Homework 3
ASEN 5417: Numerical Methods

Ryan Skinner
Due 2015/09/29

1 Introduction

We solve the following problems to better understand numerical techniques for root finding. Applications include systems of algebraic equations, and boundary value problems governed by ordinary differential equations. As will be described in the methods section, our tools primarily consist of the bisection method, the fourth-order Runge-Kutta method, and the secant method.

1.1 Problem 1

Solve the system of equations,

$$\begin{aligned} x_1 + x_2 - \sqrt{x_2} - \tfrac{1}{4} &= 0 \\ 8x_1^2 - 8x_1 x_2 + 16x_2 - 5 &= 0\,, \end{aligned} \tag{1}$$

by simple iteration, starting with $x_{1_0} = x_{2_0} = 0$, and with an iteration tolerance of $\epsilon = 10^{-6}$.

1.2 Problem 2

Calculate the boundary value problem of free convection along a vertical plate. This problem is governed by similarity equations of the form

$$\begin{aligned} F''' + 3FF'' - 2F'^2 + \theta &= 0 \\ \theta'' + 3\mathrm{Pr}F\theta' &= 0\,, \end{aligned} \tag{2}$$

where $\theta = \theta(\eta)$, $F = F(\eta)$. The boundary conditions are

$$\begin{aligned} \eta = 0: &\qquad F = F' = 0, \qquad \theta = 1 \\ \eta \to \infty: &\qquad F' \to 0, \qquad \theta \to 0\,. \end{aligned} \tag{3}$$

As formulated, this is essentially a "double-shooting" problem. For this homework, we make the following assumptions to simplify analysis:

1. More boundary conditions are known. Specifically,

$$\theta' = \begin{cases} -0.5671 & \text{if } \mathrm{Pr} = 1 \\ -1.17 & \text{if } \mathrm{Pr} = 10 \end{cases} \tag{4}$$

2. Thus the problem is reduced to a "single-shooting" problem, with coupled equations. Good starting values for the missing BC at $\eta = 0$ are $F''(0) = \{0.6, 0.41\}$ for $\mathrm{Pr} = \{1, 10\}$.
3. With $\Delta\eta = 0.02$, integrate these equations over the domain $0 \le \eta \le 10$.

Use the fourth-order Runge-Kutta method coupled with the secant method to numerically integrate this set of equations for $\mathrm{Pr} = \{1, 10\}$. It is sufficient to set the convergence criterion for the root finder to $\epsilon = 10^{-3}$. We choose $\epsilon = 10^{-6}$ for improved accuracy.

Plot F, F', and θ as a function of η for each case, and discuss the differences between the two solutions.

2 METHODOLOGY

2.1 PROBLEM 1

Note that the system (1) can be re-written as

$$\begin{aligned} \sqrt{x_2} - x_2 + \tfrac{1}{4} &= x_1 \\ f(x_2) \;=\; 8\left(\sqrt{x_2} - x_2 + \tfrac{1}{4}\right)^2 - 8\left(\sqrt{x_2} - x_2 + \tfrac{1}{4}\right)x_2 + 16x_2 - 5 &= 0\;. \end{aligned} \tag{5}$$

In this form, we apply a simple one-dimensional root finding algorithm to x_2, and then calculate the exact value of x_1.

We use the **bisection method** to determine x_2. First, we calculate values of $f(x)$ at the values $x = \{x_{2_0}, x_{2_0} \pm h\}$. If the sign of $f(x)$ changes over one of these two intervals, we bisect the interval and evaluate $f(x)$ at the bisection point, recursively approaching the true value of x_2. We stop when our interval is less than the absolute error ϵ. If the sign does not change within the interval $x_{2_0} \pm h$, the user must provide a more accurate guess of x_{2_0}, or decrease h in the case of multiple roots.

For this problem, we choose $h = 0.3$ and $x_{2_0} = 0.3$, so that $x_2 = 0$ is still included in our initial guess, as requested in the problem statement.

2.2 PROBLEM 2

To integrate the differential equation, we use the standard fourth-order Runge-Kutta (RK4) method, which was detailed at length in Homework 2. Boundary conditions are known and finite at $\eta = 0$, so this is where we start integration. To approximate $\eta = \infty$, it is sufficient to apply the corresponding 'far-field' boundary conditions at $\eta = 10$. Since the exact form of the governing equations is known, the only unknown is $F''(0)$.

We seek the appropriate value of $F''(0)$ with the **secant method**, using the function $f[F''(0)] = F'(10)$ for our root finder. Given two initial guesses x_1 and x_2 for $F''(0)$, the secant method calculates the next guess x_{n+1} as

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\;. \tag{6}$$

In our application, of course, f takes the value of $F''(0)$ as input and returns $F'(10)$. Once the condition

$$\left|\frac{x_{n+1} - x_n}{x_n}\right| < \epsilon\;, \tag{7}$$

is satisfied, where ϵ is the relative error, solution ceases and the root is declared to occur at $x_0 = x_{n+1}$.

3 RESULTS

3.1 PROBLEM 1

We find $\{x_1, x_2\} = \{0.5000000000, 0.2499999046\}$. For curiosity's sake, convergence behavior of the solution for x_2 is presented in Figure 1.

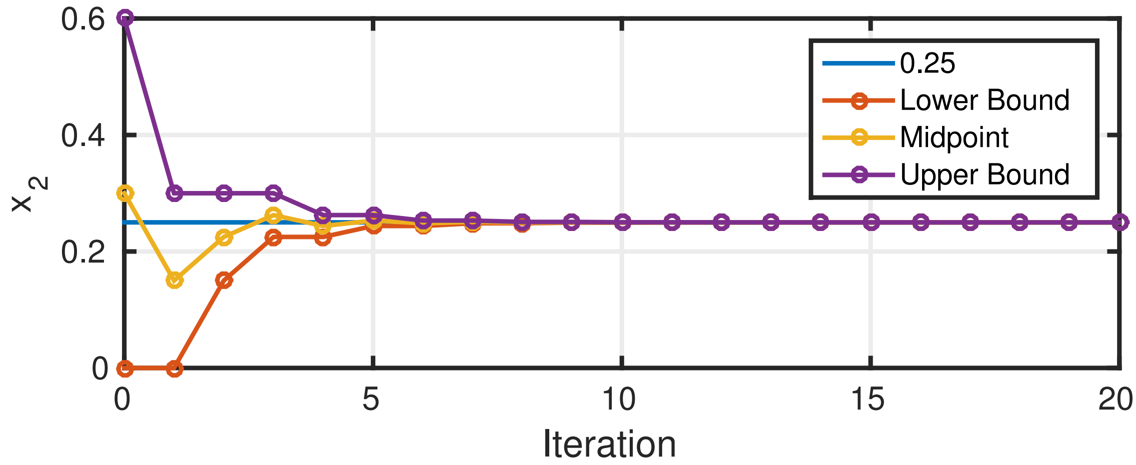

Figure 1: Convergence behavior of the bisection method as it solves for x_2.

3.2 Problem 2

We find $F''(0) = \{0.64208, 0.41967\}$ for $\mathrm{Pr} = \{1, 10\}$. Plots of F, F', and θ as functions of the similarity variable are presented in Figure 2.

4 Discussion

4.1 Problem 1

Analytical evaluation reveals that $\{x_1, x_2\} = \{\frac{1}{2}, \frac{1}{4}\}$ is the solution to (1). In this light, the bisection method is entirely adequate in producing the correct solution, but it required 20 steps compared to the presumably fewer steps the secant method would have required using the same starting interval.

Furthermore, in solving the x_2-equation in (5) directly rather than the coupled equations in (1), our numerical tolerance only applies to x_2. For a more challenging system of equations, one would need to propagate the tolerance in x_2 to determine tolerance in x_1. Since implementing the root-finding procedure is the primary objective of this assignment, we note only that ϵ can be decreased by the user if they desire a more accurate value of x_1.

4.2 Problem 2

For both $\mathrm{Pr} = 1$ and $\mathrm{Pr} = 10$, the root finder performs well. The boundary conditions $F' \to 0$ and $\theta \to 0$ as $\eta \to \infty$ are satisfied within an absolute tolerance of less than 10^{-5}.

Initial attempts at solving this problem encountered two difficulties: high sensitivity of the secant method to the initial guesses for $F''(0)$, and stability of the RK4 method. For $\mathrm{Pr} = \{1, 10\}$, the initial guess intervals were $[0.60, 0.61]$ and $[0.41, 0.46]$, respectively. For most values of η greater than the upper bound on these relatively small intervals, the RK4 method was unstable, causing the value of $F'(10)$ to blow up and the root finder to fail. This experience demonstrates the importance of understanding the interplay of all numerical methods used in a problem where

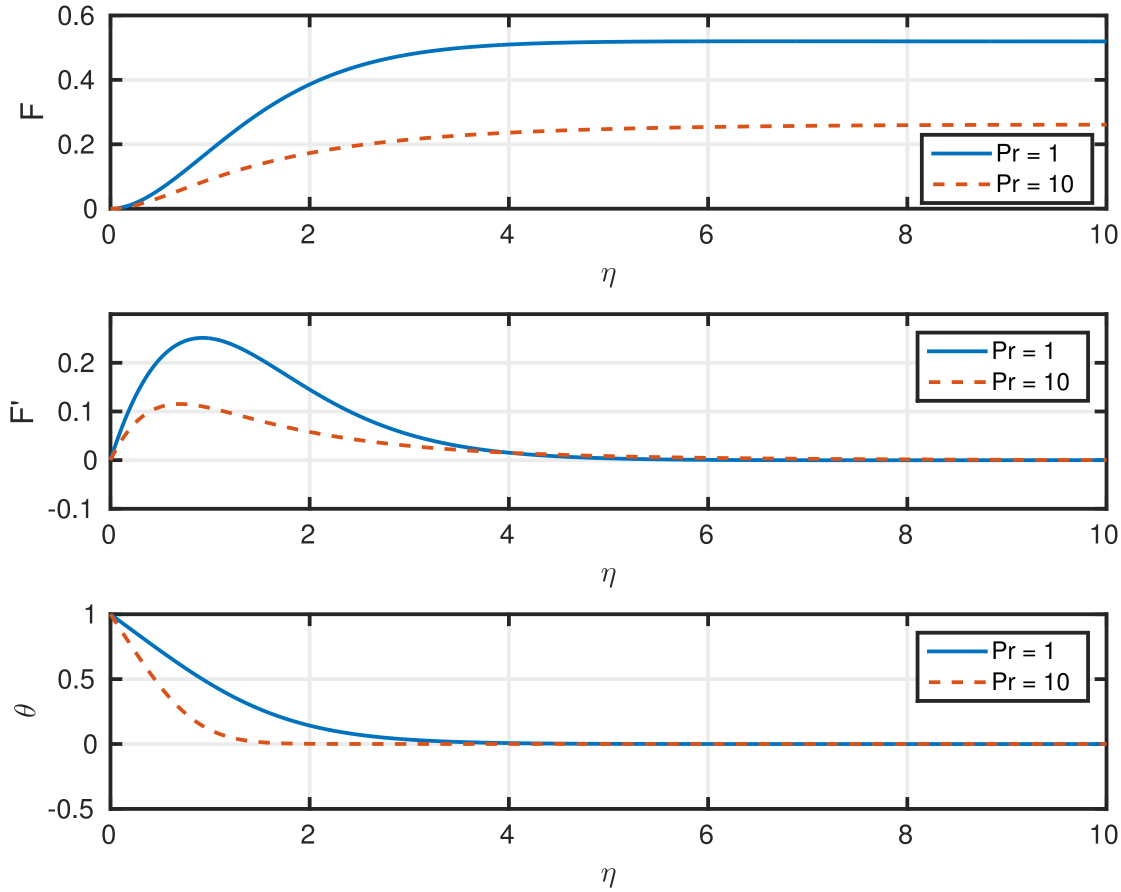

Figure 2: Plots of F, F', and θ as functions of the similarity variable η.

multiple techniques are employed. As such, shooting methods may not be the best choice for all boundary-value problems, and there are likely cases where finite difference, finite element, or spectral methods are preferred.

To discuss differences between Prandtl numbers, we note that F is the dimensionless stream function (thus F' is the velocity) and θ is the dimensionless temperature. Furthermore, the similarity variable η is proportional to y, the distance to the surface. For details on problem formulation, see Ishak (2010).

Since the Prandtl number is the ratio of viscous to thermal diffusion rates, we expect a flow with high Prandtl number to diffuse heat more slowly than momentum. This is indeed the case: the thermal boundary layer is much smaller (less developed due to lower thermal diffusivity) for our $\mathrm{Pr} = 10$ case, meaning a steeper temperature gradient near the wall and thus a higher rate of convective heat transfer from the surface to the fluid.

The same trend manifests in the behavior of F', in that the lower $\mathrm{Pr} = 1$ case has a higher maximum velocity near the surface because momentum diffusivity is higher for that case.

Of course, as $\eta \to \infty$, both velocity and temperature approach their free-stream values regardless of Prandtl number.

5 References

Ishak, A. (2010). Similarity solutions for flow and heat transfer over a permeable surface with convective boundary condition. *Applied Mathematics and Computation, 217*, 837–842.

Appendix: MATLAB Code

The following code listings generate all figures presented in this homework assignment.

Listing 1: Problem_1.m

```
function [] = Problem_1()

    %%%%%%
    % Finds the root of a function of two variables, using the bisection method.
    %
    % Ryan Skinner, September 2015
    %%%

    Set_Default_Plot_Properties();

    % Initialize functions.
    x1 = @(x2) sqrt(x2) - x2 + (1/4);
    f  = @(x2) 8*x1(x2).^2 - 8*x1(x2).*x2 + 16*x2 - 5;

    % Find x2.
    initial = 0.3;
    h = 0.3;
    [x2, guesses] = Bisect1D(f, initial, h, 10^-6);

    % Display results.
    fprintf('x1: %.10f\n', x1(x2));
    fprintf('x2: %.10f\n',    x2);

    figure();
    hold on;
    abscissa = 0:(length(guesses)-1);
    plot(abscissa, 0.25 * ones(1,length(abscissa)));
    plot(abscissa, guesses, 'o-');
    xlabel('Iteration');
    ylabel('x_2');
    hleg = legend('0.25', 'Lower Bound', 'Midpoint', 'Upper Bound');

end
```

Listing 2: Bisect1D.m

```
function [ x0, x_guesses ] = Bisect1D( f, x_initial, h, tol )

    %%%%%%
    % Simple bisection method to find root of function, f(x), based on an initial guess
    %  of the interval (x_initial +/- h). Solution is found when the interval decreases
    %  below the tolerance, tol.
    %
    % Ryan Skinner, September 2015
    %%%

    % Initialize the three test points.
    x = [ x_initial - h; ...
          x_initial;     ...
          x_initial + h ]';

    % Initialize reporting data.
    x_guesses = x;
```

```

    % Iterate until tolerance is met.
    interval = inf;
    while interval > tol

        % Determine where sign changes and update the interval if needed.
        fx = f(x);
        sign_change = (diff(sign(fx)) ~= 0);
        if sign_change(1)
            x = [ x(1);         ...
                  mean(x(1:2)); ...
                  x(2) ]';
        elseif sign_change(2)
            x = [ x(2);         ...
                  mean(x(2:3)); ...
                  x(3) ]';
        else
            error('No sign change within interval.');
        end

        % Re-calculate interval.
        interval = abs( x(3) - x(1) );

        % Catalog the current guess.
        x_guesses = cat(1, x_guesses, x);

    end

    % Return mid-point of interval.
    x0 = x(2);

end
```

Listing 3: Problem_2.m

```
function [] = Problem_2()

    %%%%%%
    % Solves the boundary value problem for free convection along a plate for Prandtl
    %  numbers of Pr = {1, 10}, using an RK4 integrator and secant-method root-finder.
    %
    % Ryan Skinner, September 2015
    %%%

    Set_Default_Plot_Properties();

    % Initialize root-finding functions for Prandtl numbers Pr = {1, 10}.
    Pr1  = @(x) theta10(x, 1);
    Pr10 = @(x) theta10(x, 10);

    % Find value of F'' using the secant method.
    fprintf('Pr = 1\n');
    [a.x0, a.x] = Secant1D(Pr1,  [0.60, 0.61], 1e-6);
    fprintf('Pr = 10\n');
    [b.x0, b.x] = Secant1D(Pr10, [0.41, 0.46], 1e-6);

    % Report values of F''.
    fprintf('Pr = 1 :   F''''(0) = %.5f\n', a.x0);
    fprintf('Pr = 10:   F''''(0) = %.5f\n', b.x0);

    % Solve differential system with optimized values of F''.
    [Ta,Ya] = RK4(@(t, y) convection(t, y, 1),  [0,10], 500, [0,0,a.x0,1,-0.5671]);
    [Tb,Yb] = RK4(@(t, y) convection(t, y, 10), [0,10], 500, [0,0,b.x0,1,-1.17]);

    % Plot F.
    figure();
    hold on;
    plot(Ta,Ya(:,1), '-','DisplayName','Pr = 1');
    plot(Tb,Yb(:,1),'--','DisplayName','Pr = 10');
```

```
    xlabel('\eta');
    ylabel('F');
    hleg = legend('show');
    set(hleg,'Location','southeast');

    % Plot F'.
    figure();
    hold on;
    plot(Ta,Ya(:,2), '-','DisplayName','Pr = 1');
    plot(Tb,Yb(:,2),'--','DisplayName','Pr = 10');
    xlabel('\eta');
    ylabel('F''');
    hleg = legend('show');
    set(hleg,'Location','northeast');

    % Plot theta.
    figure();
    hold on;
    plot(Ta,Ya(:,4), '-','DisplayName','Pr = 1');
    plot(Tb,Yb(:,4),'--','DisplayName','Pr = 10');
    xlabel('\eta');
    ylabel('\theta');
    hleg = legend('show');
    set(hleg,'Location','northeast');

end

function dy = convection(~, y, Pr)

    %%%%%%
    % Function relating solution variables to their derivatives in the equation for free
    %  convection along a vertical plate.
    %%%

    %  y = F , F' , F'' , theta , theta'
    % dy = F', F'', F''', theta', theta''

    dy = zeros(5,1);
    dy(1) = y(2);
    dy(2) = y(3);
    dy(3) = -y(4) + 2 * y(2).^2 - 3 * y(1) .* y(3);
    dy(4) = y(5);
    dy(5) = -3 * Pr * y(1) .* y(5);
end

function val = theta10(Fpp0, Pr)

    %%%%%%
    % Returns the value of theta(10) reached by RK4 integration using the given values of
    %  F''(0) and Prandtl number.
    %%%

    if Pr == 1
        thp0 = -0.5671;
    elseif Pr == 10
        thp0 = -1.17;
    else
        error('Prandl number must be 1 or 10');
    end

    initials = [0, 0, Fpp0, 1, thp0];
    odefun = @(t, y) convection(t, y, Pr);
    [~,Y] = RK4(odefun, [0,10], 500, initials);
    val = Y(end,2);
end
```

Listing 4: RK4.m

```
function [ T, Y ] = RK4( odefun, tspan, N, y0 )

    %%%%%%
    % Solves a differential equation using the RK4 method.
    % INPUTS: odefun -- function handle to the system of odes (as in ode45)
    %          tspan -- vector specifying the interval of integration
    %              N -- number of time steps
    %             t0 -- vector of initial conditions
    %
    % Ryan Skinner, September 2015
    %%%

    % Number of equations to integrate.
    neq = length(y0);

    % Number of preliminary values calculated by RK4 method.
            nk = 4;
       t_coeff = [0, 1, 1, 1]' / 2;
       k_coeff = [0, 1, 1, 2]' / 2;
    ynp1_coeff = [1, 2, 2, 1]' / 6;

    % Time-like variable at which to evaluate solution, and step size.
    T = linspace(tspan(1), tspan(2), N)';
    h = T(2) - T(1);

    % Solution vector with initial conditions.
    Y = zeros(N, neq);
    Y(1,:) = y0;

    %%%
    % Perform integration using RK4.
    %%%

    % Loop over time steps.
    for n = 1:(length(T)-1)
        k = zeros(neq, nk+1);
        % Loop over Runge-Kutta intermediary values k_1, k_2, ...
        for i = (1:nk)+1
            dy = odefun( T(n)    + t_coeff(i-1) * h,       ...
                         Y(n,:)' + k_coeff(i-1) * k(:,i-1) );
            k(:,i) = h * dy;
        end
        Y(n+1,:) = Y(n,:) + (k(:,2:nk+1) * ynp1_coeff)';
    end

end
```

Listing 5: Secant1D.m

```
function [ x0, x ] = Secant1D( f, x_initials, tol )

    %%%%%%
    % Simple secant method to find root of function, f(x), based on an initial guess
    %  of the interval (x_initials). Solution is found when the relative error decreases
    %  below the tolerance, tol.
    %
    % Ryan Skinner, September 2015
    %%%

    % Initialize the test points.
    x = x_initials;
    y = [f(x(1)), f(x(2))];

    % Iterate until tolerance is met.
    relerr = inf;
    while relerr > tol

        % Calculate next point using the secant method.
        x(end+1) = x(end) - y(end) * (x(end) - x(end-1)) / (y(end) - y(end-1));
```

```
        % Calculate the next y-value.
        y(end+1) = f(x(end));

        % Re-calculate relative error.
        relerr = abs((x(end) - x(end-1)) / x(end-1));

        % Print convergence if desired.
        fprintf('x: %10.5f, err: %10.5e\n', x(end), relerr);
        if isnan(x(end))
            error('Secant method diverged.');
        end

    end

    % Return most recent guess of x.
    x0 = x(end);

end
```