```
clc
clear clf
close all

x = [1,2,3]
y = [1;2;3]

A = x * y % this is inner product
B = y * x % but this is outer product

B .^ 3 % this gives element wise multiplication

sin(cos(B)) % function composition is intuitive

x(1,3) % one based indexing, returns 3
x(1:3) = 10 % can assign to slices, gives 10 10 10

(3:5)' % can use operations on intermediate expressions like the following

clc
clear clf
close all

7 * 9

7 + 9

5*3+4

100+100 % own example


factorial(3) % own example

%like this

log(10000)

log(3.2 * 8.5)

log(3.2) + log(8.5)

pi

a = 3*9;

str = 'Hello, world!';

disp(str);

run ../lab0/myliquid.m
```

```
for var = 1:4
    disp(var)
end

% ranges are vectors

% own examples
x = [1,2,3,4,5]
for i = x
    disp(x)
end

if pi == 3
    disp('hello')
else
    disp('goodbye')
end

clc
clear clf
close all

type expTaylorPoly.m;

x = linspace(-1, 1, 2019);
T = expTaylorPoly(x, 10);

% what happens when empty?
% own example
linspace(0, 0, 100);


% x^2 this doesnt work, bad dimensions


% dot for elementwise
x.^2;

plot(x, T)
plot(x, T, x, x)
loglog(x, T) % own example, try loglog plot

type expHorner.m;
type plotOfExp.m;

evalc plotOfExp;
```

```
x =

     1     2     3


y =
```

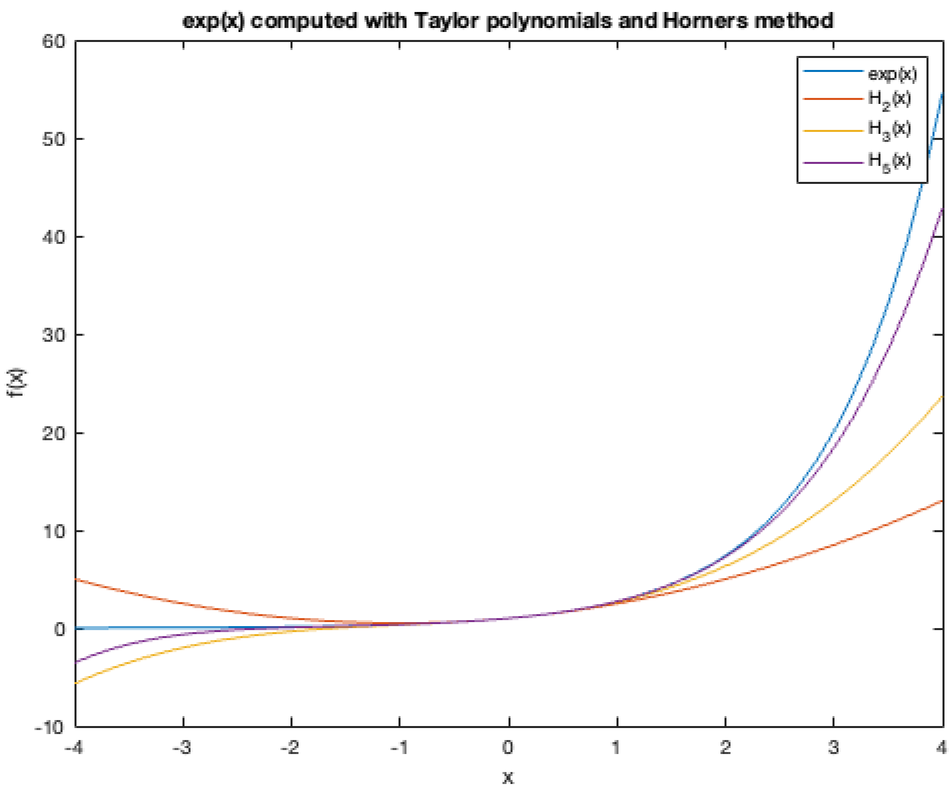
```
     1
     2
     3


A =

    14


B =

     1     2     3
     2     4     6
     3     6     9


ans =

     1     8    27
     8    64   216
    27   216   729


ans =

    0.5144   -0.4042   -0.8360
   -0.4042   -0.6081    0.8193
   -0.8360    0.8193   -0.7902


ans =

     3


x =

    10    10    10


ans =

     3
     4
     5


ans =

    63
```

```
ans =

    16


ans =

    19


ans =

   200


ans =

     6


ans =

    9.2103


ans =

    3.3032


ans =

    3.3032


ans =

    3.1416

Hello, world!
### Calculation of the volume of liquid in a spherical tank
The tank has radius r=  1.0 meters
The liquid depth has h=  0.7 meters

V =

    1.1802

The volume of liquid is V=      1.18 cubic meters
### END OF VOLUME CALCULATION\n
     1

     2
```

```
     3

     4


x =

     1     2     3     4     5

     1     2     3     4     5

     1     2     3     4     5

     1     2     3     4     5

     1     2     3     4     5

     1     2     3     4     5

goodbye
```

```
% ==========================================================================
% AUTHOR ..... David
% UPDATED .... 2024.01.18
%
% Evaluate the truncated Taylor series for exp(x) about the point x0 = 0
%
% INPUT
%   x .... Vector of values to evaluate the Taylor polynomial at
%   n .... Integer of last term to evaluate in Taylor polynomial
%
% OUTPUT
% T : Evaluated Taylor polynomial at points given by x degree n
% ==========================================================================

function T = expTaylorPoly(x, n)
    % Initialize sum as 0
    T = 0;
    % Loop over terms in series
    for k = 0:n
        T = T + x.^k ./ factorial(k);
    end
end

% ==========================================================================
% AUTHOR ..... David Tran
% UPDATED .... 2024.01.18
%
% Evaluate exp(x) about the point x0 = 0 using Horner's method
%
% INPUT
%   x .... Vector of values to evaluate the Taylor polynomial at
%   n .... Integer of last term to evaluate in Taylor polynomial
%
```

```
% OUTPUT
% H : Evaluated Taylor polynomial at points given by x degree n
% =========================================================================

function H = expHorner(x, n)
    H = 1
    for k = n : -1 : 1
        H = 1 + x .* H ./ k
    end
end

xs = linspace(-4, 4, 50);
expCurve = exp(xs);
h2Curve = expHorner(xs, 2);
h3Curve = expHorner(xs, 3);
h5Curve = expHorner(xs, 5);

plot(xs, expCurve, xs, h2Curve, xs, h3Curve, xs, h5Curve);
legend('exp(x)', 'H_2(x)', 'H_3(x)', 'H_5(x)');
title('exp(x) computed with Taylor polynomials and Horners method');
xlabel('x')
ylabel('f(x)')
```



Published with MATLAB® R2023b