Assignment 4

Name – Abhishek Agrahari

Roll Number – 190123066

Question 1-

Code

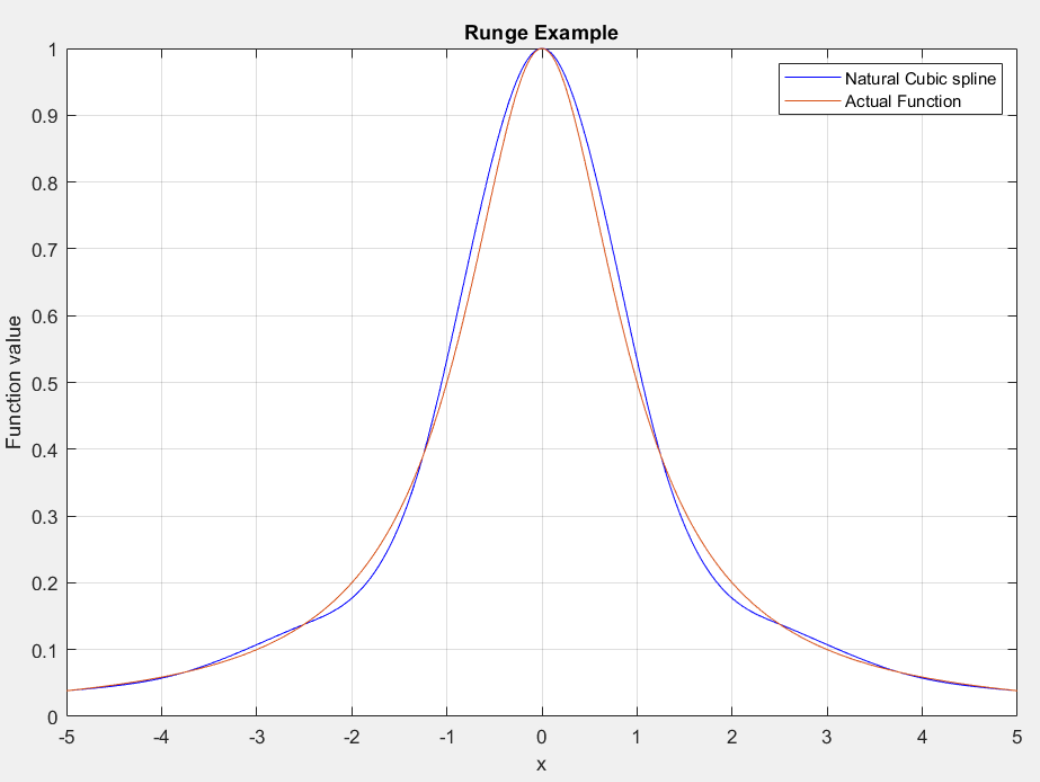
```
clc;
clear;
x = zeros(1,9);
y = zeros(1,9);
h = zeros(8);
f = @(x) 1/(1 + x^2);
for i = 1:9
    x(i) = -5 + 10*(i-1)/8;
    y(i) = f(x(i));
    if i > 1
        h(i-1) = x(i) - x(i-1);
    end
end
b = zeros(1,8);
for i = 1:8
    b(i) = (6/h(i))*(y(i + 1)-y(i));
end
u = zeros(1,7);
v = zeros(1,7);
for i = 1:7
    u(i) = 2*(h(i) + h(i + 1));
    v(i) = b(i + 1) - b(i);
end
in = zeros(7);
for i = 1:7
    in(i,i) = u(i);
    if i < 7
        in(i,i+1) = h(i);
    end
    if i > 1
        in(i,i-1) = h(i-1);
    end
end
M = in\transpose(v);
m = zeros(9);
m(1) = 0;
m(9) = 0;
m(2:8) = M;
N = 100;
xp = zeros(8,N);
yp = zeros(8,N);

for i = 1:8
    xp(i,:) = linspace(x(i),x(i + 1),N);
    for j = 1:100
        yp(i,j) = spine_interpolation(xp(i,j), m(i), m(i + 1), x(i), x(i+1), y(i), y(i + 1));
    end
    plot(xp(i,:),yp(i,:),'b');
    hold on;
```

```
end

x_actual = linspace(-5,5,300);
y_actual = 1./(1 + x_actual.^2);
plot(x_actual,y_actual);
hold on;
legend('Natural Cubic spline','','','','','','','','Actual Function'); xlabel('x'),
ylabel('Function value'), title('Runge Example'), grid on;
function val = spine_interpolation(x, m0, m1, x0, x1, y0, y1)
    h = x1-x0;
    val = ((x1-x)^3)*m0/(6*h) + ((x-x0)^3)*m1/(6*h) + (y1/h-m1*h/6)*(x-x0) + (y0/h-m0*h/6)*(x1-
x);
end
```

Ouput



Observations

- From the above output, we can observe that Natural Cubic spline estimates the given function with great accuracy.
- This is in contrast to the result we obtained in the previous lab where we estimated the given function by a single polynomial using lagrange interpolating polynomial.

Question 2-

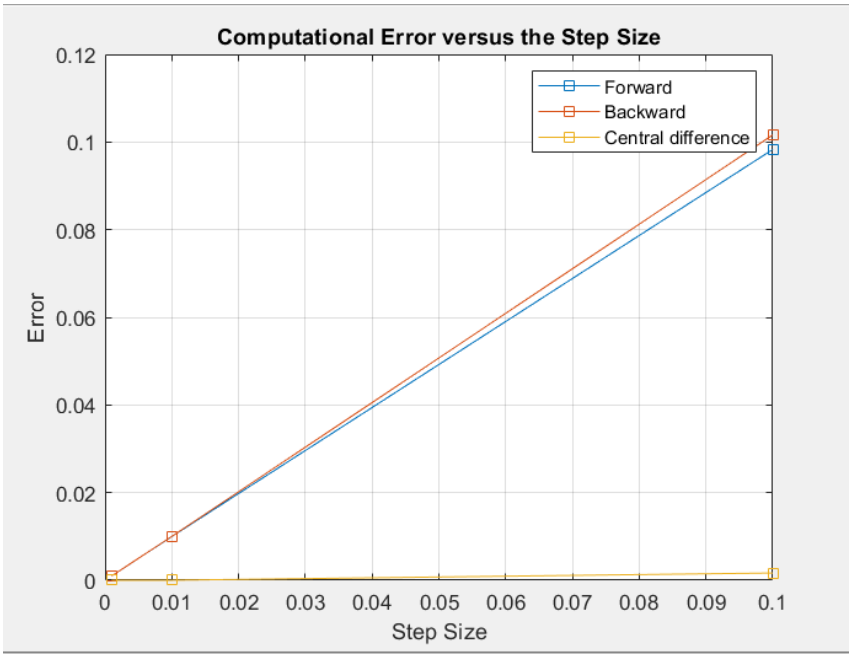
Code

```
clc;
clear;
syms x;
f = @(x) x*log(1 + x^2);
```

```
fpf = @(x,h) (f(x + h) - f(x))/h;
fpb = @(x,h) (f(x) - f(x-h))/h;
fpcd = @(x,h) (f(x + h) - f(x-h))/(2*h);
err_calc = @(a,b) abs(a-b);
x0 = 1;
fp_actual = eval((subs(diff(f,x,x0),x,x0)));
h = [0.1, 0.01, 0.001];
err = zeros(3);
fprime = zeros(3);
for i = 1:3
    f = fpf(x0,h(i));
    b = fpb(x0,h(i));
    cd = fpcd(x0,h(i));
    err(i,1) = err_calc(f, fp_actual);
    err(i,2) = err_calc(b, fp_actual);
    err(i,3) = err_calc(cd, fp_actual);
    fprime(i,1) = f;
    fprime(i,2) = b;
    fprime(i,3) = cd;
end
plot(h,err(:,1),'-s',h,err(:,2),'-s',h,err(:, 3),'-s'); legend('Forward', 'Backward', 'Central
difference'), xlabel('Step Size'),
ylabel('Error'), title('Computational Error versus the Step Size'), grid on;
fprintf("Estimated derivative value - \n");
fprintf(" h \t\t\t Forward \t\t\t Backward \t\t Central Difference\n");
for i = 1:3
    fprintf("%.3f \t %.15f \t %.15f \t %.15f\n", h(i), fprime(i,1), fprime(i,2), fprime(i,3));
end
fprintf("Error in Computation - \n");
fprintf(" h \t\t\t Forward \t\t\t Backward \t\t Central Difference\n");
for i = 1:3
    fprintf("%.3f \t %.15f \t %.15f \t %.15f\n", h(i), err(i,1), err(i,2), err(i,3));
end
```

Ouput

```
Estimated derivative value -
 h           Forward             Backward        Central Difference
0.100    1.791445865226824   1.591530198099843   1.691488031663333
0.010    1.703130472971648   1.683130556314971   1.693130514643310
0.001    1.694147013851510   1.692147013935053   1.693147013893281
Error in Computation -
 h           Forward             Backward        Central Difference
0.100    0.098298684666879   0.101616982460102   0.001659148896612
0.010    0.009983292411702   0.010016624244974   0.000016665916636
0.001    0.000999833291565   0.001000166624893   0.000000166666664
```



Observations

- We can observe that error in forward and backward difference is nearly the same.
- Error in Central difference is much less when compared to the error in forward difference and backward difference. This is because error in central difference method is $O(h^2)$, while in forward and backward difference case it is $O(h)$.

Question 3-

Code

```
clc;
clear;
h = 0.001;
x0 = 0.4;
fp_actual = 1.081072;
diff1 = @(x,h) (f(x + h) - f(x-h))/(2*h);
diff2 = @(x,h) (-f(x + 2*h) + 8*f(x + h)-8*f(x-h) + f(x-2*h))/(12*h);
err_calc = @(a,b) abs(a-b);

d1 = diff1(x0,h);
d2 = diff2(x0,h);
fprintf(" f_prime \t\t\t\t Error\n");
fprintf("%.15f \t %.15f\n", d1, err_calc(d1, fp_actual));
fprintf("%.15f \t %.15f\n", d2, err_calc(d2, fp_actual));

function val = f(x)
    X = [0.398, 0.399, 0.400, 0.401, 0.402];
    Fx = [0.408591, 0.409671, 0.410752, 0.411834, 0.412915];
    val = 0;
    for i = 1:5
        if x == X(i)
            val = Fx(i);
        end
    end
end
```

Ouput

```
Computed f_prime          Error
1.081499999999985    0.000427999999985
1.081666666666652    0.000594666666652
```

Observations

- We can observe that error in derivative estimation using central difference method is less when compared to the error using the second formula.