Yash Bansal

2022CS51133

Question 1 :-

Given IVP is $y' = 2xy^2$

$y(0) = 0.5$

To calculate :- y(1)

The modified euler method is used to calculate the required value using step size of 0.1

```
function val = modified_euler(derivative, x0, y0, xfinal, step_size)
	x = x0;
	y = y0;
	n = (xfinal - x0) / step_size; % The required number of iterations
	for i = 1 : n
		% Calculating the slopes at current point and the next point
		k1 = derivative(x, y);
		k2 = derivative(x + step_size, y + k1*step_size);
		% Updating the value of y as per modified euler's method
		y = y + 0.5*step_size*(k1 + k2);
		x = x + step_size; % going to the next value of x
	end
	val = y; % Final value of y
end
```

Value at x = 1 is :- 0.99406

Here, starting from the initial points, k1 and k2 are calculating according to modified euler method for all the points, and keep updating the values of current x and y.

The final value of x is 1 and the final value of y calculated through this method is 0.99406.

Question 2 :-

Given IVP is $y' = 2y/x + x^2e^x$

$y(1) = 0, \ 1 <= x <= 2$

Given exact solution :- $y = x^2(e^x – e)$

The rk4 function is used to calculate the value of y at all the node points between x = 1 and x = 2 using step size of 0.1

```
function runge_kutta_4(derivative, x0, y0, xfinal, step_size)
x = x0;
y = y0;
n = (xfinal - x0) / step_size; % Calculating no of iterations required
x_values = zeros(1, n+1); % Initialising the vector of x values
y_values_approx = zeros(1, n+1); % Initialising the vector of approximate y values
y_values_true = zeros(1, n+1); % Initialising the vector of true y values
x_values(1) = x;
y_values_approx(1) = y;
y_values_true(1) = 0;
for i = 1 : n
        % Calculating k1, k2, k3 and k4 values as per runge-kutta 4 method
        k1 = step_size * derivative(x, y);
        k2 = step_size * derivative(x + step_size/2 , y + k1/2);
        k3 = step_size * derivative(x + step_size/2 , y + k2/2);
        k4 = step_size * derivative(x + step_size, y + k3);
        % Updating the values of x and y
        y = y + (k1 + 2*k2 + 2*k3 + k4)/6;
        x = x + step_size;
        x_values(i+1) = x; % Storing the x value in the vector of x values
        y_values_approx(i+1) = y; % Storing the approximate y value calculated
        % through rk4 method in the vector of y values
        y_values_true(i+1) = x*x*( exp(x) - exp(1)); % Storing the true y values
end
true_errors = y_values_approx - y_values_true; % Calculating the vector of true errors at each node
```

Here, staring from the initial point, the values of k1, k2, k3 and k4 are calculated in accordance with the runge-kutta 4 method, and the values of x and y are updated accordingly. The value of true y's are also calculated and stored in the vector.

Finally, the true errors are calculated by subtracting the true values of y from approximate values of y and taking absolute values.

Results :-

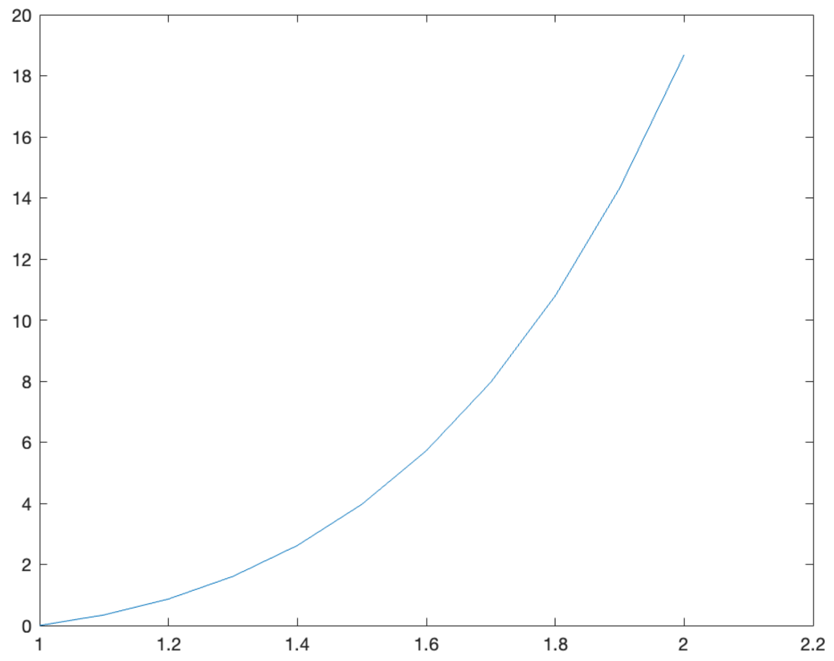
Value at x = 1 is :- 0

Value at x = 1.1 is :- 0.34591

Value at x = 1.2 is :- 0.86662

Value at x = 1.3 is :- 1.6072

```
Value at x = 1.4 is :- 2.6203

Value at x = 1.5 is :- 3.9676

Value at x = 1.6 is :- 5.7209

Value at x = 1.7 is :- 7.9638

Value at x = 1.8 is :- 10.7935

Value at x = 1.9 is :- 14.3229

Value at x = 2 is :- 18.6829


Error at x = 1 is :- 0

Error at x = 1.1 is :- 9.5892e-06

Error at x = 1.2 is :- 2.0843e-05

Error at x = 1.3 is :- 3.3731e-05

Error at x = 1.4 is :- 4.8245e-05

Error at x = 1.5 is :- 6.4396e-05

Error at x = 1.6 is :- 8.2201e-05

Error at x = 1.7 is :- 0.00010169

Error at x = 1.8 is :- 0.00012288

Error at x = 1.9 is :- 0.00014581

Error at x = 2 is :- 0.00017051
```



Question 3 :-

Given differential equation :- y'' + y = 1

y(0) = 1, y(pi / 2) = 0

The finite difference method to solve boundary value problems is used to calculate the values of y at each node point with step size of pi / 16.

```
n = 8;
h = pi / (2*n);
A = zeros(n+1, n+1); % initialising matrix A
B = zeros(1, n+1); % initialising matrix B
A(1,1) = 1;
A(n+1, n+1) = 1;
B(1) = 1;
B(n+1) = 0;
for i = 2 : n
        xi = (i-1)*h;
        A(i, i-1) = 1; % coefficient of y(i-1) was 1
        A(i, i) = h*h - 2; % coefficient of y(i) was (delta(x)^2) - 2
        A(i, i+1) = 1; % coefficient of y(i+1) was 1
B(i) = h*h; % constant term was (delta(x)^2)
end
final_ys = A\B'; % approximated values of y from finite difference method
```

Here, we got the coefficients of the terms y(i-1), y(i) and y(i+1) and the constant term by solving the following equation and comparing coefficients.

$((y(i+1) - 2y(i) + y(i-1)) / h^2) + y = 1$

The coefficient matrix is then initialised according with all the zeros, and then the required values are updated in the matrix by the coeffcients.

Also, the B matrix is initialised with all the zeros, and then updated according with the constant terms.

The final_ys matrix consisting of the final approx y values at all the nodes is then calculated by solving the linear system of equations using matlab inbuilt operator.

Results :-

Value at x = 0 is :- 1

Value at x = 0.19635 is :- 0.8046

Value at x = 0.3927 is :- 0.61673

Value at x = 0.58905 is :- 0.44364

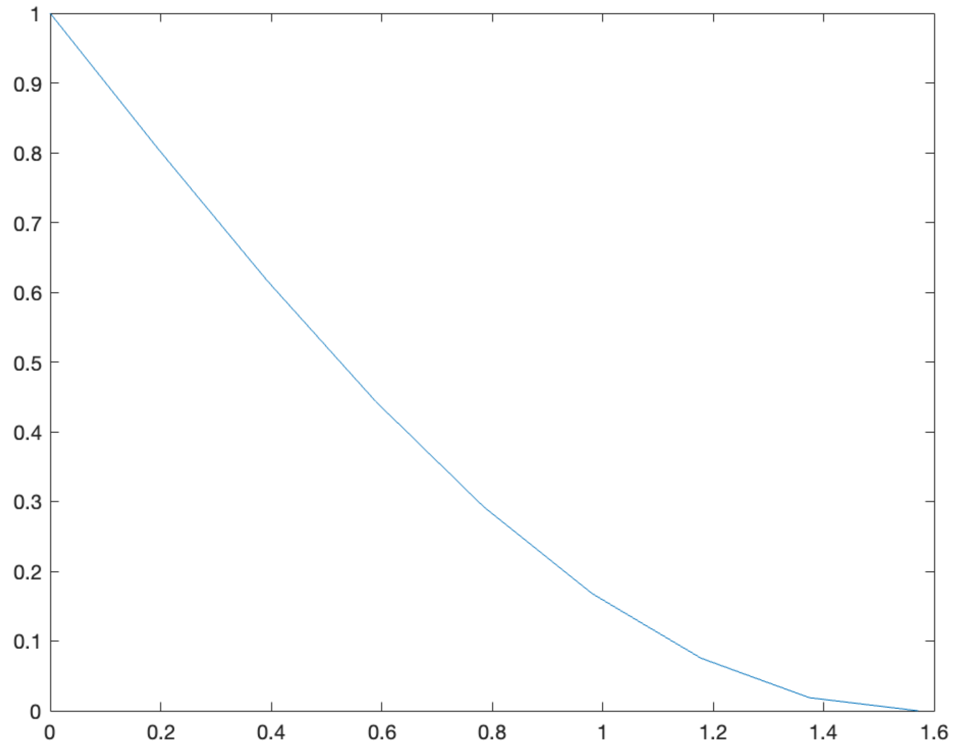
Value at x = 0.7854 is :- 0.292

Value at x = 0.98175 is :- 0.16765

Value at x = 1.1781 is :- 0.075392

Value at x = 1.3744 is :- 0.018781

Value at x = 1.5708 is :- 0


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6