CS450: Project Report

|  |  |
| --- | --- |
| **Haroun Habeeb** | **hhabeeb2@illinois.edu** |

# I am using the second edition of the course textbook. I opted for extra homework. I opted to do problems exercise 11.1, exercise 11.7, computer problem 11.4, computer problem 11.12, computer problem 11.13.

For each question I have written down the problem (roughly).

All implementations were done in python. Numpy and scipy were used extensively.

The aim of the homework was to familiarize me with PDEs and solving them.

# Exercise 11.1

Suppose you’re given a general-purpose subroutine for solving IVPs for system of n first order-ODEs and it is the only available subroutine. For each type of problem in parts (a)-(c), answer each of the following questions:

1. What is the function ?
2. How would you obtain initial conditions?
3. What special properties could be used to decide on a solver?

### (a) To compute

If we use , then the required integral is . Hence,

The initial value can be evaluated at . since the upper and lower limits of the integral in are equal.

Any solver that can solve can solve our ODE. However, unless we know , we cannot discuss special properties of the ODE.

### (b) To solve given

We need to reduce a higher order differential equation to a first order differential equation.

To do so, we’d define two variables – say such that

Where and essentially. Notice that the line above is a linear ODE and we have a subroutine for that.

Since , we know and .However, is trickier. We know that and . We would need to guess values for until the system becomes satisfiable. This is essentially the shooting method of solving BVPs.

As such, it’s a normal multi-dimensional IVP with coupled variables. There are no other special properties that would help an ODE solver. The jacobian is a simple which is well conditioned if is reasonable.

Alternatively, we can also discretize along and solve using the colocation method. We could also use a set of basis function – the choice of basis functions would produce different kinds of solutions.

### (c) Solve heat equation given ,

The heat equation is a parabolic 2nd order PDE.

Since we need to essentially remove one variable to make it an ODE, we could discretize either along or along . Lets say that we discretize along . Hence, we have an ODE at each where for . Let

However, we can no longer differentiate along . We must instead use a finite difference method.

We replace by , or any other equivalent for .

Hence the system of ODEs is:

Hence, the function is (t)

Where is a triadiagonal matrix with , .

The boundary conditions become and

The jacobian of the system is ill conditioned because of the factor. Since the jacobian is ill conditioned, our ODE system solver would have to be capable of handling such systems.

# Exercise 11.7

Prove that the SOR method diverges if doesn’t lie in

The successive over relaxation method is used to iteratively solve linear systems such as . From the book, we know that the iteration is

Following the notation in the text book, and

Let .

Therefore, the equation above becomes

The iterations converge if

However, I believe it is unclear how to solve something like this. An alternative method is to say that the spectral radius of is less than 1. This is a statement made in the book. Note that I defined in the line above.

What we need is a bound on in terms of . One way to do this is to say:

Where is the dimensionality of .

We know that the determinant of a triangular matrix is just the product of the diagonal entries.

Hence,

Using this, we see that .

If , then and hence which means that the algorithm does NOT converge, or equivalently, diverges.

# Computer Problem 11.4

Use method of lines and any ODE solve of your choice to solve

initial conditions:

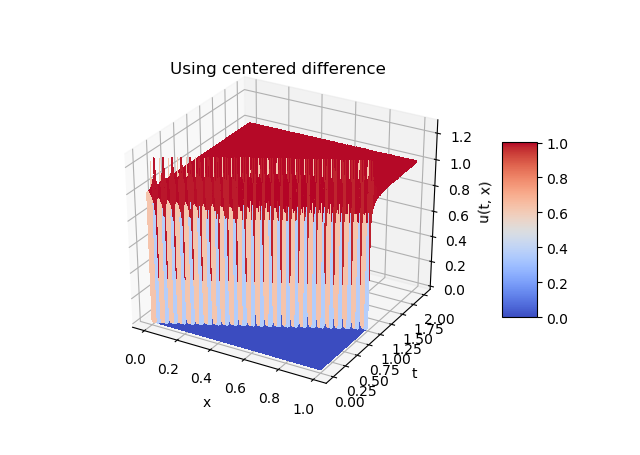
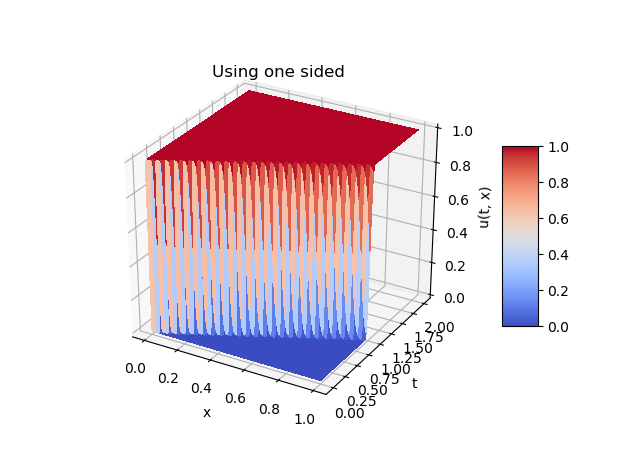
boundary conditions:

Over , plot vs (3D surface plot). Try both one sided and centered finite difference for spatial discretization. The actual solution is a step function of height 1 moving toward positive with velocity 1.

Does either scheme get close? Describe the difference between computed solutions. Which solution is smoother? Which is more accurate?

### **Answer**

Both schemes get close if we’re careful about the derivatives at the boundary conditions.





Since centered difference is non-zero at two time steps for each point, it looks less smooth than the one using a one-sided derivative.

This result is different from the general expectation because the derivatives are 0 for most of the time. Hence, only the points at at which derivative is non-zero is relevant to us. In this case, where the actual (analytical) derivative is a dirac delta, the one-sided derivative works better.

Code was implement in python. The associated file is cp.11.4.hhabeeb2.py. It can be run simply as python cp.11.4.hhabeeb2.py [nx=1000, nt=1000].. It uses scipy’s odeint function to solve a system of linear ODEs.

# Computer Problem 11.12

Implement steepest descent and conjugate gradient methods for solving symmetric positive definite linear systems. Compare their performance, both in rate of convergence and in total time. Use a representative sample of test problems – well conditioned and ill-conditioned too.

How does the rate of convergence compare with theoretical convergence rate?

# Computer Problem 11.13