Problem 1

Midpoint rule: this algorithm aproximates the integral of a function f over one panel $[x_j, x_{j+1}]$ by

$$\int_{x_j}^{x_{j+1}} f(x)\,dx \cong h f(\varepsilon) \quad \text{with} \begin{cases} h = (x_{j+1} - x_j) \\ \varepsilon = (x_j + x_{j+1})/2 \end{cases}$$

the error of this aproximation can be calculated by replacing the function with its Taylor Series expansion at ε and integrating

$$f(x) = f(\varepsilon) + (x-\varepsilon) f'(\varepsilon) + \frac{(x-\varepsilon)^2}{2} f''(\varepsilon) + \frac{(x-\varepsilon)^3}{6} f'''(\varepsilon) + \cdots$$

$$\int_{x_j}^{x_{j+1}} f(x) = f(\varepsilon)\,x \Big|_{x_j}^{x_{j+1}} + f'(\varepsilon)\frac{(x-\varepsilon)^2}{2}\Big|_{x_j}^{x_{j+1}} + f''(\varepsilon)\frac{(x-\varepsilon)^3}{6}\Big|_{x_j}^{x_{j+1}} + \cdots$$

notice that $(x_{j+1} - \varepsilon) = -(x_j - \varepsilon)$ since ε is the mid point, which implies that $(x_{j+1} - \varepsilon)^2 = (x_j - \varepsilon)^2$ and thus even terms vanish

$$\int_{x_j}^{x_{j+1}} f(x) = h f(\varepsilon) + \frac{h^3}{24} f''(\varepsilon) + \cdots \quad \text{(exactly equals)}$$

$\therefore$ when $f''(\varepsilon) = 0$ (and thus f^{IV}, f^{VI} ... also) we get

$$\int_{x_j}^{x_{j+1}} f(x) = h f(\varepsilon)$$ and thus the aproximation has no error terms

$$f''(\varepsilon) = 0 \Rightarrow f'(\varepsilon) = C \Rightarrow f(\varepsilon) = Cx + D$$ a straight line $\square$

Simpons' Rule : Similarly, this algorithm can be shown by combining the midpoint and trapezoidal rules to have an error:

$$\int_{x_j}^{x_{j+2}} f(x)\,dx = \frac{h}{3}\left(f_j + 4f_{j+1} + f_{j+2}\right) - \frac{(2h)^5}{a60} f^{(iv)}(x_{j+1}) + \cdots$$

Again all error terms vanish if f the function error terms has 0 forth derivative

$$f^{(iv)}(x_{j+1}) = 0 \Rightarrow f'''(x_{j+1}) = C \Rightarrow f''(x_{j+1}) = Cx + D \Rightarrow$$

$$\Rightarrow f'(x_{j+1}) = \frac{Cx^2}{2} + Dx + E \Rightarrow f(x_{j+1}) = \frac{Cx^3}{6} + \frac{Dx^2}{2} + Ex + F$$

a cubic function $\square$

Problem 2

(a) $I \simeq h f(a+h)$

consider the Taylor expansion at the end point $[a+h]$, $\varepsilon = a + \frac{1}{2}h$

$$f(a+h) = f(\varepsilon) + \frac{h}{2} f'(\varepsilon) + \frac{h^2}{8} f''(\varepsilon) + \frac{h^3}{48} f'''(\varepsilon) + \cdots$$

$$\therefore f(\varepsilon) = f(a+h) - \frac{h}{2} f'(\varepsilon) - \frac{h^2}{8} f''(\varepsilon) - \frac{h^3}{48} f'''(\varepsilon) - \cdots$$

Substitute $f(\varepsilon)$ in the midpoint formula obtained in class to get

$$\int_a^{a+h} f(x)\,dx = h\left[f(a+h) - \frac{h}{2} f'(\varepsilon) - \frac{h^2}{8} f''(\varepsilon) - \cdots\right] + \frac{h^3}{24} f''(\varepsilon) + \cdots$$

$$= \underbrace{h f(a+h)}_{\text{aprox (a)}} \underbrace{- \frac{h^2}{2} f'(\varepsilon) + h^3\left(\frac{1}{24} - \frac{1}{8}\right) f''(\varepsilon) + \cdots}_{\text{error terms}}$$

$\therefore$ This aproximation is order $O(h^2)$ on a single panel, and $O(h)$ over the entire interval

(b) $I \simeq h f(a+h) - \frac{h^2}{2} f'(a)$

Consider the Taylor expansion for $f'(x)$ arround ε as before

$$f'(a) = f'(\varepsilon) - \frac{h}{2} f''(\varepsilon) + \frac{h^2}{8} f'''(\varepsilon) - \cdots$$

$$\therefore f'(\varepsilon) = f'(a) + \frac{h}{2} f''(\varepsilon) - \frac{h^2}{8} f'''(\varepsilon) + \cdots$$

Substitute into expansion arround $a+h$ from part a

$$f(\varepsilon) = f(a+h) - \frac{h}{2}\left[f'(a) + \frac{h}{2} f''(\varepsilon) - \frac{h^2}{8} f'''(\varepsilon) + \cdots\right] - \frac{h^2}{8} f''(\varepsilon) + \cdots$$

$\therefore \quad f(\varepsilon) = f(a+h) - \frac{h}{2} f'(a) - h^2 \left[\frac{1}{2} + \frac{1}{8}\right] f''(\varepsilon) - \cdots$

substitute $f(\varepsilon)$ in the midpoint formula from the notes:

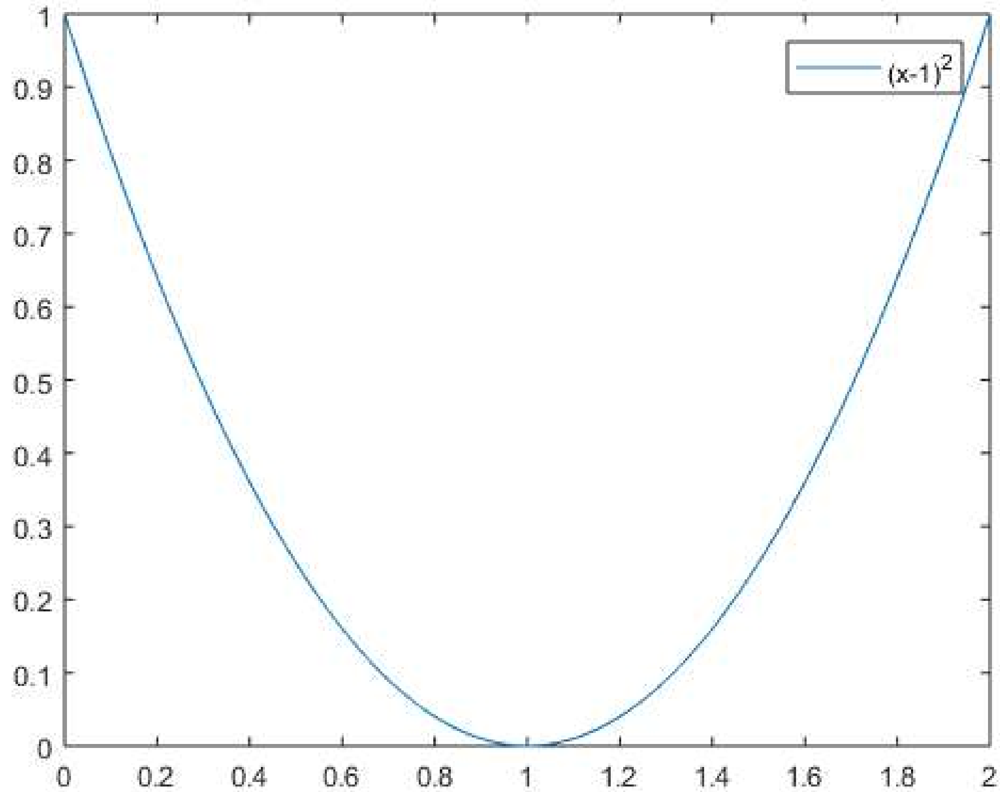
$$\int_a^{a+h} f(x)\,dx = \underbrace{h\, f(a+h) - \frac{h^2}{2} f'(a)}_{\text{aprox}} \underbrace{+ h^3\left[\frac{5}{8} + \frac{1}{24}\right] f''(\varepsilon) + \cdots}_{\text{error terms}}$$

$\therefore$ This apoximation is order $O(h^3)$ for a single panel, and $O(h^2)$ for the entire interval

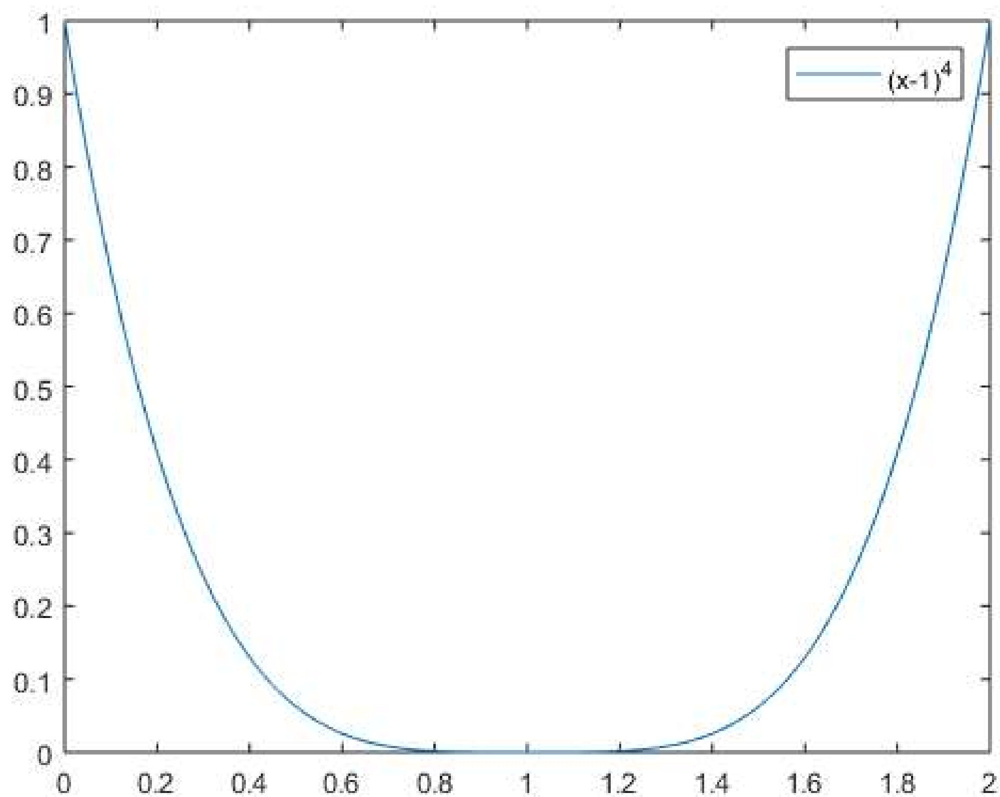
Problem 3

```
syms x
digits(16)

X = linspace(0, 2);

for n = [2, 4, 8]

    figure
    plot(X,f(X,n))
    legend(strcat('(x-1)^', num2str(n)))

    [x, k] = Newton(@f, n, @fp, 1.1, 1.e-8);
    fprintf('Testing (x-1)^%i near 1.1 gives root at %f after %i iterations \n', n, x, k);

end

function [ y ] = f (x, n)
    y = (x-1).^n;
end

function [ y ] = fp (x, n)
    y = n*(x-1).^(n-1);
end

function [ x_zero, itr ] = Newton (f, n, fprime, x0, tol)

    x(1) = x0; % must supply initial guess
    y(1) = f(x(1), n);
    yprime(1) = fprime(x(1), n);
    k = 1;
    stop = 0;
    while ~stop
        x(k+1) = x(k)-y(k)/yprime(k);
        y(k+1) = f(x(k+1), n);
        if abs(x(k+1)-x(k)) <= tol
            itr = k+1; % total number of iterations
            x_zero = x(k+1); % zero of function
            stop = 1;
        end
        yprime(k+1) = fprime(x(k+1), n);
        k = k+1;
    end

end
```

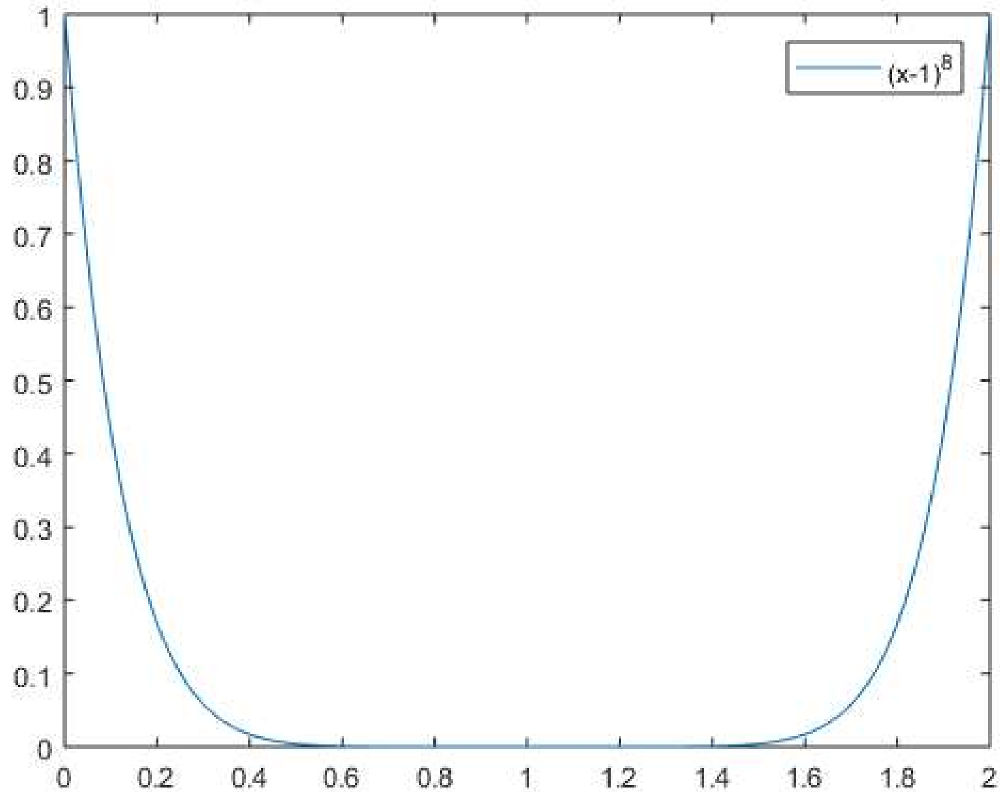
```
Testing (x-1)^2 near 1.1 gives root at 1.000000 after 25 iterations
Testing (x-1)^4 near 1.1 gives root at 1.000000 after 54 iterations
Testing (x-1)^8 near 1.1 gives root at 1.000000 after 108 iterations
```



(x-1)^2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2



(x-1)^4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2



Published with MATLAB® R2017a

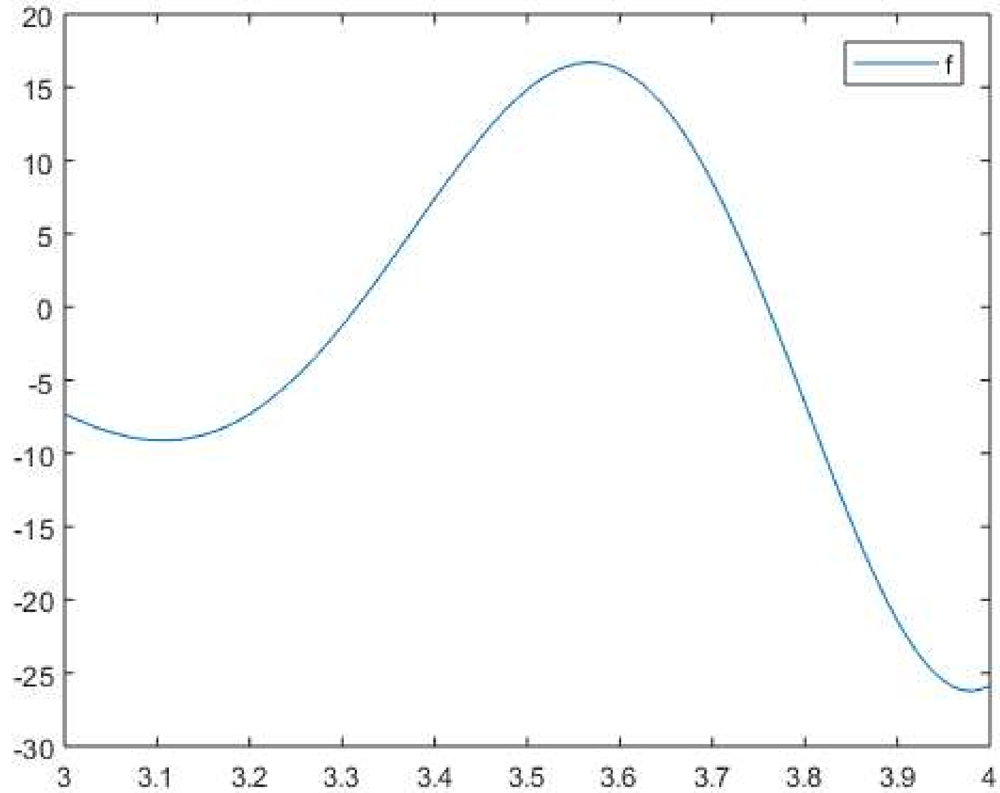
As we can see, as the value of the exponent increases (so long as it is even), the plot of the function looks more and more flat. It is therefore harder to see where the function touches the x axes. This also means that the Newton's Raphson Method will take longer to arrive to an answer since every step will be subtracting a fraction smaller and smaller denominator. We can see this in the value of k provided by the algorithm.

When n = 2 it only took 25 iterations versus twice as much for n = 4 and more thatn 4 times longer for n = 8. Because our condition was for thwo succesive x's to be closer together than the tolerance, it makes sense that flatter functions take longer tu converge.

Problem 4

```
syms x
digits(16)

X = linspace(3, 4);

figure
plot(X,f(X))
legend('f')

[x, k] = Secant(@f, 3.1, 3.5, 1.e-8);
fprintf('Testing f gives root at %f after %i iterations \n', x, k);

function [ y ] = f (x)
    y = cos(x.^2).*(x-1).^3;
end


function [ x_zero, itr ] = Secant (f, x0, x1, tol)

    x(1) = x0; % must supply initial guesses
    x(2) = x1;

    y(1) = f(x(1));
    y(2) = f(x(2));

    k = 2;
    stop = 0;
    while ~stop
        x(k+1) = x(k)-y(k)*(x(k)-x(k-1))/(y(k)-y(k-1));
        y(k+1) = f(x(k+1));
        if abs(x(k+1)-x(k)) <= tol
            itr = k+1; % total number of iterations
            x_zero = x(k+1); % zero of function
            stop = 1;
        end
        k = k+1;
    end

end
```

```
Testing f gives root at 3.315958 after 8 iterations
```



f
20
15
10
5
0
-5
-10
-15
-20
-25
-30
3
3.1
3.2
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

$$D_h(x_0) = \frac{f(x_0) - f(x_0 - h)}{h}$$ accuracy h

let $f(x) = \ln(x^2+1)$, $x_0 = 1$ find derivative starting $h = 1$

obtain formula for D_{lcm} 4 rows

$$\tilde{D}_{11} = \frac{f(x_0) - f(x_0 - h)}{h} \Rightarrow D = \tilde{D}_{11} + k_1 h + k_2 h^2 + k_3 h^3 + \dots$$

$$\tilde{D}_{11} = D - kh - k_2 h^2 - k_3 h^3 + \dots$$

let $h_1 = h/2$

$$\tilde{D}_{21} = D - \frac{kh}{2} - \frac{k_2 h^2}{4} - \frac{k_3 h^3}{8}$$

$$2\tilde{D}_{21} - \tilde{D}_{11} = D - \frac{k_2 h^2}{2} + k_2 h^2 - \frac{k_3 h^3}{4} + k_3 h^3 =$$

$$\tilde{D}_{22} = D + \frac{k_2 h^2}{2} + \frac{3}{4} k_3 h^3$$

next row

---

let $h_2 = h_1/2 = h/4$

$$\tilde{D}_{31} = D - k_1 \frac{h}{4} - \frac{k_2 h^2}{16} - \frac{k_3 h^3}{64}$$

$$2\tilde{D}_{31} - \tilde{D}_{21} = D - \frac{k_2 h^2}{8} + \frac{k_2 h^2}{4} - \frac{k_3 h^3}{32} + \frac{k_3 h^3}{8}$$

$$\tilde{D}_{32} = D + \frac{k_2 h^2}{8} + \frac{3}{32} k_3 h^3$$

$$\frac{4\tilde{D}_{32} - \tilde{D}_{22}}{3} = D + \frac{1}{8} k_3 h^3 - \frac{1}{4} k_3 h^3$$

$$\tilde{D}_{33} = D - \frac{1}{8} k_3 h^3$$

$$\tilde{D}_{k,m} = \frac{2^{m-1}\,\tilde{D}_{k,m-1} - \tilde{D}_{k-1,m-1}}{2^{m-1} - 1} \quad \text{with } O(h^k)$$

Problem 5

```
syms x

fprintf('a) \n\n')

ans = Dn (@f, 1, 1, 4, 4);

fprintf('\nb)\n\n %f percent error in final answer \n', percentErr(1, ans));


function [ err ] = percentErr (act, com)
    err = 100 * abs(com - act) / act;
end

function [ y ] = f (x)
    y = log(x^2 + 1);
end

function [ d ] = D ( f, x0, h )

    d = ( f(x0) - f(x0 - h) ) / h;

end

function [ d ] = Dn ( f, x0, h, k, m )
    if m == 1
        d = D(f, x0, h/(2^(k-1)));
    else
        d = ( 2^(m-1) * Dn(f, x0, h, k, m-1) - Dn(f, x0, h, k-1, m-1) ) / (2^(m-1) - 1);
    end

    fprintf('D_%i,%i = %f \n', k, m, d );
end
```

```
a)

D_4,1 = 0.997140
D_3,1 = 0.987440
D_4,2 = 1.006839
D_3,1 = 0.987440
D_2,1 = 0.940007
D_3,2 = 1.034873
D_4,3 = 0.997494
D_3,1 = 0.987440
D_2,1 = 0.940007
D_3,2 = 1.034873
D_2,1 = 0.940007
D_1,1 = 0.693147
D_2,2 = 1.186867
D_3,3 = 0.984209
D_4,4 = 0.999392
```

```
b)

  0.060815 percent error in final answer
```