Project 6: Ordinary Differential Equations with Boundary Values

Hieu Bui

November 2, 2022

1 Introduction

Project 6 aims to numerically solve various types of ordinary differential equations (ODE) such as boundary value problems. The goal is to learn and exercise the overarching methodology, which includes discretizing the grid, discretizing the scheme, and solve. In other words, we make the domain of interest, transform the relevant governing equations from analytical to numerical, and solve for unknowns numerically.

2 Methods

2.1 Boundary Layer and Shock Waves

In this problem, we are given the boundary conditions, number of points, and governing equations. Therefore, we set up the grid first. Since the number of points required is 101, we can calculate the $\Delta x = \frac{\text{(end point - starting point)}}{101}$. The Δ will help with the scheme discretization process, where we have to use finite central difference. The given governing equation is $fu_x = \epsilon u_x x$. We replace u_{xx} with second order finite central difference, discretizing the second order differential; in addition, we do the same for the first order u_x.

$$\frac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2}$$

$$\frac{u_{i+1}-u_{i-1}}{2\Delta x}$$

$$f(\frac{u_{i+1}-u_{i-1}}{2\Delta x}) = \epsilon(\frac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2})$$

Since f and ϵ are constants, we perform some algebraic manipulations to combine like terms for their respective coefficients. We have three coefficients, allowing us to construct a tridiagonal matrix.

$$\vec{a} = \begin{pmatrix} -(\frac{f}{2\Delta x} + \frac{\epsilon}{\Delta x^2}) \\ -(\frac{f}{2\Delta x} + \frac{\epsilon}{\Delta x^2}) \\ . \\ . \\ . \\ 0 \end{pmatrix}$$

$$\vec{b} = \begin{pmatrix} 1 \\ \frac{2\epsilon}{\Delta x^2} \\ \frac{2\epsilon}{\Delta x^2} \\ . \\ . \\ 1 \end{pmatrix}$$

$$\vec{c} = \begin{pmatrix} 0 \\ (\frac{f}{2\Delta x} - \frac{\epsilon}{\Delta x^2}) \\ . \\ . \\ . \\ (\frac{f}{2\Delta x} - \frac{\epsilon}{\Delta x^2}) \end{pmatrix}$$

$$\vec{d} = \begin{pmatrix} 2 \\ 0 \\ 0 \\ . \\ . \\ -2 \end{pmatrix}$$

Array a is the lower diagonal; array b is the main diagonal; array c is the

top diagonal. The first and last element of b are 1 to enforce the boundary conditions, corresponding to the first and last element of array d. We can simultaneously solve for the tridiagonal system using Thomas Algorithm, which is the solver. However, this procedure only works for equations with linearity. For those of non-linear, we have to guess and iterate. Instead of known f, we guess the value, then go through the process of solving until residual meets the tolerance threshold.

2.2 Existence and Uniqueness

] This problem is similar to the boundary problem in the subsection above; however, the governing equation is different and non-linear.

$$u_{xx} = au$$

Again, we can discretize the equation (the scheme) as following:

$$\frac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2} = au$$
$$\frac{u_{i-1}-2u_i+u_{i+1}}{u\Delta x^2} = a$$

We have the coefficients with respect to u_{i-1}, u_i, and u_{i+1}. We will guess u and iterate. The $\vec{d}$ elements are equal to a. If the end result function does not behave according to linearity requirement, then it means that the solution is not unique such as a parabola.

3 Programming

3.1 Boundary Layer and Shock Waves

- We create the grid with linspace function: linspace(start point, end point, number of points)
- Calculate the coefficients from the discretizing process.
- Setup diagonal arrays with boundary conditions accounted for
- Solve the tridiagonal system using Thomas Algorithm

3.2 Existence and Uniqueness

- We create the grid with linspace function: linspace(start point, end point, number of points)
- Calculate the coefficients from the discretizing process with a guess value.
- Setup diagonal arrays with boundary conditions accounted for.
- Specify an arbitrary tolerance and correction values to start the while loop
- Solve the tridiagonal system using Thomas Algorithm until the tolerance is satisfied

4 Results

4.1 Boundary Layer and Shock Waves

The slope of cases with higher ϵ values are less steep.

4.2 Existence and Uniqueness

All cases exhibits linearity quality except case 3 when a = -1.

5 Discussion

Numerically solving for boundary value problems allow us to quickly approximate the answer without much concern about the governing equations format besides the boundary conditions themselves. It is a powerful tool that significantly reduce solving time with computers. For improvement, we could refactor the code to include the special case, where we have to guess the initial value, into the ordinary boundary conditions function.