COSC2500 Assignment 5

Ryan White
s4499039

2nd of November 2020

R5.1

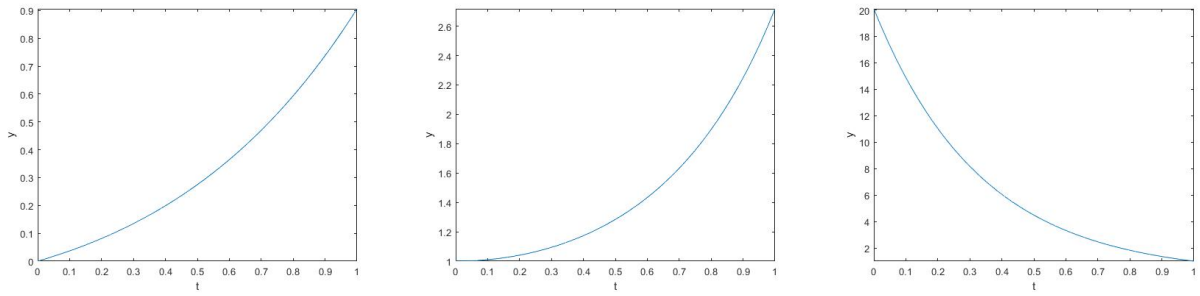
Consider the 3 linear DEs from Sauer,

$$y'' = y + \frac{2}{3}e^t \qquad y(0) = 0,\ y(1) = \frac{1}{3}e \tag{1}$$

$$y'' = (2 + 4t^2)y \qquad y(0) = 1,\ y(1) = e \tag{2}$$

$$y'' = 3y - 2y' \qquad y(0) = e^3,\ y(1) = 1 \tag{3}$$

Using the shooting method, these linear DEs were solved as follows using the code from Appendix 0.1:

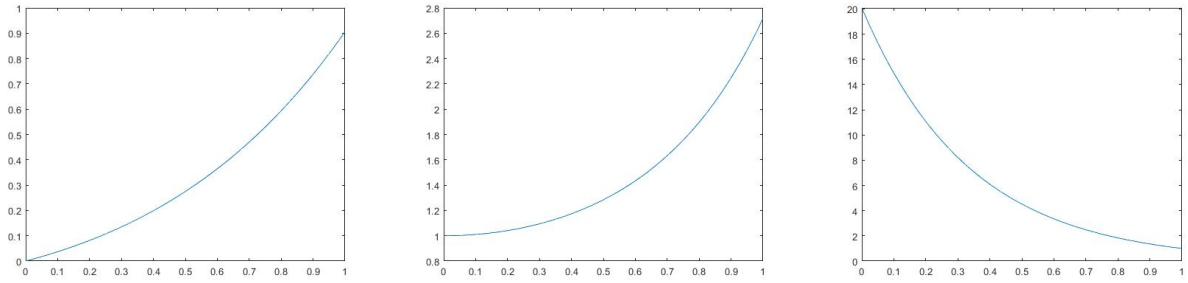


(a) Solved DE for Equation 1 (b) Solved DE for Equation 2 (c) Solved DE for Equation 3

Figure 1: Solved BVPs for the given linear DEs

R5.2

Solutions for the linear DEs from part 5.1 were also solved using the finite difference method, shown in Appendix 0.2, and shown below:



(a) Solved DE for Equation 1 (b) Solved DE for Equation 2 (c) Solved DE for Equation 3

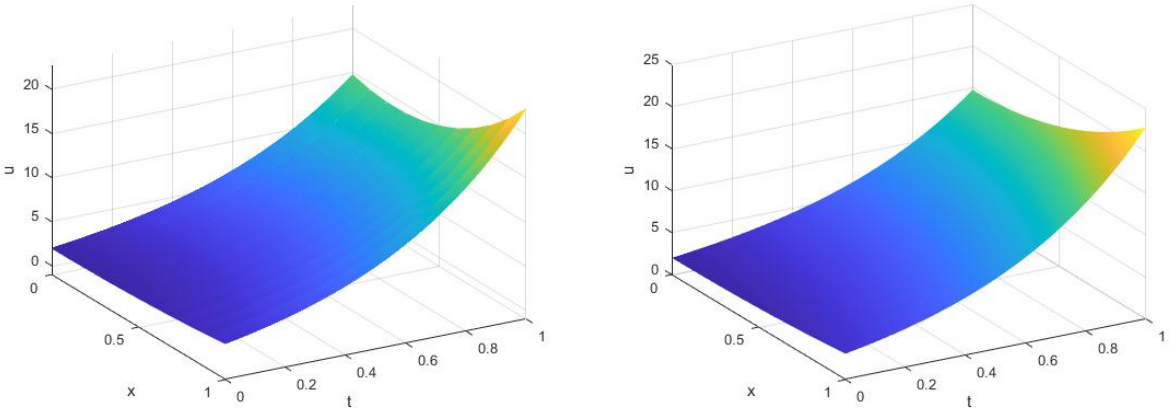
Figure 2: Solved BVPs for the given linear DEs

R5.3

a. i. Given the boundary conditions

$$\begin{aligned} u(x,0) &= 2\cosh x \qquad \text{for} \quad 0 \le x \le 1 \\ u(0,t) &= 2e^{2t} \qquad \text{for} \quad 0 \le t \le 1 \\ u(1,t) &= (e^2+1) \times e^{2t-1} \qquad \text{for} \quad 0 \le t \le 1 \end{aligned}$$

The calculated solution, and exact solution are shown in Figure 3, plotted using the code from Appendix 0.3, and the exact solution $u(x,t) = e^{2t+x} + e^{2t-x}$:



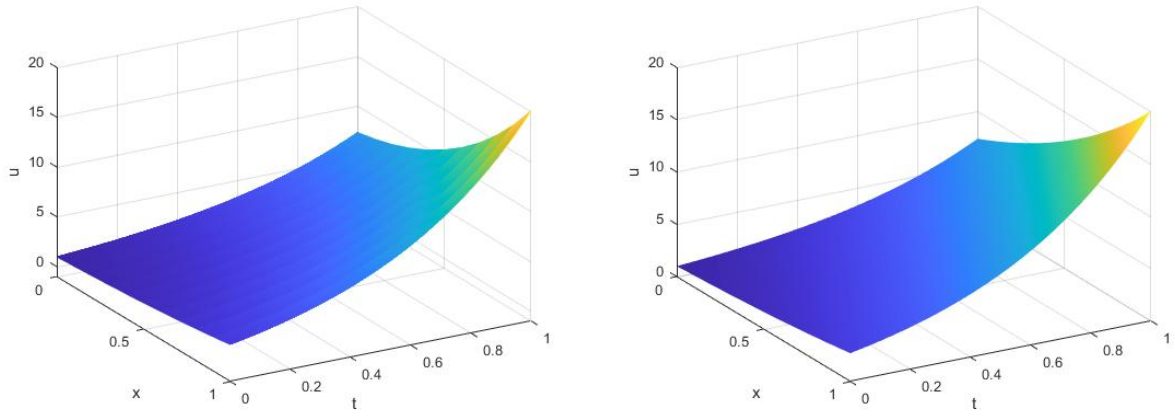
(a) Approximation of DE i. with $h = 0.1$, $k = 0.002$ (b) Exact Solution for DE i.

Figure 3: Solved BVPs for the given linear DEs

ii. Given the boundary conditions

$$\begin{aligned} u(x,0) &= e^{x} \qquad \text{for} \quad 0 \le x \le 1 \\ u(0,t) &= e^{2t} \qquad \text{for} \quad 0 \le t \le 1 \\ u(1,t) &= e^{2t+1} \qquad \text{for} \quad 0 \le t \le 1 \end{aligned}$$

The calculated and exact solutions are shown in Figure 4, with the exact solution $u(x,t) = e^{2t+x}$:



(a) Approximation of DE ii. with $h = 0.1$, $k = 0.002$

(b) Exact Solution for DE ii.

Figure 4: Solved BVPs for the given linear DEs

In both of the cases i. and ii., the approximate solution appears reasonably accurate, with only a more defined curve as a function of x value being immediately noticeable. As the boundary conditions for x are clearly defined, the approximate solution follows the exact solution (within accuracy bounds) along the x boundaries.
Also in both cases i. and ii., the model broke down for time steps of about $k \geq 0.01$, compared with the (working) $k = 0.002$. An example of this is shown in Figure 5, with the extremities being in excess of 25 orders of magnitude.

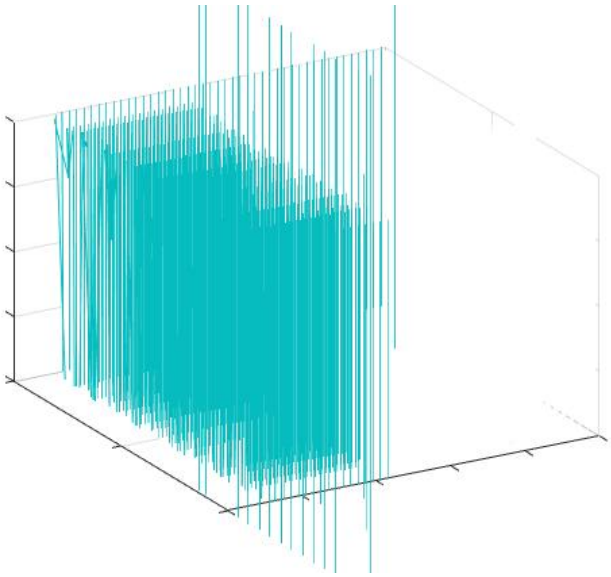


Figure 5: Model Breakage for Large Time Steps ($k = 0.01$)

R5.4

The paper analysed was "*A PDE Perspective On Climate Modeling*", authored by Broome, S, and Ridenour, J. It had aimed to create a model of atmospheric conditions across some 1 dimensional vector at the tropopause, given initial and boundary conditions across a number of discrete intervals of the total length. The authors first began with altering the Navier-Stokes equations to simplify them down to a set of Euler Equations, which was then linearised to form a Hyperbolic PDE. Then, using the Kronecker product, they had eventually formulated a method that reduced each PDE to a system of ODEs which were then approximated iteratively over the range of time steps.

The paper consisted of an extremely in-depth derivation of the model, including appropriate mathematical definitions and proofs of relevant deductions. Consequently, it had seemed out-of-scope of undergraduate students, perhaps except for those at the highest levels. That said, the logical progression of the derivation seemed to be consistent and was clear in intent.

Unfortunately, the simulation was not reproducible (as no data was given), even though the matlab code for one of the method's formulae was given. Results of the simulation, including graphs of respective changes in atmospheric conditions were given, however the bare minimum of discussion of these results was given, and so the model could not be verified beyond anecdote.

In the conclusion section of the article, the authors mentioned that the implications therein were *"considered visionary in the field at this time,"* and so, given their professional opinion, scholars in the field of climatology and atmospheric sciences would undeniably be interested in the models presented. The authors make clear the practical applications of their model, but fail to propose further developments to the existing simulation beyond extension into higher dimensions.

Appendices

0.1 Matlab Code for R5.1

```
clear all
func1 = @(t, y) [y(2); y(1) + 2/3 * exp(t)];
func2 = @(t, y) [y(2); (2 + 4 * t^2) * y(1)];
func3 = @(t, y) [y(2); 3 * y(1) - 2 * y(2)];
conds1 = [0, (1/3) * exp(1)];
conds2 = [1, exp(1)];
conds3 = [exp(1)^3, 1];

temp = @(s) bc_mismatch(s, func3, [0, 1], conds3);

sstar = fzero(temp, [-100, 100]);

sol = ode45(func3, [0, 1], [conds3(1), sstar]);
fplot(@(x)deval(sol, x, 1), [0, 1]);
xlabel('t'); ylabel('y');

function final_error = bc_mismatch(x, f, interval, conds)
% Find the roots of this for R5 .1 shooting method
% Solve the IVP

[t, y] = ode45(f, interval, [conds(1) x]);

% Compare with final BC
final_error = y(end, 1) - conds(2);
end
```

0.2 Matlab Code for R5.2

```
%func1pts = (1/h^2, -1 - 2/h^2, 1/h^2, 2/3 * exp(tn(n)), 1, exp(1)/3);
%func2pts = (1/h^2, -2 - 2/h^2 - 4 * tn(n)^2, 1/h^2, 0, 1, exp(1));
%func3pts = (1/h^2 + 2/h, -2/h^2 - 3 - 2/h, 1/h^2, 0, exp(3), 1);

% How many points do we want ?
N = 100;
% Pre - allocate memory for matrix and vector
A = zeros(N, N);
c = zeros(N, 1);
tn = linspace(0, 1, N); % Need to change for 2(a), since it goes to 1.5
h = tn(2) - tn(1);
% Boundary conditions
A(1, 1) = 1;
c(1) = 1; % First BC
A(N, N) = 1;
c(N) = exp(1); % End BC
% Make the matrix
for n = 2:(N-1)
A(n, n-1) = 1/h^2;
A(n, n) = -2 - 2/h^2 - 4 * tn(n)^2;
A(n, n+1) = 1/h^2;
```

```
c(n) = 0;
end
Y = A\c;
plot(tn, Y)
```

0.3 Matlab Code for R5.3

```
w = heatfd(0, 1, 0, 1, 10, 500);    %comment out for exact solution
% [X, T] = meshgrid(0:0.002:1);     %remove comment for exact solution
% U = exp(2.*T + X) + exp(2.*T - X);
% surf(X, T, U, 'EdgeColor', 'none')
% view(60, 30);
xlabel('x'); ylabel('t'); zlabel('u');



function w = heatfd(xl, xr, yb, yt, M, N)
%Program 8.1 Forward difference method for heat equation
%input: space interval [xl, xr], time interval [yb, yt],
%      number of space steps M, number of time steps N
%output: solution w
%Example usage: w = heatfd(0, 1, 0, 1, 10, 250)
    f = @(x) 2 * cosh(x);
    l = @(t) 2 * exp(2 * t);
    r = @(t) (exp(2) + 1) * exp(2 * t - 1);
    D = 1; % diffusion coefficient
    h = (xr - xl)/M; k = (yt - yb)/N; m = M - 1; n = N;
    sigma = D * k/(h * h);
    a = diag(1 - 2 * sigma * ones(m, 1)) + diag(sigma * ones(m-1, 1), 1);
    a = a + diag(sigma * ones(m-1, 1), -1); % define matrix a
    lside = l(yb + (0:n)*k); rside = r(yb + (0:n) * k);
    w(:,1) = f(xl + (1:m) * h)'; % initial conditions
    for j=1:n
        w(:, j+1) = a * w(:, j) + sigma * [lside(j); zeros(m - 2, 1); rside(j)];
    end
    w = [lside; w; rside]; % attach boundary conds
    x = (0:m + 1) * h; t = (0:n)*k;
    mesh(x, t, w') % 3-D plot of solution w
    view(60, 30); axis([xl xr yb yt -1 1])
end
```