PDE II PROJECT 1

THOMAS ARMSTRONG

1. Question 1

1.1. **Approach to Problem.**

We want to find the solution $u(x,t)$ that solves the initial boundary-value problem:

$$u_t = u_{xx} + 25x(1-x^2), \quad 0 \le x \le 1, \quad 0 \le t \le 0.3$$
$$u(0,t) = 0,\ u(1,t) = 0.9 \quad \forall t, \quad u(x,0) = 0.9\,\sin\left(\frac{\pi x}{2}\right)$$

To approximate this solution, we will split the solution into two parts, one which satisfies the boundary conditions and one which satisfies zero boundary conditions. Let $v(x,t)$ be a solution with zero boundary conditions and $g(x,t)$ be the solution satisfying the boundary condition. That is:

$$u(x,t) = v(x,t) + g(x,t)$$

Where $v(x,t)$ satisfies the inhomogenous initial boundary-value problem with zero boundary conditions, this will be solved by the Crank-Nicolson method as later described. Now, $v(x,t)$ is a solution to the following problem:

$$v_t = v_{xx} + \tilde{f}(x,t), \quad 0 \le x \le 1, \quad 0 \le t \le 0.3$$
$$v(0,t) = 0,\ v(1,t) = 0 \quad \forall t, \quad v(x,0) = v_0(x)$$

Using the lecture notes, we can define $g(x,t)$ as:

$$g(x,t) = \mu_1(t) + \frac{(\mu_2(t) - \mu_1)}{L}\,x$$

where $u(0,t) = \mu_1(t) = 0$, $u(1,t) = \mu_2(t) = 0.9$ and $L = 1$. We therefore obtain:

$$g(x,t) = g(x) = 0.9\,x$$

We can also find $\tilde{f}(x,t)$ and $v_0(x)$ using the formulae in the lecture notes:

$$\begin{aligned}\tilde{f}(x,t) &= f(x,t) - \frac{\partial g}{\partial t}(x,t) + K\frac{\partial^2 g}{\partial x^2}(x,t)\\ &= 25x(1-x^2)\end{aligned}$$

and

$$\begin{aligned}v(x,0) &= u(x,0) - g(x,0)\\ &= 0.9\sin\left(\frac{\pi x}{2}\right) - 0.9x \qquad (1)\end{aligned}$$

1.1.1. *Crank-Nicolson.*

Using the lecture notes to obtain the Crank-Nicolson method for the inhomogeneous heat equation (with zero boundary conditions), we have the approximation for $v(x,t)$ is:

$$\frac{w_{i,j+1} - w_{i,j}}{\tau} - K\frac{(w_{k+1,j+1} - 2w_{k,j+1} + w_{k-1,j+1} + w_{k+1,j} - 2w_{k,j} + w_{k-1,j})}{2h^2} = f(x_k, t_j + \tau/2)$$

In our problem, $K = 1$ and $f(x,t) = 25x(1-x^2)$. We also replace the right-hand side of the equation by:

$$\frac{1}{2}\left(f(x_k,t_j) + f(x_k,t_{j+1})\right) = f(x_k)$$

We have been able to simply this since our f has no time dependence and therefore $f(x_k,t_{j+1}) = f(x_k,t_j)$. We also Taylor expanded the term $f(x_k, t_j + \tau/2)$ prior to simplifying it. We can now re-write the equation so that we can use the double-sweep method.

Date: February 2019.

1.1.2. *Double-Sweep Method.* We can re-write the general double-sweep formula from the lecture notes to our specific problem.

$$\underbrace{\frac{\gamma}{2} w_{k-1,j+1}}_{A_i} - \underbrace{(1+\gamma)}_{C_i} w_{k,j+1} + \underbrace{\frac{\gamma}{2}}_{B_i} w_{k+1,j+1} = \underbrace{-(1-\gamma) w_{k,j} - \frac{\gamma}{2} w_{k+1,j} - \frac{\gamma}{2} w_{k-1,j} - \tau f(x_k, t_j)}_{F_i}$$

Now we will be able to find the α_is and β_is required.

$$\alpha_{i+1} = \frac{\frac{\gamma}{2}}{1 + \frac{\gamma}{2}(2 - \alpha_i)}$$

$$\beta_{i+1} = \frac{\frac{\gamma}{2}\beta_i - F_i}{1 + \frac{\gamma}{2}(2 - \alpha_i)}$$

1.2. **MATLAB solution.** My solution to this problem is based on the "crank_nicol_heat.m" solution to Exercise 2 of the 4th practical session. The following code is typed into the Command Window before my function file is called:

```
>> x = (0:30)/30;
>> T = 0.3;
>> M = 30;
>> u0 = 0.9*sin(pi*x/2);
```

The `x` variable represents the 31 grid points from 0 to 1. The `T` term denotes the end point of our time grid, as specified in the question, whilst `M`+1 is how many grid points we want to consider for t. I chose this value of M so that we will be able to slice our solution at times $t = 0,\ 0.1,$ and 0.3 which is required in the next question. The last line of code specifies the initial condition of the problem through `u0`. We then call the function "`project_c_n.m`" which returns the surface plot of our approximation of the solution to $u(x,t)$ as well as the matrix used to produce this plot; this function is called using the following command in the Command Window:

```
>> u = project_c_n(T,M,x,u0);
```

1.2.1. *The Function File.* In the function file `project_c_n.m`, we split the problem into two sections, first we approximate the solution of $v(x,t)$ using the Crank-Nicolson and double-sweep method, then we find $g(x,t)$ by evaluating the function on the grid `x`. At the end of the function file, we then add the two solutions together to obtain $u(x,t)$ and return the surface plot. First, the function calculates the step size in time and space, it then creates a grid for t based on the `T` and `M` inputs. It then calculates the initial condition of $v(x,t)$ from the initial condition `u0`, using the equation (1).

To perform the double sweep method, we first note that α_1 and β_1 are 0 since $v(x,t)$ is 0 at the boundary. We then implement nested loops as seen in the function file to calculate all the α_i and β_i coefficients before performing the backwards sweep to calculate the entries of the matrix `w`.

Finally, we calculate $g(x,t)$ at every spacial step. Since $g(x,t)$ does not depend on t, we obtain a matrix which consists of a vector, repeated at every time step. This is then added to `w`, our approximation for $v(x,t)$, to obtain $u(x,t)$. The `surf` function is then called after transposing `u`, to obtain the required surface plot. Note that in our function `u` is returned so that we can answer the next question.

2. Question 2

To plot $u(x,t)$ against x at different time intervals, we use the matrix `u` that was returned when we called the function file in the previous section. By slicing the matrix at points where $t = 0, 0.1,$ and 0.3 separately, we obtain the values of `u` over the range $x \in [0,1]$ as required. We then call a separate file named `plot_u_on_single_figure.m` which takes the inputs `x` and `u` (the output of the `project_c_n.m` function) and returns the plot required. In the command window, this is called by:

```
>> plot_u_on_single_figure(x,u);
```

where `x` is the grid we defined at the start. The file `plot_u_on_single_figure.m` uses the `plot` and `hold on` commands to produce multiple plots in a single figure.