1 Introduction

Consider the linear convection-diffusion equation

$$\frac{\partial T}{\partial t} + v\frac{\partial T}{\partial y} = \alpha\frac{\partial^2 T}{\partial y^2}\,, \qquad v = \sin(\pi y)\,, \qquad \alpha = \frac{1}{\text{Pr Re}}\,, \tag{1}$$

subject to the initial conditions

$$\begin{aligned} &\text{(a)} \quad T(y, t=0) = \cos(2\pi y)\sin(\pi y) \\ &\text{(b)} \quad T(y, t=0) = \cos(2\pi y)\,, \end{aligned} \tag{2}$$

and parameters

$$\begin{aligned} \text{Re} &= 1 && \text{(Reynolds number, molten glass)} \\ \text{Pr} &= 25 && \text{(Prandtl number, molten glass)} \\ \Delta t &= 0.001 && \text{(Time step)} \\ L_y &= 2 && \text{(Domain } y\text{-length)} \\ N &= 2^n + 1 && \text{(Number of } y\text{-points, where } n = 5, 6)\,. \end{aligned} \tag{3}$$

1.1 Problem 1

Use the Fourier pseudo-spectral method to numerically integrate (1) with the given parameters. Use the Euler explicit method for time advancement. Higher resolution with $n = 6$ will improve the accuracy of the method for the initial condition (a). Plot T as a function of time t at $t = \{0.2, 2, 5, 10\}$.

1.2 Problem 2

Use the FTCS Euler explicit method with second-order finite differences for the same computation, and compare results to the Fourier pseudo-spectral method using the same mesh resolution.

2 Methodology

2.1 Problem 1

We re-write (1) as

$$\frac{\partial T}{\partial t} = \alpha\frac{\partial^2 T}{\partial y^2} - v\frac{\partial T}{\partial y}\,. \tag{4}$$

The first and second spatial derivatives of T are calculated by taking the Fourier transform of T, multiplying the Fourier coefficients by ik_n and $-k_n^2$, respectively, and then taking the inverse Fourier transform. With values of $\partial T/\partial t$ known at all grid points now, values of T at the next time step are calculated using the explicit Euler method,

$$T^{n+1} = T^n + \Delta t\frac{\partial T}{\partial t}\,. \tag{5}$$

2.2 Problem 2

To implement the FTCS explicit method, we approximate (1) using forward-differences in time and central-differences in space as

$$\frac{T_i^{n+1}-T_i^n}{\Delta t}+\nu\frac{T_{i+1}^n-T_{i-1}^n}{2\Delta y}=\alpha\frac{T_{i+1}^n-2T_i^n+T_{i-1}^n}{\Delta y^2}\,, \tag{6}$$

which can be solved for T_i^{n+1} as

$$T_i^{n+1}=T_i^n+\Delta t\left(\alpha\frac{T_{i+1}^n-2T_i^n+T_{i-1}^n}{\Delta y^2}-\nu\frac{T_{i+1}^n-T_{i-1}^n}{2\Delta y}\right)\,. \tag{7}$$

This constitutes an explicit equation for T^{n+1}, and thus we can step forward in time without the need to solve any matrix equations. We enforce periodic boundary conditions by wrapping T_{i-1} to T_N when $i=1$ and vice versa.

3 Results

4 Discussion

The FTCS method for initial condition (b) has difficulty capturing the diffusive behavior accurately; the limiting Fourier value is $T\sim 0.765$ for both $n=\{5,6\}$, whereas the FTCS method's limiting values are $T\sim\{0.806,0.788\}$ respectively.

5 References

No external references were used other than the course notes for this assignment.

Appendix: MATLAB Code

The following code listings generate all figures presented in this homework assignment. The Fourier codes `find_dfdn.m` and `find_d2fdn2.m` are provided by Prof. Biringen.

Listing 1: `Problem_1.m`

```
function [T_history] = Problem_1(varargin)

    %%%%%%
    % Solves the linear convection-diffusion equation using the Fourier pseudo-spectral
    % method.
    %
    % Ryan Skinner, November 2015
    %%%

    Set_Default_Plot_Properties();

    switch length(varargin)
        case 0
            n_plot = 41;
        case 1
            n_plot = varargin{1};
        otherwise
            error('Too many arguments passed to Problem_1');
    end

    cases = {{'a',5},{'a',6},{'b',5},{'b',6}};
    T_history = cell(length(cases),1);
```

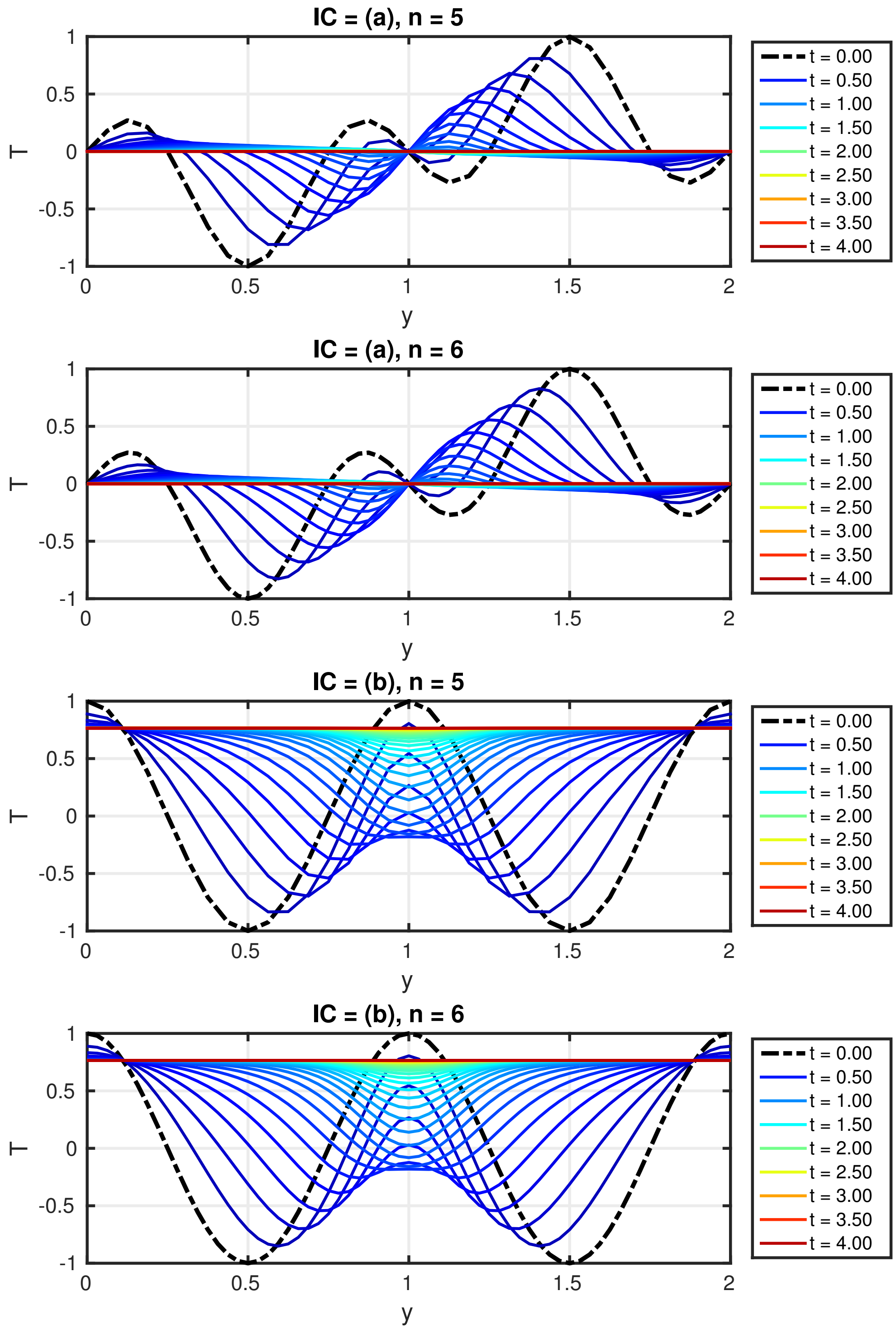

Figure 1: Solutions to the linear convection-diffusion equation using the Fourier pseudo-spectral method for different initial conditions and mesh resolution parameters. Most interesting behavior occurs when $t < 4$. Trends can be extrapolated to future times $t = 5, 10$, which are not shown. Initial condition displayed as black dot-dashed line (·–·–).

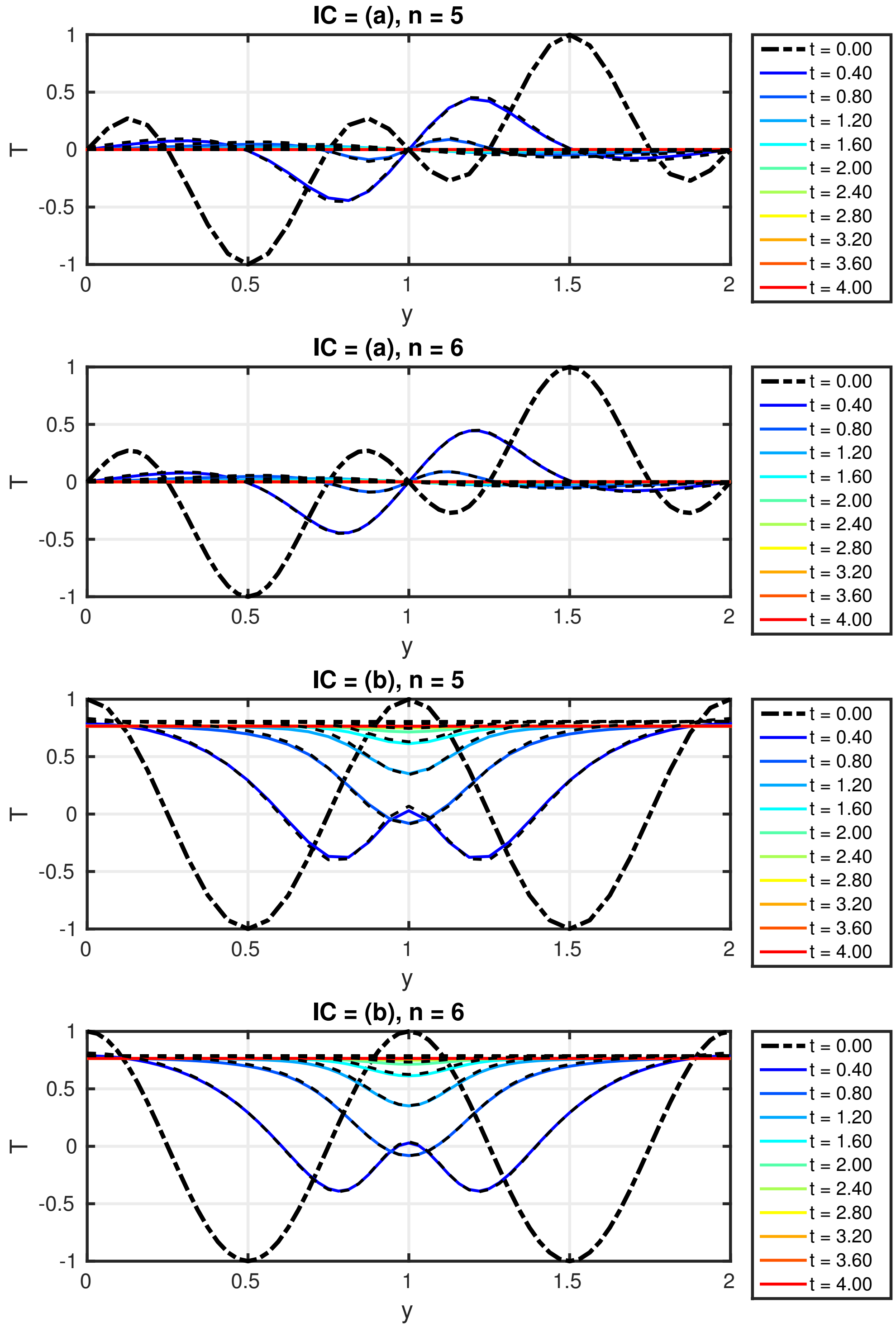

Figure 2: Solution comparison between the Fourier pseudo-spectral method (colored lines) and the FTCS method (– – –) for $t < 4$. Initial condition displayed as black dot-dashed line (· – · –). Agreement between the Fourier and FTCS methods is decent. The limiting Fourier value is $T \sim 0.765$ for both $n = \{5, 6\}$, whereas the FTCS method's limiting values are $T \sim \{0.806, 0.788\}$ respectively. Further analysis is deferred to Figure 3.

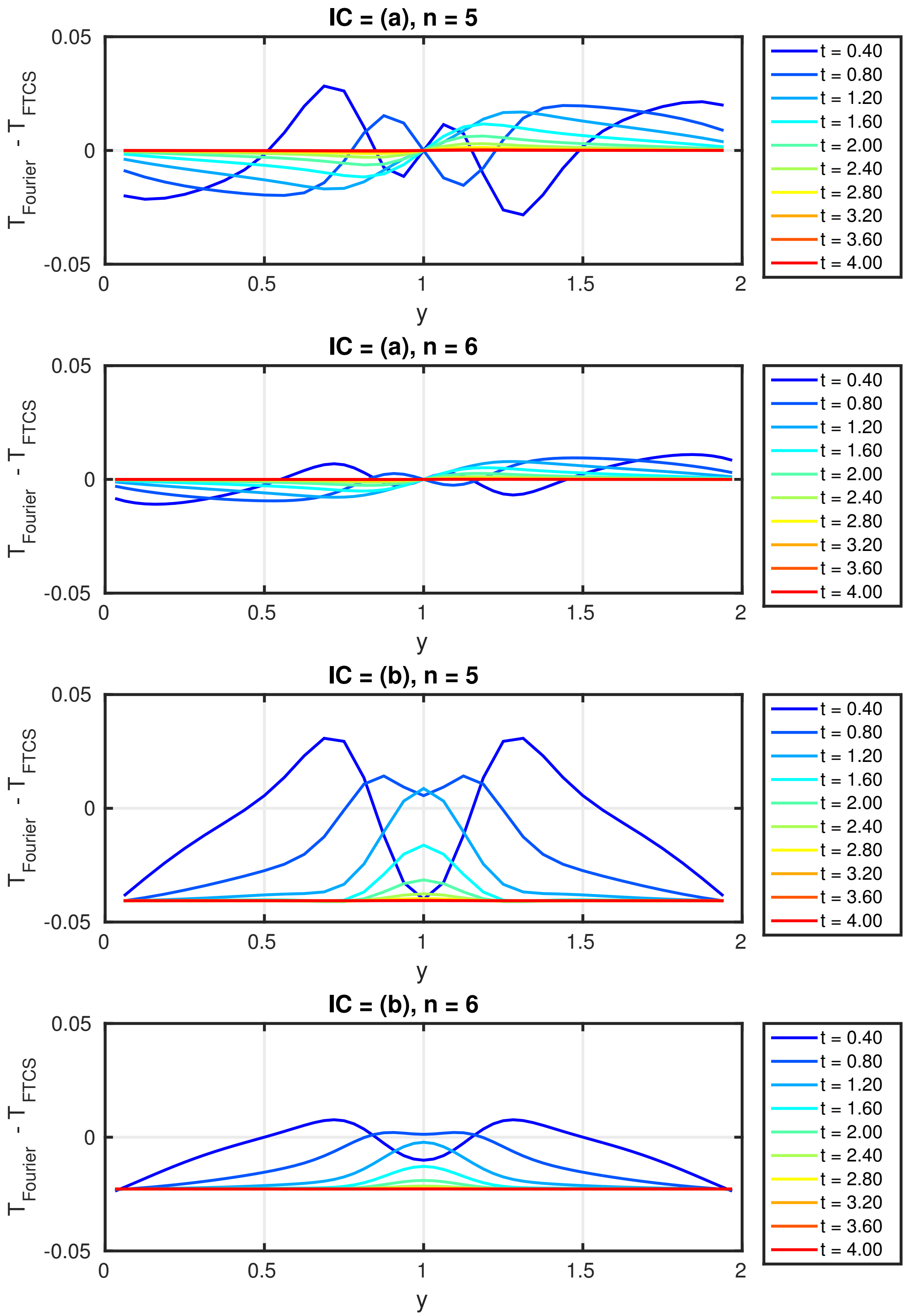

Figure 3: Difference between the Fourier pseudo-spectral and FTCS solutions at evenly-spaced discrete times $t < 4$, omitting the initial condition. Different initial conditions and spatial resolutions produce different behavior.

```
    for case_i = 1:length(cases)

    IC_str = cases{case_i}{1};
         n = cases{case_i}{2};

    % Spatial domain.
    nn = 2^n + 1;
    Ly = 2.0;
    y = linspace(0, Ly, nn)';

    % Temporal domain.
    dt = 0.001;
    if strcmp(IC_str,'a')
        t_final = 4;
    else
        t_final = 4;
    end
    t = [0:dt:t_final];

    % Physical parameters.
    Re = 1;
    Pr = 25;
    alpha = 1 / (Re * Pr);
    v = sin(pi*y);

    % Initialize solution.
    T = nan(nn,length(t));
    if strcmp(IC_str,'a')
        T(:,1) = cos(2*pi*y) .* sin(pi*y);
    else
        T(:,1) = cos(2*pi*y);
    end

    %%%
    % Solve problem numerically.
    %%%

    for t_n = 1:(length(t)-1)

        Tn = T(:,t_n);

        dTdy   = find_dfdn(  Tn',nn,Ly)';
        d2Tdy2 = find_d2fdn2(Tn',nn,Ly)';

        % Update solution.
        T(:,t_n+1) = Tn + dt * (alpha * d2Tdy2 - v .* dTdy);

    end

    %%%
    % Process results.
    %%%

    cmap = jet(n_plot);
    step_numbers = round(linspace(1,length(t),n_plot));
    hf = figure(case_i);
    set(hf,'Position',[100,500,900,300]);
    hold on;
    plot_handles = [];
    for t_n = 1:length(step_numbers)
        tmp = sprintf('t = %.2f', t(step_numbers(t_n)));
        if t_n == 1
            hp = plot(y, T(:,step_numbers(t_n)), 'k-.', 'LineWidth', 3, 'DisplayName', tmp);
        else
            hp = plot(y, T(:,step_numbers(t_n)), 'DisplayName', tmp, 'Color', cmap(t_n,:));
        end
        if(mod(t_n-1,5) == 0 || n_plot <= 11)
            plot_handles(end+1) = hp;
        end
```

```
    end
    title(sprintf('IC = (%s), n = %.0f',IC_str,n));
    xlabel('y');
    ylabel('T');
    ylim([-1,1]);
    xlim([0,Ly]);
    hleg = legend(plot_handles);
    set(hleg,'Location','eastoutside');

    T_history{case_i} = T;

    end

    disp('Done.');

end
```

Listing 2: `find_dfdn.m`

```
function dfdn = find_dfdn(f,nn,L)
% ASEN 5327 Class Notes Fall 2010
N=nn-1;
n=[0:(N/2)-1 -(N/2):-1];
%L=1; %show fuction is periodic on wavelength of L
kn=2*pi*n./L;
kn((N/2)+1)=0;

%find Fourier components
f_tilde=fft(f,N);
dfdn=ifft(i*kn.*f_tilde,N);
dfdn(nn)=dfdn(1);

end
```

Listing 3: `find_d2fdn2.m`

```
function  d2fdn2 = find_d2fdn2(f,nn,L)
%ASEN5327 Class notes Fall 2010
N=nn-1;
n=[0:(N/2)-1 -(N/2):-1];
kn=2*pi*n./L;
kn((N/2)+1)=0;
f_tilde=fft(f,N);
d2fdn2=ifft(-(kn.^2).*f_tilde,N);
d2fdn2(nn)=d2fdn2(1);
end
```

Listing 4: `Problem_2.m`

```
function [] = Problem_2()

    %%%%%%
    % Solves the linear convection-diffusion equation using the FTCS explicit method.
    %
    % Ryan Skinner, November 2015
    %%%

    Set_Default_Plot_Properties();

    % Display results from Problem 1 at only a few time steps.
    n_plot = 11;
    T_prob1 = Problem_1(n_plot);

    cases = {{'a',5},{'a',6},{'b',5},{'b',6}};
    for case_i = 1:4

    IC_str = cases{case_i}{1};
```

```
         n = cases{case_i}{2};

    % Spatial domain.
    nn = 2^n + 1;
    Ly = 2.0;
    y = linspace(0, Ly, nn)';
    dy = y(2) - y(1);

    % Temporal domain.
    dt = 0.001;
    if strcmp(IC_str,'a')
        t_final = 4;
    else
        t_final = 4;
    end
    t = [0:dt:t_final];

    % Physical parameters.
    Re = 1;
    Pr = 25;
    alpha = 1 / (Re * Pr);
    v = sin(pi*y);

    % Initialize solution.
    T = nan(nn,length(t));
    if strcmp(IC_str,'a')
        T(:,1) = cos(2*pi*y) .* sin(pi*y);
    else
        T(:,1) = cos(2*pi*y);
    end

    %%%
    % Solve problem numerically.
    %%%

    for t_n = 1:(length(t)-1)

        Tn = T(:,t_n);

        for i = 1:nn
            if i == 1 ; Tnim1=Tn(nn); else Tnim1=Tn(i-1); end % Periodic BCs
            if i == nn; Tnip1=Tn( 1); else Tnip1=Tn(i+1); end % Periodic BCs
            T(i,t_n+1) = Tn(i) + dt * ( alpha * (Tnip1 - 2*Tn(i) + Tnim1) / (dy^2) ...
                                       - v(i) * (Tnip1           - Tnim1) / (2*dy) );
        end

    end

    %%%
    % Process results.
    %%%

    step_numbers = round(linspace(1,length(t),n_plot));
    figure(case_i);
    hold on;
    for t_n = 2:length(step_numbers)
        plot(y, T(:,step_numbers(t_n)), 'k--');
    end

    cmap = jet(n_plot);
    hf = figure(length(cases) + case_i);
    set(hf,'Position',[100,500,900,300]);
    hold on;
    plot_handles = [];
    for t_n = 2:length(step_numbers)
        tmp = sprintf('t = %.2f', t(step_numbers(t_n)));
        method_err = (T_prob1{case_i}(:,step_numbers(t_n)) - T(:,step_numbers(t_n)));
        hp = plot(y(2:end-1), method_err(2:end-1), 'DisplayName', tmp, 'Color', cmap(t_n,:));
        if(mod(t_n-1,5) == 0 || n_plot <= 11)
```

```
            plot_handles(end+1) = hp;
        end
    end
    title(sprintf('IC = (%s), n = %.0f',IC_str,n));
    xlabel('y');
    ylabel('T_{Fourier} - T_{FTCS}');
    ylim([-0.05,0.05]);
    xlim([0,Ly]);
    hleg = legend(plot_handles);
    set(hleg,'Location','eastoutside');

    end

    disp('Done.');

end
```