《机械工程中的数值分析技术》

作业

学　　　生：易弘睿

学　　　号：20186103

专业班级：机械一班

作业编号：2021052502

重庆大学-辛辛那提大学联合学院

二〇二一年五月

4.13 Use zero- through third-order Taylor series expansions to predict $f(3)$ for

$$f(x) = 25x^3 - 6x^2 + 7x - 88$$

using a base point at $x = 1$. Compute the true percent relative error ε_t for each approximation.

```
%% 4.13
% x_p  预测位置
% x0   展开位置
% n    展开精度
% fun  符号函数

clc;clear all;
syms x;
% x_p=input('The x location you want to predict: \n');
% x0=input('The base point: \n');
% n=input('The number of times: \n');
% fun=input('The function: \n');
% TL(x_p,n,x0,fun);

x_p=3;
x0=1;
n=3;
fun=25*x^3-6*x^2+7*x-88;
TL(x_p,n,x0,fun);

function Q413 = TL(x_p,n,x0,fun)
    output=zeros(1,n+1);
    func_value=matlabFunction(fun);
    value=func_value(x_p);
    if n>0
        for i=1:1:n+1
            if i~=n+1
                temp = func_value(x0)*(x_p-x0)^(i-1)/factorial(i-1);
            else
                temp = (func_value()/factorial(i-1))*(x_p-x0)^(i-1);
                % 对匿名函数最高次求导会将其转化为常数，匿名函数无输入，否则报
错
            end
            output(i+1)=output(i)+temp;
            RE=abs(value-output(i+1));        % Absolute error
```

```
            RPE=abs(value-output(i+1))/value; % Absolute percentage error
            fprintf("----------Number of approximation: %d-----------\n", i-1);
            fprintf("Tthe true percent relative error: %.3f%%\n", RPE);
            fun=diff(fun);                    % 对此阶函数求导
            func_value=matlabFunction(fun);   % 重新转化为匿名函数
        end
    end
end
```

命令行窗口

```
----------Number of approximation: 0-----------
Tthe true percent relative error: 1.112%
----------Number of approximation: 1-----------
Tthe true percent relative error: 0.859%
----------Number of approximation: 2-----------
Tthe true percent relative error: 0.361%
----------Number of approximation: 3-----------
Tthe true percent relative error: 0.000%
```

4.16 Use forward and backward difference approximations of $O(h)$ and a centered difference approximation of $O(h^2)$ to estimate the first derivative of the function examined in Prob. 4.13. Evaluate the derivative at $x = 2$ using a step size of $h = 0.25$. Compare your results with the true value of the derivative. Interpret your results on the basis of the remainder term of the Taylor series expansion.

```
%% 4.16
% x_d   求导位置
% fun   求导函数（符号函数）
% h     差分距离

clc;clear all;
syms x;
x_d=2;
h=0.25;
fun=25*x^3-6*x^2+7*x-88;
DA(x_d,h,fun);
function Q416 = DA(x_d,h,fun)
    func_value=matlabFunction(fun);
    x=[x_d-h,x_d,x_d+h];
```

```matlab
    y=zeros(length(x));
    for i=1:length(x)
        y(i)=func_value(x(i));
    end
    diff_value = matlabFunction(diff(fun));
    tv = diff_value(x_d);
% 向前、向后、中心差分近似
    %向前差分近似
    derivative_fd = (y(3) - y(2))/h;
    error_fd = abs(tv-derivative_fd);
    %向后差分近似
    derivative_bd = (y(2)-y(1))/h;
    error_bd = abs(tv-derivative_bd);
    %中心差分近似
    derivative_cd = (y(3)-y(1))/(2*h);
    error_cd = abs(tv-derivative_cd);

% 泰勒余项近似
    % 向前、向后差分近似
    diff_2nd_value = matlabFunction(diff(diff(fun)));
    tr_fb = diff_2nd_value(x_d)/factorial(2)*h;
    % 中心差分近似
    diff_3rd_value = matlabFunction(diff(diff(diff(fun))));
    tr_cd = diff_3rd_value()/factorial(3)*h^2;
% 打印结果
    fprintf("------------------------Comparison------------------------\n")
    fprintf("True derivative: \t\t%.3f\n", tv)
    fprintf("O(h):\t%.3f\n", tr_fb)
    fprintf("Forward difference:\t\t%.3f\t\tAbsolute
Error: %.3f\n",derivative_fd, error_fd)
    fprintf("Back difference:\t\t%.3f\t\tAbsolute Error: %.3f\n",derivative_bd,
error_bd)
    fprintf("O(h):\t%.3f\n", tr_cd)
    fprintf("Centered difference:\t%.3f\t\tAbsolute
Error: %.3f\n",derivative_cd, error_cd)
    fprintf("Centered difference is more accurate.")
end
```

命令行窗口

```
-----------------------Comparison-----------------------
True derivative:        283.000
O(h):    36.000
Forward difference:     320.563     Absolute Error: 37.563
Back difference:        248.563     Absolute Error: 34.438
O(h):    1.563
Centered difference:    284.563     Absolute Error: 1.563
fx Centered difference is more accurate.>> |
```

4.19 To calculate a planet's space coordinates, we have to solve the function

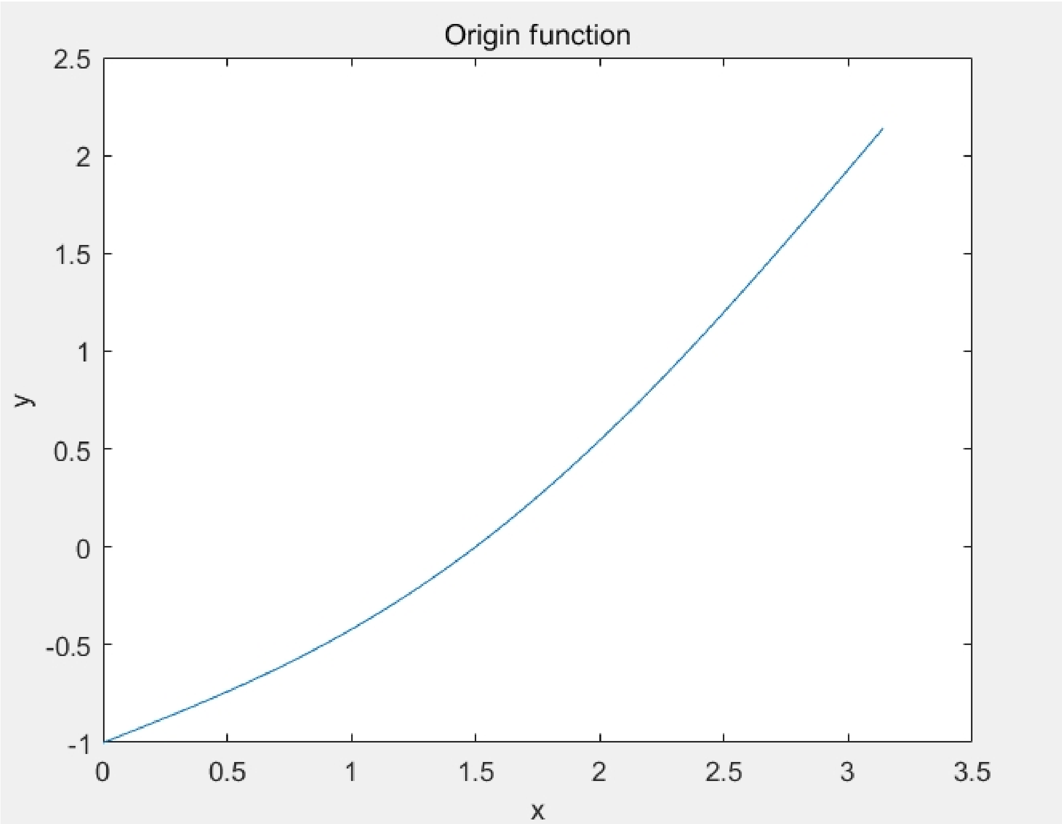
$$f(x) = x - 1 - 0.5\sin x$$

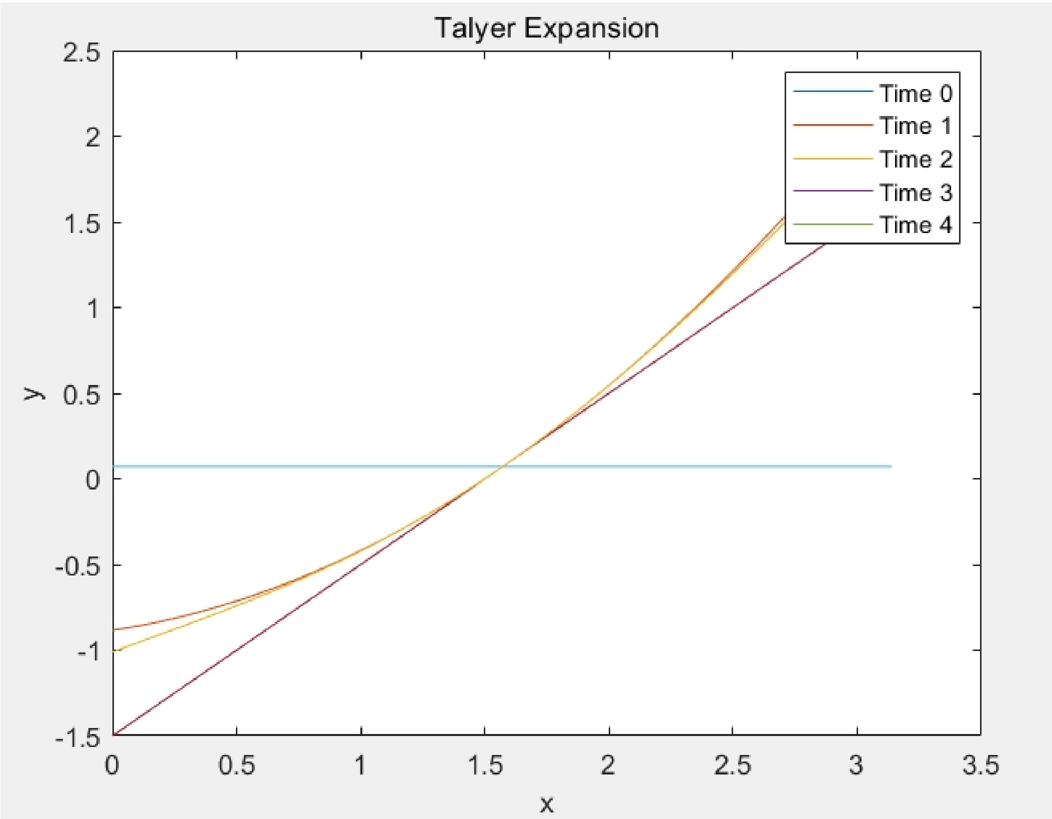
Let the base point be $a = x_i = \pi/2$ on the interval $[0, \pi]$. Determine the highest-order Taylor series expansion resulting in a maximum error of 0.015 on the specified interval. The error is equal to the absolute value of the difference between the given function and the specific Taylor series expansion. (Hint: Solve graphically.)

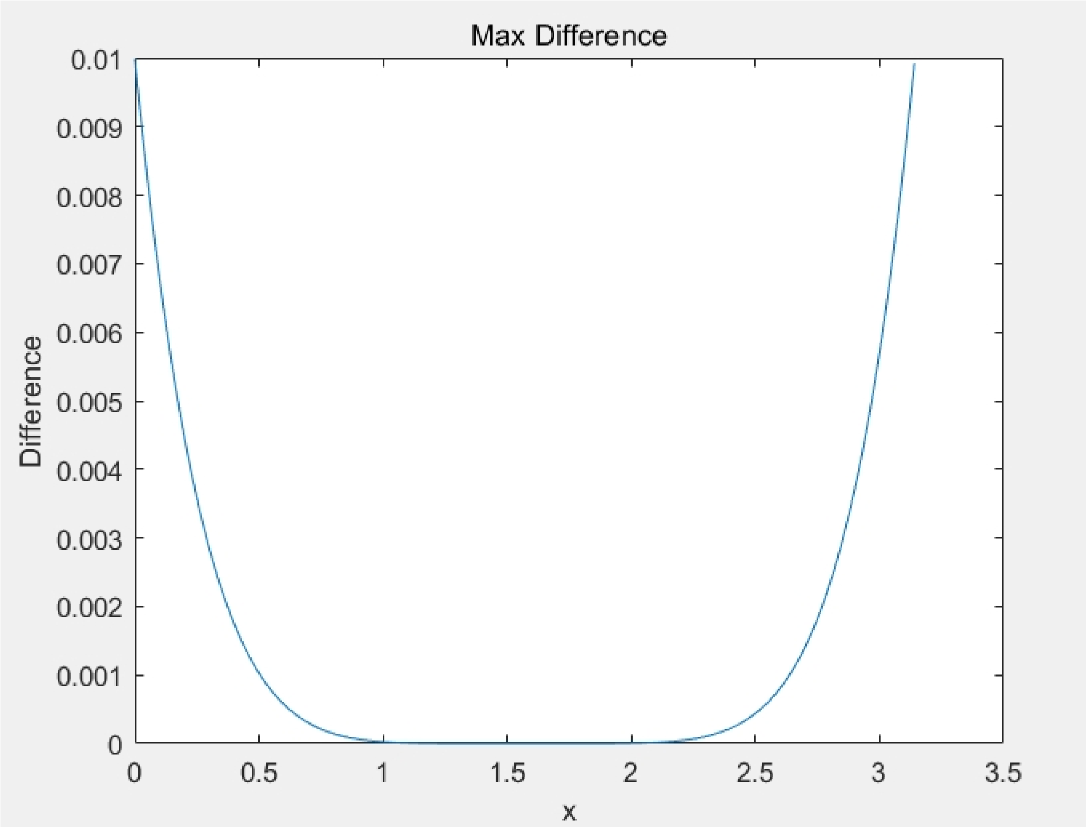
```
%% 4.19
% x_left      指区间左端
% x_right     指区间右端
% x0      指泰勒展开位置
% fun     指需要展开的方程
% error   指需要满足的误差值
clc;clear all;
syms x;
domain_left=0;
domain_right=pi;
x0=pi/2;
fun=x-1-0.5*sin(x);
error=0.015;
SC(domain_left, domain_right, x0,fun, error);
function SC(domain_left, domain_right, x0,fun, error)


    % 定义间距
    interval = 0.01;
    % x轴
    plot_x = domain_left:interval:domain_right;
```

```
len_x = length(plot_x);
% 匿名函数
value_func = matlabFunction(fun);
% 真实函数值
true_y = value_func(plot_x);
% 绘制真实函数值与x的关系
figure(1)
plot(plot_x, true_y)
xlabel("x")
ylabel("y")
title("Origin function")

i = 0;
output = zeros(1, len_x);
cal_error = 9999;

% 泰勒展开
figure(2)
while (cal_error > error)
    % 计算每个展开式的值
    step_output = (value_func(x0)/factorial(i)) * (plot_x-x0).^i;
    % 与之前展开式的值合并成为这一轮展开式的值
    output = output + step_output;
    % 计算展开式的值与真实值之间的差别，取最大的差别作为此轮差距
    cal_error = max(abs(output - true_y));
    % 绘图
    plot(plot_x, output)
    hold on
    % 执行求导
    fun = diff(fun);
    % 转化成匿名函数
    value_func = matlabFunction(fun);
    i = i+1;
end
xlabel("x")
ylabel("y")
title("Talyer Expansion")
legend("Time 0", "Time 1", "Time 2", "Time 3", "Time 4")

% 符合精确度的展开式与实际值之间的差距
figure(3)
plot(plot_x, abs(output-true_y))
xlabel("x")
ylabel("Difference")
```

```
    title("Max Difference")
end
```







4.24 One common instance where subtractive cancellation occurs involves finding the roots of a parabola, $ax^2 + bx + c$, with the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

For cases where $b^2 >> 4ac$, the difference in the numerator can be very small and roundoff errors can occur. In such cases, an alternative formulation can be used to minimize subtractive cancellation:

$$x = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$

Use 5-digit arithmetic with chopping to determine the roots of the following equation with both versions of the quadratic formula.

$$x^2 - 5000.002x + 10$$

```matlab
%% 4.24
% poly      指需要求解的多项式
% digits    指保留的小数位

clc; clear all;
syms x;
poly = [1, vpa(-5000.002, 7), 10];
digits = 5;
function PL(poly, digits)

    a = poly(1);
    b = vpa(poly(2), 7);
    c = poly(3);

    x_real_1 = (-b+sqrt(b^2-4*a*c))/(2*a);
    x_real_2 = (-b-sqrt(b^2-4*a*c))/(2*a);
    b_digits = vpa(b ,digits);

    delta = vpa(double(floor(sqrt(b^2-4*a*c)*10)/10), 5);

    x_version_11 = double(floor(((-b_digits+delta)/2*a)*10)/10);
    x_version_12 = double(floor(((-b_digits-delta)/2*a)*100)/100);

    x_version_21 = -2*c/(double(ceil(10*(b_digits+delta))/10));
    x_version_22 = -2*c/(double(ceil(10*(b_digits-delta))/10));
    disp("-----------------True Value-----------------")
    fprintf("x_1 = %.2f, x_2 = %.2f\n", x_real_1, x_real_2)
    disp("-----------------Version 1------------------")
    fprintf("x_1 = %.2f, x_2 = %.2f\n", x_version_11, x_version_12)
    fprintf("Error of x_1: %f%%\n", abs((x_version_11-x_real_1)/x_real_1)*100)
    fprintf("Error of x_2: %f%%\n", abs((x_version_12-x_real_2)/x_real_2)*100)
    disp("-----------------Version 2------------------")
    fprintf("x_1 = %.2f, x_2 = %.3f\n", x_version_21, x_version_22)
    fprintf("Error of x_1: %f%%\n", abs((x_version_21-x_real_1)/x_real_1)*100)
    fprintf("Error of x_2: %f%%\n", abs((x_version_22-x_real_2)/x_real_2)*100)
end
```

命令行窗口

```
-----------------True Value-----------------
x_1 = 5000.00,  x_2 = 0.00
-----------------Version 1------------------
x_1 = 4999.90,  x_2 = 0.05
Error of x_1: 0.002000%
Error of x_2: 2400.000000%
-----------------Version 2------------------
x_1 = 200.00,  x_2 = 0.002
Error of x_1: 96.000000%
Error of x_2: 0.001000%
```