School of international Liberal Studies
Waseda University
Advanced Course: Virtual Earth
Spring 2025

Problem Set 4

1. For which right sides (find a condition on b_1, b_2, b_3) are these systems solvable?

(a) $$\begin{pmatrix} 1 & 4 & 2 \\ 2 & 8 & 4 \\ -1 & -4 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

(b) $$\begin{pmatrix} 1 & 4 \\ 2 & 9 \\ -1 & -4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

2. What are the special solutions to $\mathbf{Ax} = \mathbf{0}$ for these $\mathbf{A}$?

(a) $$\begin{pmatrix} 1 & 0 & 2 & 3 \\ 0 & 1 & 4 & 5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(b) $$\begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

3. The solution to any partial differential equation depends not only on the "physics" embodied by that equation, but also on the nature of the boundary conditions (BCs). For example, in the 1-dimensional heat diffusion problem with specified temperature at the two ends of the metal bar, the final steady state solution is always a "straight line" connecting the temperatures at the ends. That is, it is doesn't matter what the initial distribution of temperature is. There is a unique solution to this problem. On the other hand if the metal bar is insulated at the two ends so that no heat flows in or out, i.e., a zero flux BC represented mathematically by $\partial T/\partial x = 0$ at $x = 0$ and $x = L$, the final steady state temperature distribution will be just a uniform constant along the bar whose actual value depends on the initial condition. The physical reason for this is that the energy in the bar is conserved (the ends are insulated) and so the final temperature of the bar depends on how much energy you started out with.

   What does this mean from a mathematical and numerical point of view? Regardless of the type of BCs, the spatially-discretized diffusion equation can always be written in matrix form as: $d\mathbf{T}/dt = \mathbf{AT} - \mathbf{b}$, where $\mathbf{b}$ contains the boundary conditions. But the matrix $\mathbf{A}$ is very different for the two BCs. With specified temperature, $\mathbf{A}$ is invertible and the steady state solution ($d\mathbf{T}/dt = \mathbf{0}$) is simply: $\mathbf{T}_{\text{steady}} = \mathbf{A}^{-1}\mathbf{b}$. But with specified fluxes (zero or otherwise) $\mathbf{A}$ is *not* invertible and one cannot find the steady state solution directly. (You can still time-step the equation toward a steady state. That will always work.)

To see this, in this problem you will explicitly work out the matrix $\mathbf{A}$ for a situation where energy in the metal bar is conserved. But because the problem of insulated ends is somewhat tricky (although I urge you to give it a go), we will consider a slightly different situation: a metal bar that is bent into a circle so that the two ends are connected. In essence then we have a metal bar (or rather ring) without any physical "boundaries". What, then, is the "boundary condition"? That's easy: if L is the length of the original bar and x the distance along the bar after it has been bent into a ring, the temperature at $x = 0$ must be equal to the temperature at $x = L$. This is because these two "ends" are at physically the same point in space and there cannot be a "jump" in the temperature. (This is not always the case and it takes a bit of math to show why a discontinuity is not allowed for this particular equation.) This is known as a "periodic boundary condition" and it is one of the most common types of BCs you will encounter. Mathematicians love it because it simplifies their lives. But it is also applicable to the real world. For example, the Earth is a sphere and obviously the temperature of the atmosphere (or anything else for that matter) doesn't show a sudden jump when you go from just west of 0° longitude to just east! We can therefore think of temperature in the atmosphere as satisfying periodic boundary conditions.

Now on to the actual problem:

(a) Use finite differences to discretize the spatial derivatives with a grid spacing of Δx and apply a periodic boundary condition at $x = 0$. Note that unlike before we don't know the solution at $x = 0$ and we must solve for it by including it in our $\mathbf{T}$ vector that represents temperature at the grid points. (We don't know the temperature at $x = L$ either, but since $x = 0$ and $x = L$ are the same point and $T(x = 0, t) = T(x = L, t)$ we don't need to include it in $\mathbf{T}$.)

(b) You should end up with an equation like $d\mathbf{T}/dt = \mathbf{AT}$. The steady state solution to it satisfies $\mathbf{AT}_{\text{steady}} = \mathbf{0}$. Mathematically, this solution lies in what is called the nullspace of $\mathbf{A}$ (N($\mathbf{A}$)), i.e., the set of all $\mathbf{x}$ that satisfy $\mathbf{Ax} = \mathbf{0}$. But if $\mathbf{x}$ and $\mathbf{y}$ both lie in the nullspace, then so does $\mathbf{x} + \mathbf{y}$ (or any linear combination of them). There are therefore an infinite number of solutions to $\mathbf{Ax} = \mathbf{0}$ (or in our case $\mathbf{AT}_{\text{steady}} = \mathbf{0}$). Which, then, is the "correct" one? Or in other words, which solution does nature "know" to pick? Explain how *you* would pick the correct solution out of the infinite number at your disposal.

(c) Solve the above problem using any of the Euler time-stepping methods. You can use the same parameter values for κ, Δx, etc as in problem set 3, but with an initial condition for temperature described by a "hat" function: $T(x, t = 0) = (2/L)x$ for $0 \leq x \leq L/2$ and $T(x, t = 0) = (2/L)(L - x)$ for $L/2 \leq x \leq L$. Make plots of the transient and final steady-state solution.

4. We've spent a lot of time in class on the diffusion equation, which was first derived by

Fourier in 1822 to describe heat flow, and subsequently by Fick in the 1850s to describe the diffusion of a chemical in a fluid. But these derivations were phenomenological and it wasn't until much later that a truer understanding emerged, famously with Einstein's 1905 paper elucidating the theoretical mechanism of Brownian motion. (As is well known, in this "annus mirabilis", Einstein published 4 papers: the photoelectric effect, for which he won the Nobel Prize; special relativity; mass-energy equivalence, i.e., $E = mc^2$; and of course Brownian motion, his most cited work.)

The problem Einstein set out to address was why tiny particles on the surface of a fluid (originally, charcoal sprinkled on alcohol) seemed to jump around randomly, i.e., perform what we now call a "random walk". Einstein invoked the then still controversial kinetic theory of gases to show that the probability of a Brownian particle's displacement at time t being between x and $x + dx$ is given by $P(x, t)dx$, where P was governed by:

$$\frac{\partial P}{\partial t} = D\frac{\partial^2 P}{\partial t^2},$$

which you will recognize as the diffusion equation with diffusivity D. (That probability can diffuse like heat was shown a few years earlier by Louis Batchelier in his 1900 Ph.D. thesis on stock market fluctuations, an insight widely used today in the form of the Black-Scholes equation to price options.) The solution to this equation is a Gaussian:

$$P(x, t) = \frac{1}{\sqrt{4\pi Dt}}e^{-x^2/4Dt},$$

which shows that the mean square position of a Brownian particle (i.e., the variance of its displacement) increases linearly in time, specifically as $\sigma^2 = 2Dt$.

But Einstein being, well Einstein, had bigger fish to fry than figuring out how a lump of sugar diffuses in a cup of tea. Bizarre as it may sound to us, at that time the existence of atoms was still debated and what Einstein had really set out to do was establish their reality. His *pièce de résistance* was to show that the diffusivity $D = 2RT/N_A f$, where R is the gas constant, T the absolute temperature, f a friction parameter that depends on the viscosity of the fluid and size of the particle, and N_A the Avogadro number. A few years later Jean Perrin experimentally verified Einstein's prediction that the mean square displacement of a Brownian particle increases linearly in time at a rate proportional to D and, using the above expression for D, calculated Avogadro's number (for which he was awarded the Nobel prize). It established once and for all the atomic nature of matter.

Along with work a few years later by Paul Langevin, Einstein's Brownian motion paper also established entire new fields of physics and maths, including a whole new type of calculus known as stochastic calculus, with its very own stochastic differential equations (in the terminology of this field, the diffusion equation is just a special case of a whole class of differential equations known as the Fokker-Planck equation).

Why does any of this concern us? Because random walks, stochastic differential equations and such are widely applied in our field, from modeling the behavior of aerosols in clouds (of massive importance to making accurate projections of climate change) to figuring out where microplastics end up in the ocean.

(a) Write a simple program to simulate the trajectory of a Brownian particle in one dimension (the x axis). The particle starts at $x = 0$ and in each time increment Δt, it can move in discrete steps of $\pm\Delta x$. How do we decide whether it moves in the $+$ direction or $-$ direction? We toss a coin. Heads, it moves by $+\Delta x$; tails by $-\Delta x$. How does one "toss a coin" in a computer program? You use a "random number generator" (RNG), a function that returns a random number with some specified characteristics. We don't need to get into the details of how RNGs work, except to know that they involves some pretty complex maths and computer science. And that the sequence of numbers they produce are not strictly random (which is why they're more correctly called "pseudo-random"). In Matlab, the most important function for this is called `rand` (in Python see the `numpy.random` module). It produces a uniformly distributed random number between 0 and 1, which is to say that any number (that can be represented on a computer) in that interval is equally likely. So one way to "toss a coin" is to call `rand` and if the returned value is < 0.5, we consider that "heads"; otherwise it is tails. The important thing is that the outcome does not depend on the present or past location of the particle: the process is "memoryless".

   You'll notice that the time increment Δt does not really come into the picture. So you can set it to "1". Similarly, for our present purposes, you can also set $\Delta x = 1$. Simulate the trajectory of the particle for a few hundred time steps and plot the position of the particle as a function of time.

(b) In order to test Einstein's prediction that the mean square position of a Brownian particle increases linearly with time, one trajectory of a random walker is not enough. We need a "lot" of them, each one independent of the other. You could run your program many times to generate an "ensemble" of trajectories. Or you could simply simulate a "swarm" of Brownian particles, each one doing its own thing. It really doesn't matter. Let's call $x_i(t)$ the position of the i^{th} particle a time t. For a fixed time t, the set of $x_i(t)$'s is just a set of random numbers (the positions of the particles) whose variance is the required mean square. In Matlab you can calculate this (or rather its square root) with `std`.

   Make a plot of the trajectories of your ensemble/swarm of particles as a function of time. Also plot the mean square position as a function of time to show that Einstein got his maths right. To get meaningful results you'll need a pretty decent sample size. Play around with the number of particles until you get results that are reasonably smooth

and stable, i.e., don't change as you increase the number of particles.

(c) Use the solution from above to show that the probability $P(x,t)dx$ of a particle lying between x and $x+dx$ at time t is a Gaussian (see above). To do so look up the Matlab functions `hist` and `histc`.

5. **Bonus problem**: Consider a night out in London where you and your friends have had a bit too much to drink (the "do not try this at home" fine print applies!). In fact, you're so inebriated that you random-walk your way through the Underground by taking the tube from one station to another. If there are multiple tube lines at a station, you pick one at random and take it to the next station. Rinse and repeat.

   How can we mathematically represent the London Underground? We use the concept of a graph (`https://en.wikipedia.org/wiki/Graph_theory`). Very simply (and simplistically), a graph is a collection of points ("nodes" or "vertices") connected to each other by lines ("edges"). Here, the stations are the vertices and the tube lines the edges. A node can be connected to another by more than one line. The more lines there are linking two nodes, the stronger the connection between them. (Search engines such as Google's are based on representing the internet as one giant graph.) In this problem you will simulate a drunken trip on the Underground. If you can find appropriate data for Tokyo or other cities then feel free to simulate those instead. I'd appreciate if you can share the data with me. (You might have some luck here: `https://www.odpt.org/` and here: `https://github.com/Jugendhackt/tokyo-metro-data`.)

   (a) First, download the file london_underground_connectivity.mat. It is based on data I obtained from `https://github.com/nicola/tubemaps/tree/master/datasets`, specifically the files `london.connections.csv` and `london.stations.csv` from that website. I have cleaned up the original data and constructed a connection graph.

   (b) Load the file into Matlab (see below for how to load it into Python). This file contains a connection matrix $\mathbf{C}$ such that $C(i,j)$ is the number of tube lines connecting station i and station j. By definition this is a symmetric matrix, i.e., $C(j,i) = C(i,j)$ The file also contains a cell array `stationName` such that `stationName{i}` is the name of station i. Thus, `C(11,104)` has the value 3 indicating that there are 3 lines connecting Baker Street (#11) and Great Portland Street (#104) stations. How do we know this? In Matlab:

```
>> stationName{11}

ans =
```

```
    'Baker_Street'

>> stationName{104}

ans =

    'Great_Portland_Street'
```

On the other hand, to do the reverse, i.e., figure out the index of a particular station, we search through `stationName` until we find a match. How do we do that? Easy-peasy:

```
>> find(contains(stationName,'Baker_Street'))

ans =

    11

>> find(contains(stationName,'Picadilly_Circus'))

ans =

   197
```

To load the file in Python, install the `pymatreader` module and do:

```
import numpy as np
from pymatreader import read_mat
g=read_mat('london_underground_connectivity.mat')
```

This loads a dictionary `g` whose members are NumPy arrays for the connection matrix C (i.e., `g['C']`), the longitudes `g['stationLongitude']`, and the latitudes `g['stationLatitude']`, and a list of station names `g['stationName']`. Note: If you're trying to reproduce the above Matlab code remember that in Matlab indexing starts at 1 whereas in Python it starts at 0. So subtract 1 from all indices in the above Matlab code. Thus, if you do `g['stationName'].index('Picadilly_Circus')`, you should get 196.

(c) Suppose you're at station j. How do you pick the station to go to next? (If there is only one station connected to j, then obviously that's where you go.) If j is connected to two stations by a single tube line each, the probability of picking either one is 50% and you could "toss a coin" as in the previous problem. If there are three stations connected to j by a single line each then the probability of picking a particular one is 33.3333%. And so on. But what if there are multiple lines connecting j with another station? Then,

since we're assuming that every tube line is equally likely, the probability of picking that station to go to is proportional to the number of connecting lines. Thus, if our network is made up of three stations A, B and C, with A and B connected by two lines and A and C by one, a random walker at A is twice as likely to end up at B than C. (If you can find the appropriate data, you can make this more interesting by, for example, (inversely) weighting the probability of taking a particular line by the distance or ticket price to the next station on that line.)

To put these ideas into practice we start by looking at the j-th column of $\mathbf{C}$. These are the number of lines connecting j with each of the other stations. Most will be zero. If you divide each number by the total number of lines j is on, you get the probability of picking the corresponding station. Let's call these probabilities p_i. Suppose there are 3 stations connected to j. Imagine breaking up the interval between 0 and 1 into 3 bins whose width are given by p_1, p_2 and p_3. By definition, the p_i's sum up to exactly one. So a uniformly distributed random number must fall into one of those bins and the likelihood of falling into any particular bin (i.e., picking the corresponding station) is proportional to the width of that bin.

Now implement those ideas in a function `updateParticlesOnGraph` that takes your current station index j_{curr} and the connection matrix $\mathbf{C}$ as arguments, and returns the index j_{new} of the station you go to next.

(d) Use the above function in a program to simulate your drunken trip through the Underground starting at a station of your choice.

(e) The file I've provided contains the arrays `stationLongitude` and `stationLatitude` which have the positions of the stations (e.g., `stationLongitude(11)` is the longitude of station 11 which, as we know, is Baker Street). Using this data make a plot of your trajectory. You can just use `plot`. But if you're feeling ambitious try it with the `webmap` and associated functions (you will need to install the mapping toolbox): `https://www.mathworks.com/help/map/ref/webmap.html`. (It is possible to do this in Google Maps but not programmatically, at least not that easily.) Make an animation of your path overlaid on all the station locations.

(f) Once you have a working program, you can try all kinds of interesting things. For example, you could ask how many times you visit each station or if there are stations you visit more often than others. Or your friends could start with you but then go their own way so that you have a swarm of drunken random walkers. And so on.