Lab Class 1

Math3101/5305

Term 2, 2023

In lectures, we discussed the two-point boundary-value problem

$$-u'' = f(x) \quad \text{for } 0 < x < L, \quad \text{with } u(0) = \gamma_0 \text{ and } u(L) = \gamma_L,$$

and how to use a finite difference method to compute an approximate solution

$$U_p \approx u(x_p) \quad \text{for } 0 \le p \le P, \quad \text{where } x_p = p\,\Delta x \text{ and } \Delta x = \frac{L}{P}.$$

Recall that U_1, U_2, …, U_{P-1} are found by solving a $(P-1)\times(P-1)$ linear system, e.g., when $P = 6$,

$$\frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} + \frac{1}{\Delta x^2}\begin{bmatrix} \gamma_0 \\ 0 \\ 0 \\ 0 \\ \gamma_L \end{bmatrix}.$$

1. Write a function

```
U, x = solve_bvp(L, f, gamma, P)
```

that computes the vectors $\boldsymbol{U} = [U_0, U_1, \ldots, U_P]^T$ and $\boldsymbol{x} = [x_0, x_1, \ldots, x_P]^T$. The argument `f` is a function, and `gamma` is a vector of length 2 holding the values of γ_0 and γ_L.

2. Consider the example

$$L = 2, \quad \gamma_0 = -1, \quad \gamma_L = 2.5, \quad f(x) = 5e^{-x},$$

in which case the exact solution is

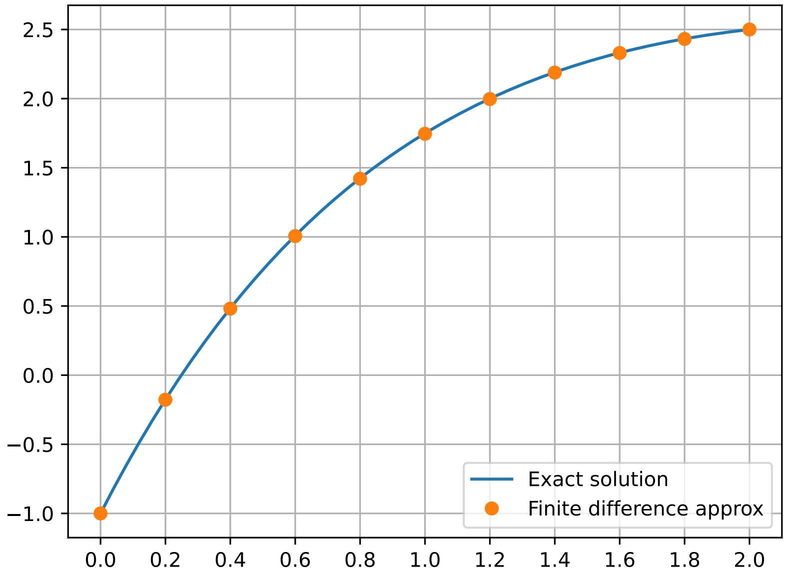
$$u(x) = A + Bx - 5e^{-x},$$

where

$$A = \gamma_0 + 5 \quad \text{and} \quad B = \frac{1}{L}(\gamma_L - A + 5e^{-L}).$$

Write a script that compares the exact solution and the numerical solution by plotting both on the same axes, like in Figure 1.

Figure 1: Comparison of the exact and numerical solutions.



3. Let $e_P = \max_{0\le p\le P} |U_p - u(x_p)|$ denote the maximum error. Write a second script to investigate the behaviour of e_P as we repeatedly double the value of P. Hence estimate the exponent r such that $e_P \approx Ch^r$, using the fact that $e_{P/2}/e_P \approx 2^r$ and thus $r \approx \log_2(e_{P/2}/e_P)$. Print your output as a table of the form

P	e_P	estimated r
8		
16		
32		
64		
128		

Python notes: Use `scipy.linalg.solve_banded` to solve the linear system.

Matlab notes: Use `spdiags` to construct the coefficient matrix and solve the linear system with the \ operator.

Julia notes: Use a `LinearAlgebra.SymTridiagonal` matrix and solve the linear system with the \ operator.