Question 2

```
function q2()
    fprintf("%d\n", BinarySearch(@(d) abs(d) * (1 + d^2) / (1 - d^2), 10, 0, 1, 1e-6));
end
```

Listing 1: Code for question 2 calls BinarySearch.m.

```
function x  = BinarySearch(f, goal, min, max, tol)
    x = (min+max)/2;
    y = f(x);
    while abs(y-goal) > tol
        if y < goal
            min = x;
        else
            max = x;
        end
        x = (min+max)/2;
        y = f(x);
    end
end
```

Listing 2: Code for Binary search called in q2.m.

Question 3

```
function y = ForwardSubstitute(L, b)
    % Inputs:
    %   L: A square lower triangular matrix
    %   b: a vector of the length as the columns of U
    % Outputs:
    %   y: A vector that is the solution to Ly = b.
    n = size(L, 1);
    % obtain the first element of y
    y = b / L(1, 1);
    % obtain the other elements of y by iterating downwards in L.
    for k=2:n
        y(k) = (1 / L(k, k)) * (b(k) - L(k, 1:k-1)*y(1:k-1));
    end
end
```

Listing 3: Code used for question 3a.

```
function x = BackwardSubstitute(U, b)
    % Inputs:
    %   U: A square upper triangular matrix
    %   b: a vector of the length as the columns of U
    % Outputs:
    %   x: A vector that is the solution to Ux = b.
    n = size(U, 1);
    % obtain the last element of x
    x = b / U(n, n);
    % obtain the other elements of x by iterating upwards in U.
    for k=n-1:-1:1
        x(k) = (1 / U(k, k)) * (b(k) - U(k, k+1:n) * x(k+1:n));
    end
end
```

Listing 4: Code used for question 3d.

```
function A = LUDecomposition(A)
    % Inputs:
    %   A: A square matrix
    % Outputs:
    %   A: A square matrix whose upper and lower triangle (below the
    %   diagonal) are the respective U and L such that A = LU.
    n = size(A, 1);
    for k=1:n-1
        for i=k+1:n
            A(i, k) = A(i, k) / A(k, k);
            for j=k+1:n
                A(i, j) = A(i, j) - A(i, k) * A(k, j);
            end
        end
    end
end
```

Listing 5: Code used for question 3e.

```
function x = LUSolve(A, b)
    % Inputs:
    %   A: A square matrix
    %   b: a vector of the length as the columns of A
    % Outputs:
    %   x: A vector that is the solution to Ax = b done by LU Decomposition.
    A = LUDecomposition(A);
    % Perform the LU Decomposition of A
    L = tril(A, -1) + eye(size(A, 1));
    U = triu(A);
    % Solve Ly = b and Ux = y in order to solve Ax = b
    x = BackwardSubstitute(U, ForwardSubstitute(L, b));
end
```

Listing 6: Code used for question 3f.

Question 4

```
function q4c()
    % Code to produce graphs for A1.4
    n = arrayfun(@(x) 10 * 2^x, 0:5);
    for k=0:length(n)-1
        [T, X, ~] = ODESolve(@(t) 100 * (t^2 + 1), ...
                            @(t) 20 * cos(pi * t), ...
                            @(t) sin(pi * t), ...
                            2, -1, 0, 1, n(k+1), "LU");
        plot(T, X, 'DisplayName',"n = " + k);
        hold on;
    end
    legend()
end
```

Listing 7: Code used for question 4c.

```
1  function q4e()
2      addpath 'C:\Users\rjust\f2023\math578'
3      % Code to produce graphs for A1.4e
4      %[T, X, iter] = ODESolve(@(t) 1000 * (t^2 + 1), ...
5      %                 @(t) 20 * cos(pi * t), ...
6      %                 @(t) sin(pi * t), ...
7      %                 2, -1, 0, 1, 2500, "J");
8      %[T, X, iter] = ODESolve(@(t) 10^8 * (t^2 + 1), ...
9      %                 @(t) (1/20) * cos(pi * t), ...
10     %                 @(t) sin(pi * t), ...
11     %                 2, -1, 0, 1, 2500, "J");
12     [T, X, iter] = ODESolve(@(t) 10^8 * (t^2 + 1), ...
13                     @(t) (1/20) * cos(pi * t), ...
14                     @(t) sin(pi * t), ...
15                     2, -1, 0, 1, 2500, "LU");
16     plot(T, X);
17     fprintf("%i\n", iter);
18 end
```

Listing 8: Code used for question 4e. It was run three times by uncommenting lines 3-6, 7-10, and 12-15 to produce the respective left and right subplots of Figure 2 (The last two blocks produce identical plots).

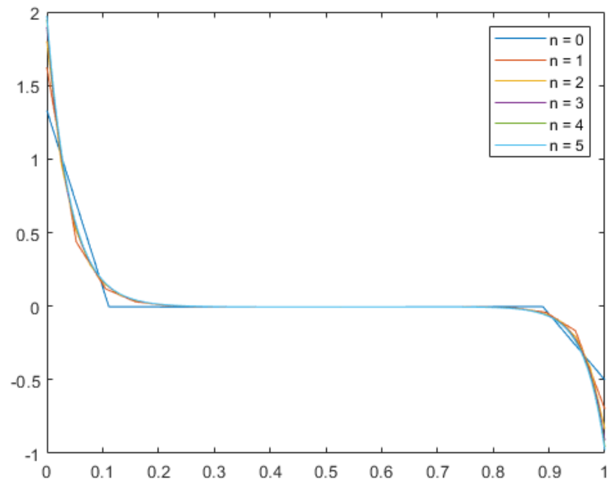

Figure 1: Graph of solution to the ODE for question 4c.

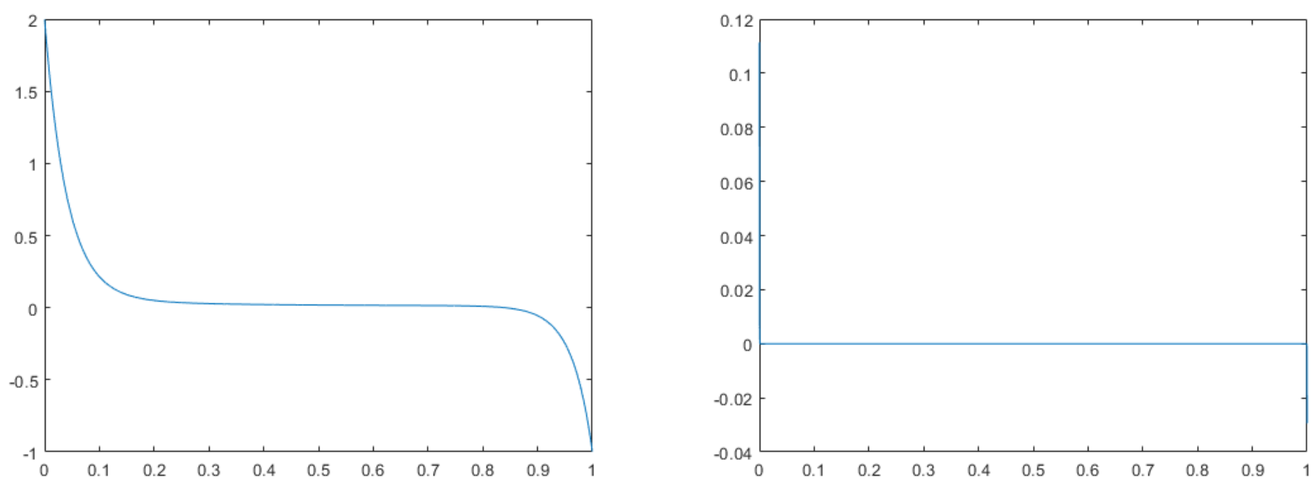

Figure 2: Graph of solution to the respective ODEs for question 4e. The new ODE given (the right subplot) presents much sharper transitions at $x = 0$ and $x = 1$.

Question 5

```
function c = LeastSquaresQuadratic(x, y)
    % Inputs:
    %   x: Vector of independant of the data
    %   y: Vector of dependant of the data
    % Outputs:
    %   c: Vector representing the coefficients of the best fitting
    %   quadratic to the data
    % Form M from x
    M = [ones(size(x)) x x.*x];

    % Solve M^TMc = M^Ty
    c = LUSolve(M.' * M, M.' * y);
end
```

Listing 9: Code used for question 5d.

```
function c = q5e()
    % code to produce graphs and error margins for 1.5e
    % form points
    addpath 'C:\Users\rjust\f2023\math578'
    x = [5, 5.5, 6.5, 8, 8.5, 10.8, 11.5, 13.7, 14.5, 15.9];
    y = [1, 4, 7, 8, 9.5, 9.2, 9, 6, 3, 1];

    % fitting
    c = LeastSquaresQuadratic(x.', y.');
    fit_fn = @(X) c(1) + c(2)*X + c(3)*X^2;

    % calculate residuals
    fprintf("%d\n", sqrt(sum((y - arrayfun(fit_fn, x)).^2)));

    % plot
    scatter(x, y)
    hold on;
    t = linspace(min(x), max(x));
    plot(t, arrayfun(fit_fn, t));
end
```

Listing 10: Code used for question 5e.