

Labb1

Magnus Sörensen and Natnael Zekiewos

20160921

0.1 Part1

0.1.1 Objectives

Objective of this part is to implement a numerical method in MATHLAB.

0.1.2 details

The function $f(x)$ can be written like 0.1 and like the Honer's method 0.2.

$$f(x) = 6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x \quad (0.1)$$

$$P_b(x) = x(2 + x(3 + x(4 + x(5 + 6x)))) \quad (0.2)$$

0.1.3 Code

The native version of the polynomial is written like this in MATLAB.

```
1 function y = native_polynomial_calc(x)
2     % Calculate the polynymomial
3     % f(x)=6x^5 + 5x^4 + 4x^3.+ 3x^2 + 2+ 3x^2 + 2+ 3x^2 + 2x
4     y = 6.*x.^5 + 5.*x.^4 + 4.*x.^3 + 3.*x.^2 + 2.* x;
5 end
```

And the execution of $x = -\pi$ gives the result.

```
1 >> native_polynomial_calc(-pi)
2 ans =
3     -1.449772132366787e+03
```

The Honer's version of the polynomial is written like this in MATLAB.

```
1 function y = Honer_polynomial_calc(x)
2     % Honer version of the polynomial.
3     % f(x) = 6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x
4     % f(x) = x.*(2 + x.*(3 + x.*(4 + x.*(5+ 6.*x))))
5     y = x.*(2 + x.*(3 + x.*(4 + x.*(5+ 6.*x))));
6 end
```

And the output of $x = -\pi$ gives the result.

```
1 >> Honer_polynomial_calc(-pi)
2 ans =
3     -1.449772132366787e+03
```

0.1.4 Time consumption

The time consumption reported by the script provided in the aberrational instructions.

```
1 Naivemethod:
2 Elapsed time is 1.329380 seconds.
3 Horner smethod:
4 Elapsed time is 0.343049 seconds.
```

0.1.5 Why honer's function is faster.

The difference in time is do to the number of calculations the computer needed to calculate the equations. In the native version of the equations the computer required 20 calculations while in the Honer version the computer only reacquired 10 calculations to derive the correct answers. Also take in consideration that the error the computer makes while doing those 10 extra steps.

0.2 Part2

The derivative in a point of an equation that is continuous in that point can be approximated with the central dividing derivative (CDD).

$$f'_{cdd}(x) \approx \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2 * h} \quad (0.3)$$

That limit can be acquired by expanding the series 0.4 around $k = 1$. Prof.

$$f'(x) \approx \sum_{\substack{i=0 \\ i \neq k}}^{2k} \left(f(x - (k-i)h) \frac{(-1)^{k+i}(k!)^2}{(k-i)!i!(2k-i)!h} \right) \quad (0.4)$$

Proof for exansion of 0.4 around $k = 1$ is 0.3

$$\begin{aligned} &= f(x-h) * \frac{-1}{2h} + f(x+h) * \frac{1}{2 * h} \\ &= \frac{-f(x-h)}{2h} + \frac{f(x+h)}{2 * h} = \frac{f(x+h) - f(x-h)}{2h} \quad \clubsuit \end{aligned}$$

For a even more precise central dividing derivative we set $k = 2$ and get the flowing expansion of 0.4.

$$\begin{aligned} &f(x - (2-0)h) \frac{(-1)^{2+0}(2!)^2}{(2-0) * 0! * (2 * 2 - 0)! * h} + f(x - (2-1)h) \frac{(-1)^{2+1}(2!)^2}{(2-1) * 1! * (2 * 2 - 1)! * h} \\ &+ f(x - (2-3)h) \frac{(-1)^{2+3}(2!)^2}{(2-3) * 3! * (2 * 2 - 3)! * h} + f(x - (2-4)h) \frac{(-1)^{2+4}(2!)^2}{(2-4) * 4! * (2 * 2 - 4)! * h} \\ &\Rightarrow f(x-2h) \frac{4}{48h} + f(x-h) \frac{-4}{6h} + f(x+h) \frac{4}{-6h} + f(x+2h) \frac{4}{48h} \\ &= \frac{f(x-2h) + f(x+2h)}{12h} + \frac{2 * (f(x+h) - f(x-h))}{3 * h} \end{aligned}$$

0.2.1 code.

The code written in the listing is using a for loop to generate the general solution for all values of k .

```
1 function y = deriv(f, x, k, h)
2 % Calculates the k derivative of function f = f(x) at point x.
3     y = 0;
4     for i = 0:2*k
5         if i ~= k
6             y = y + f(x - (k - i).*h).*((-1).^(k+i)).*(factorial(k)).^2/((k
7             -i).*factorial(i).*factorial(2.*k - i).*h);
8         end
9     end
10 end
```

0.2.2 Plot of the log error.

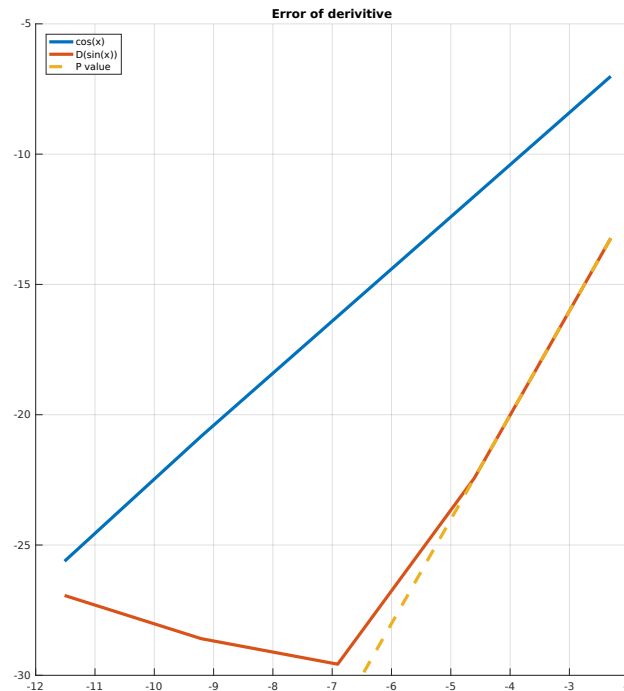


Figure 0.1: plot of the error of derivative when $k = 1$ and $\cos(x)$ as a reference

The figure 0.1 shows the error for the error for both $\cos(x)$ and $\frac{d}{dx}\sin(x)$ where a straight line is the most wanted result. The knee at -7 for $\frac{d}{dx}\sin(x)$ is due to the computer in

ability to calculate small numbers. So for the $\frac{d}{dx}\sin(x)$ our p value lined by dotted lines in the graph have a value around $p \approx 4$. Code to draw that figure.

```

1 function plot_error(f)
2     % Plots the error of dx
3
4     f = @(x) sin(x);
5     h = 1./10.^(1:5);
6     x = 1;
7     yE = cos(x);
8     for i = 1:5
9         y(i, 1) = deriv(f, x, 1, h(i));
10        y(i, 2) = deriv(f, x, 2, h(i));
11        log_err1(i) = log(abs(yE - y(i, 1)));
12        log_err2(i) = log(abs(yE - y(i, 2)));
13        log_h(i) = log(h(i));
14    end
15    p1 = 1; p2=2;
16    for px = 1:4
17        delta_x(px) = log_h(px) - log_h(px+1);
18        delta_y(px) = log_err2(px) - log_err2(px+1);
19        p(px) = delta_y/delta_x;
20    end
21    disp('p(1)='); disp(p(1));
22    m = log_err2(1) - p(1) .* log_h(1)
23    disp('p=');
24    disp(transpose(p));
25    disp('medium p');
26    disp(sum(p)/4)
27    disp('log_h');
28    disp(transpose(log_h));
29    axis([-12 -2 -30 -5]);
30    hold on;
31    grid on;
32    plot(log_h, log_err1, 'LineWidth', 3)
33    plot(log_h, log_err2, 'LineWidth', 3)
34    plot(log_h, p(1)*(log_h)+m, '--', 'LineWidth', 3)
35    legend('cos(x)', 'D(sin(x))', 'P value', 'Location', 'northwest');
36    title('Error of derivitive');
37 end

```