Homework 2016-03-11

Chuan Lu
13300180056

March 11, 2016

Problem 1.
Display the Runge Phenomenon.

Proof. **0.1 The code is shown as follows.**

```
function [ value ] = newton_coefficient( given_points, function_values )
    n = length(function_values);
    for i =1: n - 1
        function_values(i+1:n) = (function_values(i+1:n) - function_values(i:n - 1))
                ./ (given_points(i+1:n) - given_points(1:n - i));
%             disp(f_values(i:n));
    end
    value = function_values;
```

```
function [ f_series ]  = Newton_eval( given_points, function_values, eval_points)
%NEWTON_EVAL
args = newton_coefficient( given_points, function_values );

f_series = polynomail_eval(args, eval_points, given_points);

% plot( eval_points, f_series );
```

```
function [value] = polynomail_eval( args, eval_points, given_points )
value = args( length( args ) );
for i = length( args ) - 1:-1:1
    value = value .* (eval_points - given_points( i ) )
     + args( i );
end;
```

```
f = @(x)(1 ./ (1 + 25 * (x .^ 2)));
plot_points = -1:1e-6:1;
yy = feval(f, plot_points);
plot(plot_points, yy);
hold on

for n = 4:4:12
    given_points = -1:1/n:1;
    function_values = feval(f, given_points);
    f_series = Newton_eval(given_points, function_values, plot_points);
    plot(plot_points, f_series);
    hold on
end
```

```
axis([-1, 1, -1, 1]);
legend('f(x)', 'n = 4', 'n = 8', 'n = 12', 'Location', 'best');
```

0.2 The result is shown as follows.

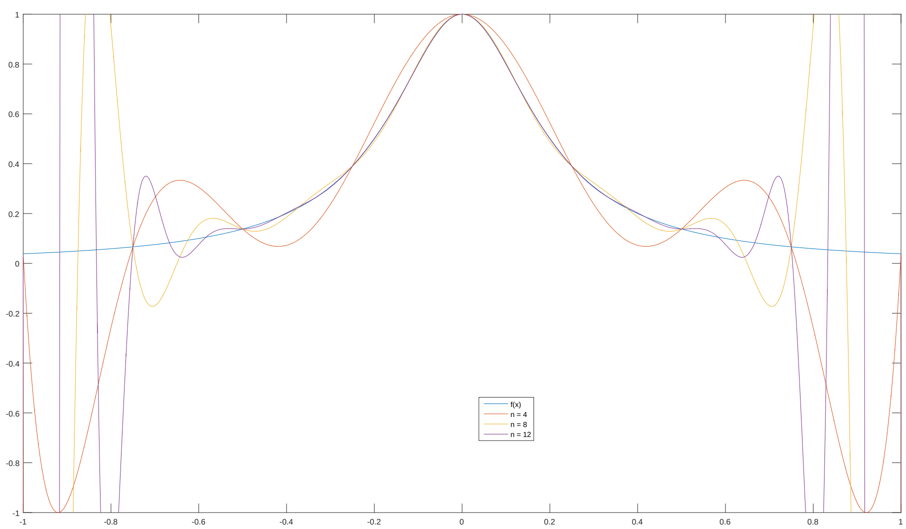

Figure 1: The Runge Phenomenon

□

Problem 2.
Calculate the integral of x^n with Newton-Cotes Formula.

Proof. **0.3 The code is shown as follows.**

```
function [ coeff ] = cotes_coefficient( order )
%Calculate the cotes coefficient with the order given
%Use the equation given at Page179, exercise3
n = order
matrix = zeros(n, n);
for i = 1 : n
    matrix( i , : ) = (1:n) .^ i;
end
c = (n .^ (1:n)) ./ (2:n+1);
c = transpose(c);
coeff = matrix \ c;
coeff = [coeff(n); coeff];
end
```

```
function[ coeff ] = get_coeff( order )
% get_coeff of newton_cotes formula of interval
% order means the order of the formula
switch order
    case 1
        coeff = [1, 1];
```

```
    case 2
        coeff = [1, 4, 1];
    case 3
        coeff  = [1,3,3,1];
    case 4
        coeff = [7,32, 12, 32, 7];
    case 5
        coeff = [19, 75, 50, 50, 75, 19];
    case 6
        coeff = [41, 216, 27, 272, 27, 216, 41];
    case 7
        coeff = [751, 3577, 1323, 2989, 2989, 1323, 3577, 751];
    case 8
        coeff = [989, 5888, -928, 10496, -4540, 10496, -928, 5888, 989];
    otherwise
        error('order should be between 0 and 8 --get_cotes_coeff');
end
    coeff = coeff / sum(coeff);
```

```
function [ coeff ] = new_cotes_coefficient( order )
%Calculate the cotes coefficient with the order given
%Use the equation given at Page179, exercise3
n = order;
matrix = zeros(n, n);
for i = 1 : n
    matrix( i , : ) = (((1:n) ./ n) .^ i);
end
c = 1 ./ ( 2: n + 1);
c = transpose(c);
coeff = matrix \ c;
coeff = [coeff(n); coeff];
end
```

```
function [ result ] = newton_cotes( func, inteval, order )
%NEWTON_COTES Use Newton_Cotes Formula to Calculate the intergral.

if nargin < 2
    error(' Too few inputted arguments, please check if function and inteval for integratin
-- Newton_Cotes');
elseif nargin == 2
    order = 2;
end

if order < 0 || order > 8
    error(' Order Error: The order should be between 0 and 8. --Newton_Cotes ');
end

low = inteval( 1 );
high = inteval( 2 );

if order == 0
    result = feval(func, (low + high) / 2) * (high - low);
else
    coeff = get_coeff(order);
```

	Exact Value	medium formula	trapezium formula	Simpson formula	3/8 formula	Cotes formula
x^1	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000
x^2	0.3333	0.2500	0.5000	0.3333	0.3333	0.3333
x^3	0.2500	0.1250	0.5000	0.2500	0.2500	0.2500
x^4	0.2000	0.0625	0.5000	0.2083	0.2037	0.2000
x^5	0.1667	0.0313	1 0.5000	0.1875	0.1759	0.1667
x^6	0.1429	0.0156	0.5000	0.1771	0.1584	0.1432
x^7	0.1250	0.0078	0.5000	0.1719	0.1471	0.1263

Table 1: Integral of x^n using Newton-Cotes Formula

```
    eval_points = low : (high - low) / order : high;
    eval_points = transpose( eval_points );
    result = sum( coeff * feval(func, eval_points)) * (high - low);
end
```

```
function func = get_func( n )
    func = @(x)(x .^ n);
```

```
% Homework1.3.1
inteval = [0, 1];
result_matrix = zeros(7, 6);
for n = 1:7
    func = get_func(n);
    result_matrix(n, 1) = integral(func, 0, 1);
    for order = 0:4
        func = get_func(n);
        result_matrix(n, order + 2) = newton_cotes(func, inteval, order);
    end
end
disp(result_matrix)
```

0.4 The result is shown as the table above.

□