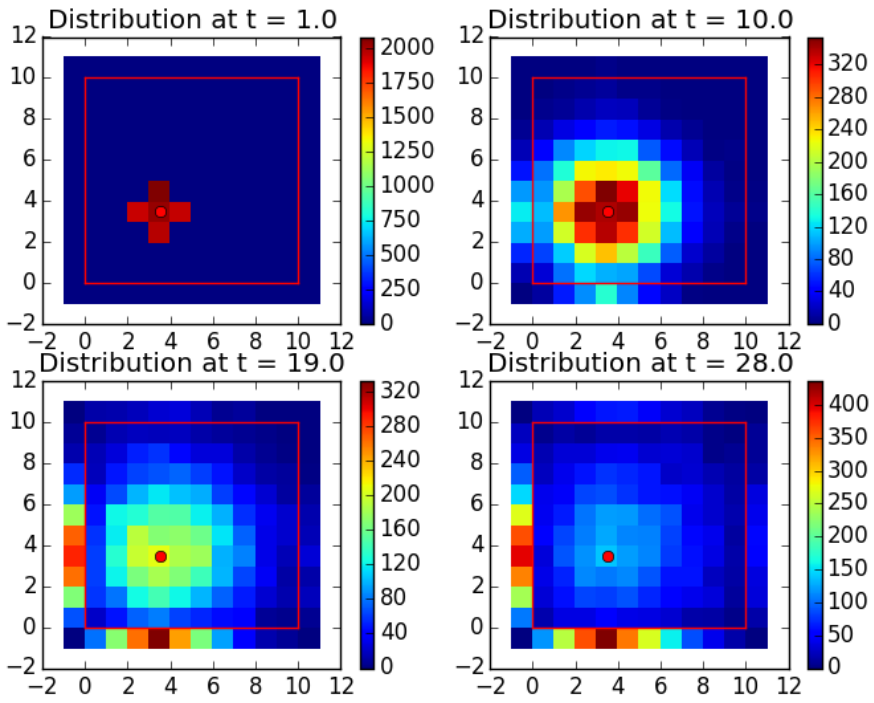


Fig. 3.14: Particle distribution for 10×10 grid at different time steps

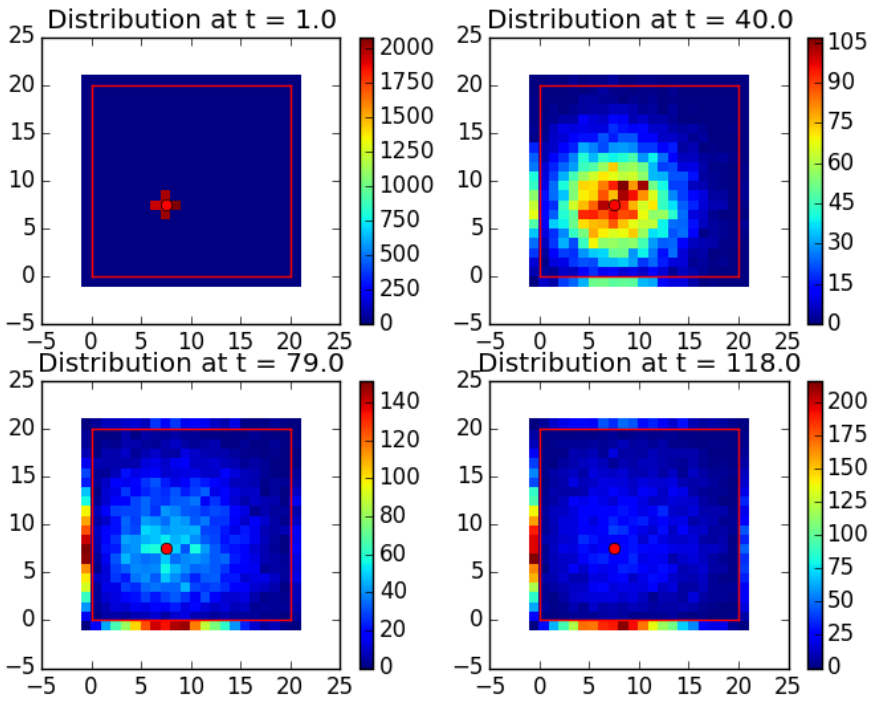


Fig. 3.15: Particle distribution for 20×20 grid at different time steps

Chapter 4

Control and optimization

4.1 Objectives

Once a system has been enough understood, there is still to do the main work: enhance it. This means most of the time optimization, and control can be seen as an online optimization (i. e. continuously making the system better). There are several control techniques as there are many optimization techniques. Since this reader aims at keeping things practical, we will not develop all possible ideas here. Instead we insist onto 2 main ideas:

1. explicit a criteria to be optimized: in principle, this is rooted in the system description and what we want to do with the system. There are many ways to enhance a system, and very often the models are too complicated to have an explicit optimum (whenever they have one). But without criteria, there is even no way to know what is better (or worse).

2. simplify the model so that the optimization (or control) problem can be solved by simple tools. As a rule, it is often better to get a simple but efficient control rather than to try to compute the optimal one. This rules also holds for optimization: the best is the enemy of the good; well enough is enough in lots of practical systems.

Therefore some tools for PDE are applied in the next sections as well as simple tools that can be applied by smartly simplifying the problem.

4.2 Tools and examples

4.2.1 Kalman filter

Control model We consider here the simplest model for which a Kalman filter can really bring an added value: a linear, Gaussian and real (i.e. 1D) model:

$$x_{k+1} = Ax_k + Bu_k + v_k \quad (1)$$
$$y_k = Cx_k + w_k \quad (2)$$

where $X = (x_k)$ is the state of the system (typically a distance to a prescribed value, in whatever unit), $U = (u_k)$ is the control, $Y = (y_k)$ is the measurement. $V = (v_k)$ (resp. $W = (w_k)$) is a Gaussian noise: the v_k (resp. w_k)are independent and identically distributed random variables with 0 mean and variance Q (resp. R). The initial state x_0 is also a centered random variable with variance $\bar{P}_O$. Note that these notation perfectly adapt to multidimensional systems, with matrices: see any textbook on control to get the matrix equations.

This is not specifically needed for Kalman filter, but the usual optimization criteria is quadratic:

$$J(X, U) = \frac{1}{N} \sum_{k=0}^{N} \alpha u_k^2 + \beta x_k^2 \quad (3)$$

The Equations (1)-(2) are often written for each time step so the scaling with respect to time has to be careful. In a system with a 100 Hz time step (also refered to as control frequency), typical values are:

$$\begin{aligned} A &= 1.001 \\ B &= .01 \\ C &= 1 \\ Q &= .001 \\ R &= .01 \end{aligned}$$

We have taken $C = 1$ because most of the time we try to have a measurement as close to the state as possible and if $C \neq 1$ it is possible to compensate it. $A > 1$ means that the system is unstable: Figure 4.1 shows the divergent behavior.

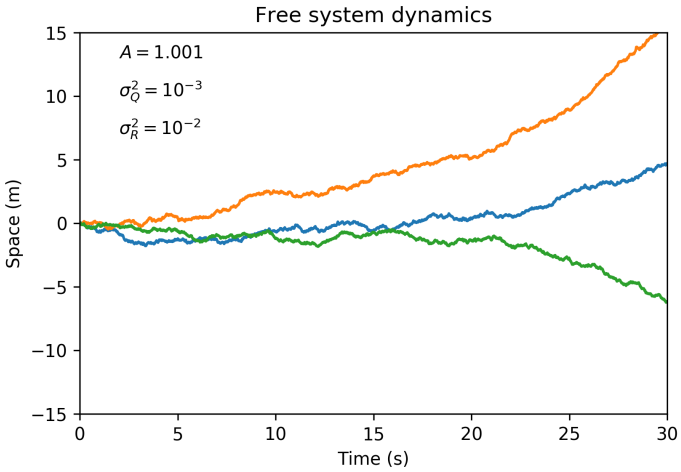


Fig. 4.1: The free system dynamics: it shows instability and the system tends to diverge exponentially. The variance σ_R^2 (resp. σ_Q^2) is denoted by R (resp. Q) in the system model.

A naive control Now, we would like to stabilize the state of the system as close to zero as possible, and if possible at a reasonable cost. This means we can adjust the weights α and β in the cost function (3). Or simpler measure the 2 partial cost functions

$$J_1(X) = \frac{1}{N}\sum_{k=0}^{N} x_k^2 \tag{4}$$

$$J_2(U) = \frac{1}{N}\sum_{k=0}^{N} u_k^2 \tag{5}$$

Now, we introduce the simplest possible control: a proportional control with gain K defined by:

$$u_k = -Ky_k \tag{6}$$

Since we have "unit" parameters, let's start with a unit gain $K = 1$. This indeed stabilizes the system as we see in Figure 4.2.

Now, if we try to maximize the precision (at all cost), we can try to compensate the term Ax_k in Equation (1) by the term $Bu_k = -BKY_k = -(BKC)x_k - BKw_k$. Since we have no control onto the noise, and since this noise is centered, let's take $A = BKC$, i.e. in our case $K = A/BC = 100.1$. This lead to the result of Figure 4.3. What is impressive is not so much the increase in precision, J_1 is divided by about 5, that the explosion of the control cost J_2, multiplied by about 3,000.

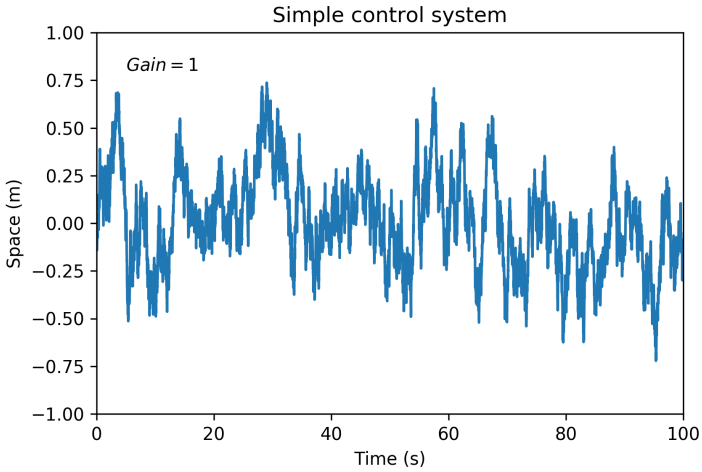


Fig. 4.2: The same system as in Figure 4.1 with the proportional control as defined by Equation (6) with gain $K = 1$. The system is indeed stable with empirical costs $J_1 = 0.06$ and $J_2 = 0.07$.

In order to learn more about the trade-off between precision and control cost, one can test intermediary gains, and for $K = 10$, we get the Figure 4.4. There we have a surprise: the precision is better than with our naively optimal gain $K = 100.1$, not to speak of the decrease of the control cost. There are now 2 paths for us: either — pragmatically — adjust the gain to get the best control, or — analytically — try to understand what was wrong. Indeed, this numerical experiment clearly shows we are missing an important idea. And the idea we are missing is the following: we can have much better estimates of x_k than y_k (even after compensating some multiples in the C coefficient. What we do with the heavy gain $K = 100.1$ is simply to add noise. And if we go even further (e.g. $K = 200$), the system becomes unstable.

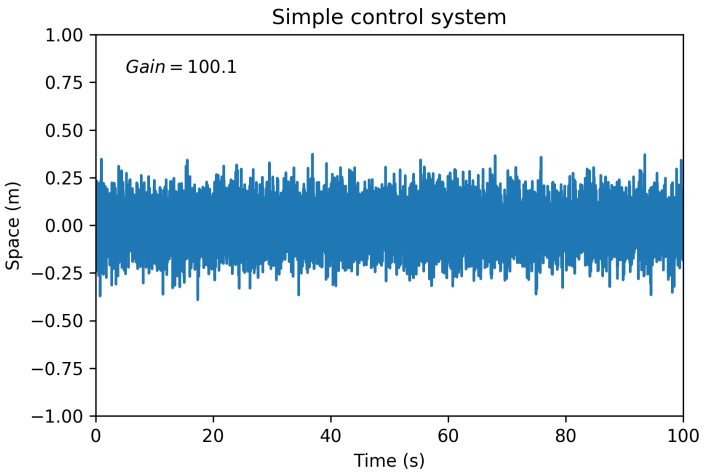


Fig. 4.3: The same system as in Figure 4.2 with $K = 100.1$ in order to fully compensate A. The system is very stable with empirical costs $J_1 = 0.011$ and $J_2 = 209$.

The Kalman filter Now, we introduce the equations of the Kalman filter, which we know are optimal to estimate x_k.

$$\begin{aligned}
\hat{x}_0 &= 0 && (7)\\
P_0 &= \bar{P}_0 && (8)\\
\hat{x}^-_{k+1} &= A\hat{x}_k + Bu_k && (9)\\
P^-_{k+1} &= AP_kA + Q && (10)\\
\hat{x}_{k+1} &= \hat{x}^-_{k+1} + L_{k+1}\left(y_{k+1} - C\hat{x}^-_{k+1}\right) && (11)\\
L_{k+1} &= P^-_{k+1}C\left(CP^-_{k+1}C + R\right)^{-1} && (12)\\
P_{k+1} &= \left(1 - L_{k+1}C\right)P^-_{k+1} && (13)
\end{aligned}$$

These equations can be found in any textbook on control and are almost true as is for multidimensional states and measures. The most striking here is that the filter works whatever the control.

There is not much difference with Equation (6) in our control model, except that we take the best estimate:

$$u_k = -K\hat{x}_k \tag{14}$$

This leads to the result of Figure 4.5. The precision is better than our previous best control (that was obtained for a lower gain) and the control cost is much lower than the control with same gain but based on Equation (6).

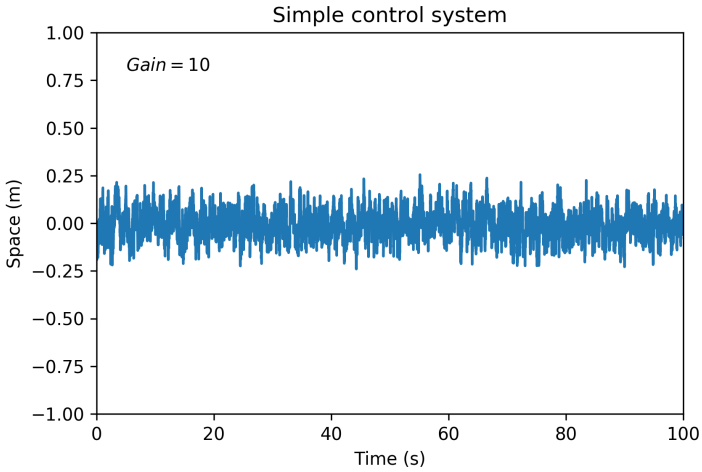


Fig. 4.4: The same system as in Figure 4.2 with $K = 10$. The empirical costs are $J_1 = 5.8\ 10^{-3}$ and $J_2 = 1.59$.

Empirically, one sees that this gain is the optimal value in term of precision. As we are curious, we also vary the gain and we get the following results: for $K = 10$, $J_1 = 8\ 10^{-3}$ and $J_2 = 0.53$; for $K = 1$, $J_1 = 0.072$ and $J_2 = 0.069$. So we see that for lower gains (and this is much closer to realistic values), the precision is not much improved and the control cost improvement is noticeable mainly for high gains. Again, this certainly means we are missing some concepts. And here what we are missing is the notion of a better feedback control: the proportional control is too simple to be optimal. We could try a PID (Proportional-Integral-Derivative) or other controls (e.g. with pole placement), but in any case we learned that a Kalman filter allow to reduce the noise in the estimation of the state x_k.

In conclusion, note that what we learned practically about estimates, noise reduction and control are very general concepts that can be adapted to many situations, including PDEs. However, it generally requires some know-how to apply these notions.

4.2.2 PID control

4.2.3 Adjoint optimization

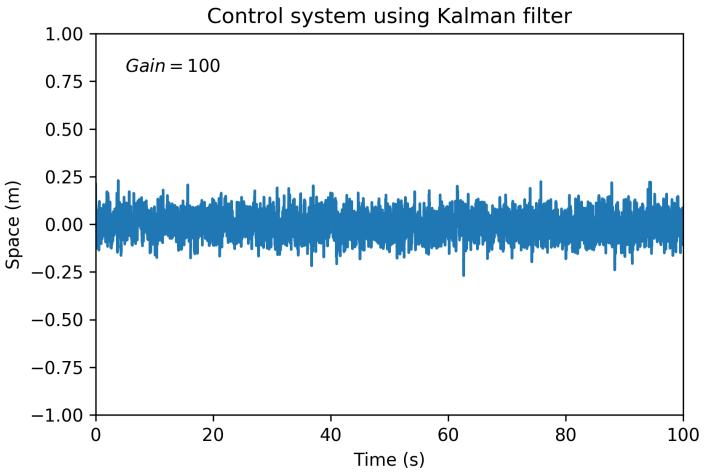


Fig. 4.5: The system with a proportional control based on Kalman filter with $K = 100.1$. The empirical costs are $J_1 = 3.77\ 10^{-3}$ and $J_2 = 10.2$.