# 1 INTRODUCTION

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

## 1.1 PROBLEM 1

Solve the system of equations,

$$\begin{aligned} x_1 + x_2 - \sqrt{x_2} - \tfrac{1}{4} &= 0 \\ 8x_1^2 - 8x_1x_2 + 16x_2 - 5 &= 0\,, \end{aligned} \tag{1}$$

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 0.6 for $\mathrm{Pr} = 1$; and 0.41 for $\mathrm{Pr} = 10$.
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 $\epsilon = 10^{-3}$.

Plot $F$, $F'$, and $\theta$ as a function of $\eta$ 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 $\epsilon$. 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. 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 $F''(\eta)$ at $\eta = 0$. We seek the appropriate value of $F''(0)$ with the secant method.

## 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.

### 3.2 PROBLEM 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.

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

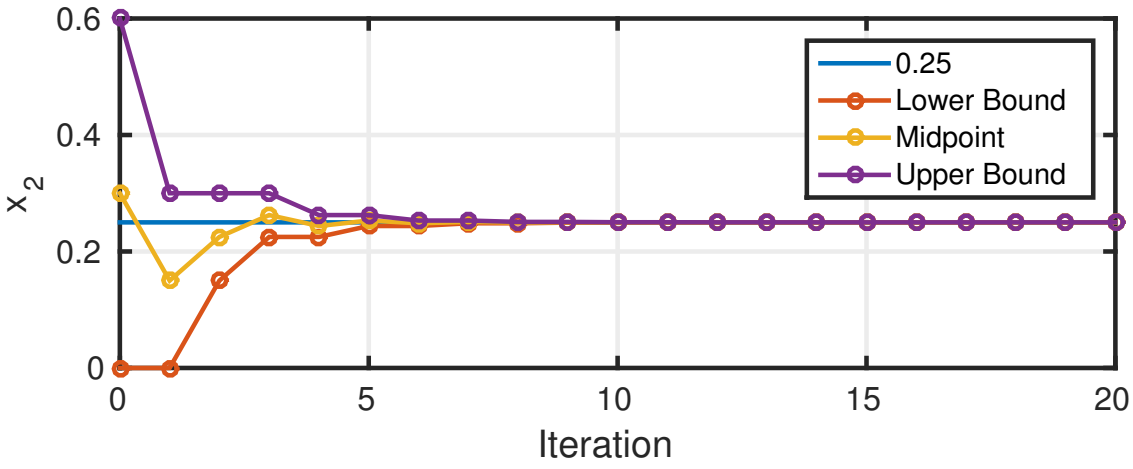


**Figure 1:** Convergence behavior of the bisection method as it solves for $x_2$.

the root-finding procedure is the primary objective of this assignment, we note only that $\epsilon$ can be decreased by the user if they desire a more accurate value of $x_1$.

### 4.2 Problem 2

## 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.

**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.
    %%%

    % 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
```