Assignment 2 - Solving two 1D problems

Contents

This assignment makes up 20% of the overall marks for the course. The deadline for submitting this assignment is **5pm on Thursday 3 November 2022**.

Coursework is to be submitted using the link on Moodle. You should submit a single pdf file containing your code, the output when you run your code, and your answers to any text questions included in the assessment. The easiest ways to create this file are:

- Write your code and answers in a Jupyter notebook, then select File -> Download as -> PDF via LaTeX (.pdf).
- Write your code and answers on Google Colab, then select File -> Print, and print it as a pdf.

Tasks you are required to carry out and questions you are required to answer are shown in bold below.

The assignment

Part 1: Solving a wave problem with sparse matrices

In this part of the assignment, we want to compute the solution to the following (time-harmonic) wave problem:

$$\begin{aligned} \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + k^2 u = 0 \qquad & \text{in } (0,1), \\ u = 0 \qquad & \text{if } x = 0, \\ u = 1 \qquad & \text{if } x = 1, \end{aligned}$$

with wavenumber $k = 29\pi/2$.

In this part, we will approximately solving this problem using the method of finite differences. We do this by taking an evenly spaced values $x_0 = 0, x_1, x_2, \ldots, x_N = 1$ and approximating the value of u for each value: we will call these approximations u_i. To compute these approximations, we use the approximation

$$\frac{\mathrm{d}^2 u_i}{\mathrm{d}x^2} \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2},$$

where $h = 1/N$.

With a bit of algebra, we see that the wave problem can be written as

$$(2 - h^2 k^2) u_i - u_{i-1} - u_{i+1} = 0$$

if x_i is not 0 or 1, and

$$\begin{aligned} u_i = 0 \qquad & \text{if } x_i = 0, \\ u_i = 1 \qquad & \text{if } x_i = 1. \end{aligned}$$

This information can be used to re-write the problem as the matrix-vector problem $\mathrm{A}\mathbf{u} = \mathbf{f}$, where A is a known matrix, $\mathbf{f}$ is a known vector, and $\mathbf{u}$ is an unknown vector that we want to compute. The entries of $\mathbf{f}$ and $\mathbf{u}$ are given by

$$\begin{aligned} [\mathbf{u}]_i &= u_i, \\ [\mathbf{f}]_i &= \begin{cases} 1 & \text{if } i = N, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The rows of A are given by

$$[\mathrm{A}]_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$

if $i = 0$ or $i = N$; and

$$[\mathrm{A}]_{i,j} = \begin{cases} 2 - h^2 k^2 & \text{if } j = i, \\ -1 & \text{if } j = i + 1, \\ -1 & \text{if } j = i - 1. \\ 0 & \text{otherwise,} \end{cases}$$

otherwise.

Write a Python function that takes N as an input and returns the matrix A and vector f. You should use an appropriate sparse storage format for the matrix A.

The function `scipy.sparse.linalg.spsolve` can be used to solve a sparse matrix-vector problem. Use this to **compute the approximate solution for your problem for $N = 10$, $N = 100$, and $N = 1000$.** Use `matplotlib` (or any other plotting library) to **plot the solutions for these three values of N.**

Briefly (1-2 sentences) comment on your plots: How different are they to each other? Which do you expect to be closest to the actual solution of the wave problem?

This wave problem was carefully chosen so that its exact solution is known: this solution is $u_{\text{exact}}(x) = \sin(kx)$. (You can check this by differentiating this twice and substituting, but you do not need to do this part of this assignment.)

A possible approximate measure of the error in your solution can be found by computing

$$\max_i |u_i - u_{\text{exact}}(x_i)|.$$

Compute this error for a range of values for N of your choice, for the method you wrote above. On axes that both use log scales, **plot N against the error in your solution**. You should pick a range of values for N so that this plot will give you useful information about the methods.

For the same values of N, **measure the time taken to compute your approximation for your function**. On axes that both use log scales, **plot N against the time taken to compute a solution**.

We now want to compute an approximate solution where the measure of error is 10^{-8} or less. By looking at your plots, **pick a value of N that you would expect to give error of 10^{-8} or less**. **Briefly (1-2 sentences) explain how you picked your value of N and predict how long the computation will take**.

Compute the approximate solution with your value of N. Measure the time taken and the error, and **briefly (1-2 sentences) comment on how these compare to your predictions**. Your error may turn out to be higher than 10^{-8} for your value of N: if so, you can still get full marks for commenting on why your prediction was not correct. Depending on your implementation and your prediction, a valid conclusion in the section could be "My value of N is too large for it to be feasible to complete this computation in a reasonable amount of time / without running out of memory".

Part 2: Solving the heat equation with GPU acceleration

In this part of the assignment, we want to solve the heat equation

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= \frac{1}{1000}\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} && \text{for } x \in (0,1), \\ u(x,0) &= 0, && \text{if } x \neq 0 \text{ and } x \neq 1 \\ u(0,t) &= 10, \\ u(1,t) &= 10. \end{aligned}$$

This represents a rod that starts at 0 temperature which is heated to a temperature of 10 at both ends.

Again, we will approximately solve this by taking an evenly spaced values $x_0 = 0, x_1, x_2, \ldots, x_N = 1$. Additionally, we will take a set of evenly spaced times $t_0 = 0, t_1 = h, t_2 = 2h, t_3 = 3h, \ldots$, where $h = 1/N$. We will write $u_i^{(j)}$ for the approximate value of u at point x_i and time t_j (ie $u_i^{(j)} \approx u(x_i, t_j)$).

Approximating both derivatives (similar to what we did in part 1), and doing some algebra, we can rewrite the heat equation as

$$\begin{aligned} u_i^{(j+1)} &= u_i^{(j)} + \frac{u_{i-1}^{(j)} - 2u_i^{(j)} + u_{i+1}^{(j)}}{1000h}, \\ u_i^{(0)} &= 0, \\ u_0^{(j)} &= 10, \\ u_N^{(j)} &= 10. \end{aligned}$$

This leads us to an iterative method for solving this problem: first, at $t = 0$, we set

$$u_i^{(0)} = \begin{cases} 10 & \text{if } i = 0 \text{ or } i = N, \\ 0 & \text{otherwise;} \end{cases}$$

then for all later values of time, we set

$$u_i^{(j+1)} = \begin{cases} 10 & \text{if } i = 0 \text{ or } i = N, \\ u_i^{(j)} + \dfrac{u_{i-1}^{(j)} - 2u_i^{(j)} + u_{i+1}^{(j)}}{1000h} & \text{otherwise.} \end{cases}$$

Implement this iterative scheme in Python. You should implement this as a function that takes N as an input.

Using a sensible value of N, **plot the temperature of the rod at $t = 1$, $t = 2$ and $t = 10$. Briefly (1-2 sentences) comment on how you picked a value for N.**

Use `numba.cuda` to parallelise your implementation on a GPU. You should think carefully about when data needs to be copied, and be careful not to copy data to/from the GPU when not needed.

Use your code to estimate the time at which the temperature of the midpoint of the rod first exceeds a temperature of 9.8. **Briefly (2-3 sentences) describe how you estimated this time**. You may choose to use a plot or diagram to aid your description, but it is not essential to include a plot.

By Timo Betcke & Matthew Scroggs