HW #3

Math 501

CSUF Spring 2007.

Nasser Abbasi

problems:

section 3.3 # 4, 5, 6

Section 3.4 # 4, 5, 10, 12, 13, 29, 40

includes Computer Problems.

HW # 3
Computer Assignments

1. Taylor approximation 15/15

2. Bisection and secant methods for 1-D 10/10

3. Horner Method. 10/10

Note: Matlab Source Code if needed can be Found at This temporary folder:

http:// 12000.org/tmp/021407

due Wed.

Name: Nasser Abbasi

Math 501 – Numerical Analysis & Computation – Dr. Lee – Spring 2007

Computer Assignment 02/05/2007

Given f(x) = exp(x), the Taylor approximation for f(x) for x near 0 can be found as

$$P_N(x) = \sum_{k=0}^{N} \frac{1}{k!} x^k \quad \text{for} \quad N \geq 1.$$

1) Write a MATLAB function that takes in x and N and computes $P_N(x)$.

2) Write a MATLAB for-loop program that uses the subplot command to plot $P_N(x)$ for $N = 1, \ldots, 6$ and $x \in [-1, 1]$.

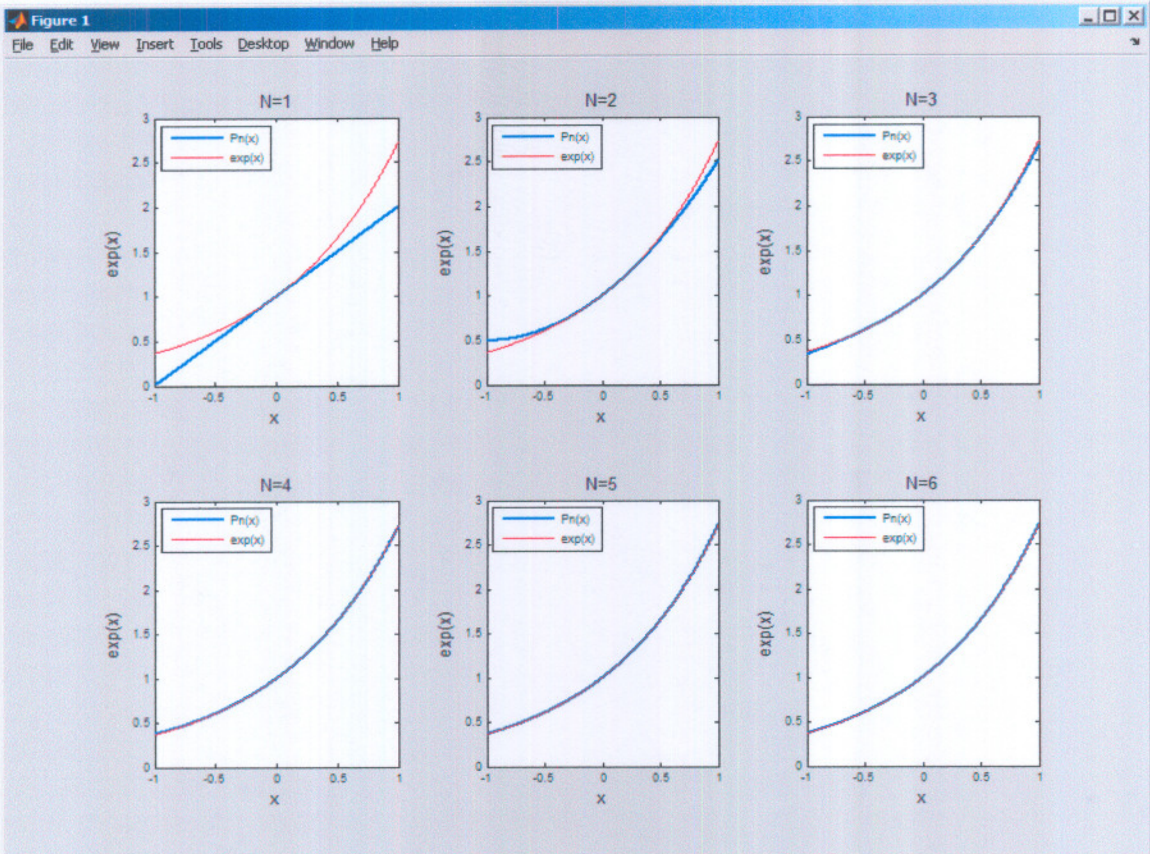
3) For each N, plot the absolute and relative errors

Part #1 of first Computer Assignment

```
function pn=nma_Taylor(x,numberOfTerms)

%
% function pn=nma_Taylor(x,N)
% Taylor approximation for exp(x) for N terms
%

%INPUT:
%  x: the x-value to estimate exp(x) at
%  numberOfTerms: number of terms in tayler series to us
%
%OUTPUT:
% pn:  The estimated value of exp(x) using numberOfTerms
%

% By Nasser Abbasi. HW3 computer assignment.
% Math 501, CSUF.  Computer assignment 2/5/07
% PART (1)
%

%EXAMPLE RUNS
% >> pn=nma_Taylor(10,30)
% pn =
%     2.202646403625892e+004
%
% compare to actual exp()
% >> exp(10)
% ans =
%     2.202646579480672e+004
% >>


if nargin < 2
   error 'number of arguments must be 2'
end

if numberOfTerms<1
   error 'numberOfTerms must be >=1'
end

if ~ ( isnumeric(x) && isnumeric(numberOfTerms) )
    error 'input parameters must be numeric'
end

pn=0;

for k = 0 : numberOfTerms
    pn = pn + x^k/factorial(k);
end
```

Part #2, first Computer Assignment

```
%
% Part(2) solution to computer assignment, 2/05/2007.
% Nasser Abbasi. Math 501, CSUF, spring 2007
%
% Write a script that uses a for loop that uses subplot to plot
% Pn(x) for N=1..6 and x in [-1,1]. Use the function written in
% part(1) of the assignment. see nma_Taylor.m

%
%PLot the actual exp(x) using RED line, and the approximated exp(x)
%using BLUE line

clear all; close all;

currentPlotNumber = 1;
MAX_ITERATIONS    = 6;

x=linspace(-1,1,1000);

figure(1);

for n = 1:MAX_ITERATIONS

    subplot(2,3,currentPlotNumber);

    y = nma_Taylor(x,n);

    plot(x,y,'LineWidth',2);
    hold on;
    plot(x,exp(x),'r');

    currentPlotNumber = currentPlotNumber + 1;
    xlabel('x'); ylabel('exp(x)');
    title(sprintf('N=%d',n));
    legend('Pn(x)','exp(x)','Location','NorthWest');
    set(gca,'FontSize',7);
end
```



Notice: As N increases, approximation (blue Color) approaches actual value (Red Color).
at N=6 There is almost No visible difference.

Part # 3 First Computer Assignment

```
%
% Part(3) solution to computer assignment, 2/05/2007.
% Nasser Abbasi. Math 501, CSUF, spring 2007
%
% Write a script that uses a for loop that uses subplot to plot
% absolute and relative error. part(3) of the assignment. see
nma_Taylor.m

clear all; close all;

currentPlotNumber = 1;
MAX_ITERATIONS    = 6;

x=linspace(-1,1,1000);

for n = 1:MAX_ITERATIONS

    figure;

    approxValue = nma_Taylor(x,n);
    trueValue   = exp(x);

    absError      = abs(trueValue-approxValue);
    relativeError = abs(trueValue-approxValue)./abs(trueValue);

    subplot(1,2,1);
    plot(x,absError,'LineWidth',2);
    title(sprintf('N=%d, Abs error',n)); xlabel('x'); ylabel('error');

    subplot(1,2,2);
    plot(x,relativeError,'r');
    title(sprintf('N=%d, Rel error',n)); xlabel('x'); ylabel('error');

end
```

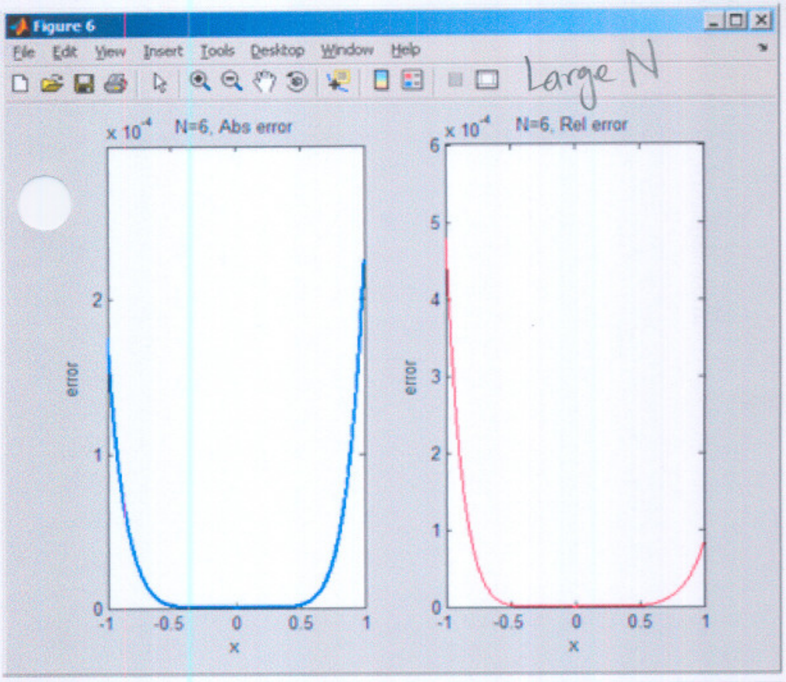


Figure 6
File Edit View Insert Tools Desktop Window Help
Large N
x 10^-4 N=6, Abs error
x 10^-4 N=6, Rel error
error
x

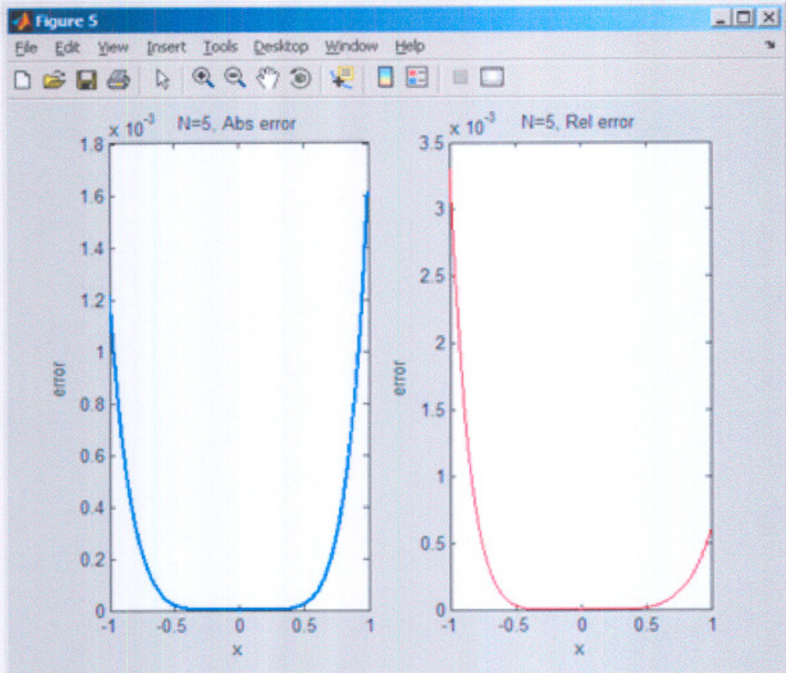


Figure 5
File Edit View Insert Tools Desktop Window Help
x 10^-3 N=5, Abs error
x 10^-3 N=5, Rel error
error
x

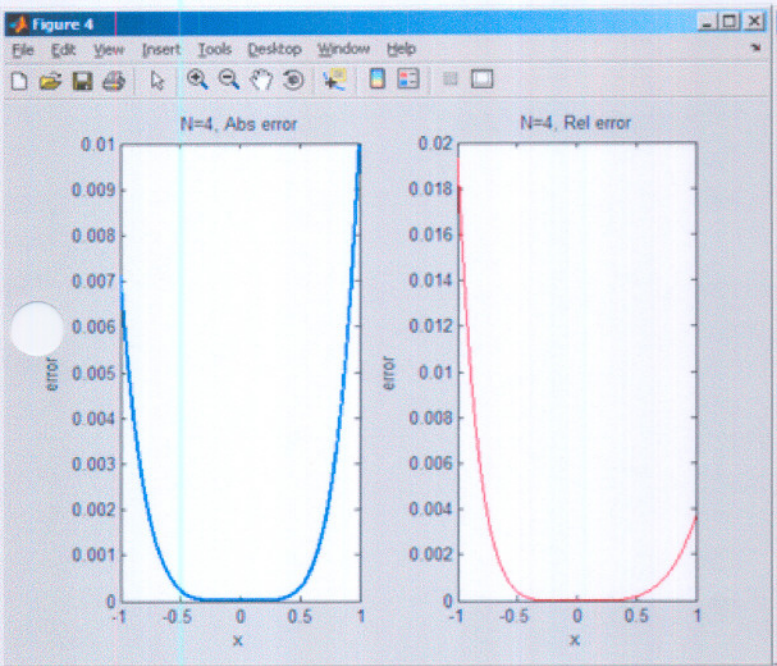


Figure 4
File Edit View Insert Tools Desktop Window Help
N=4, Abs error
N=4, Rel error
error
x

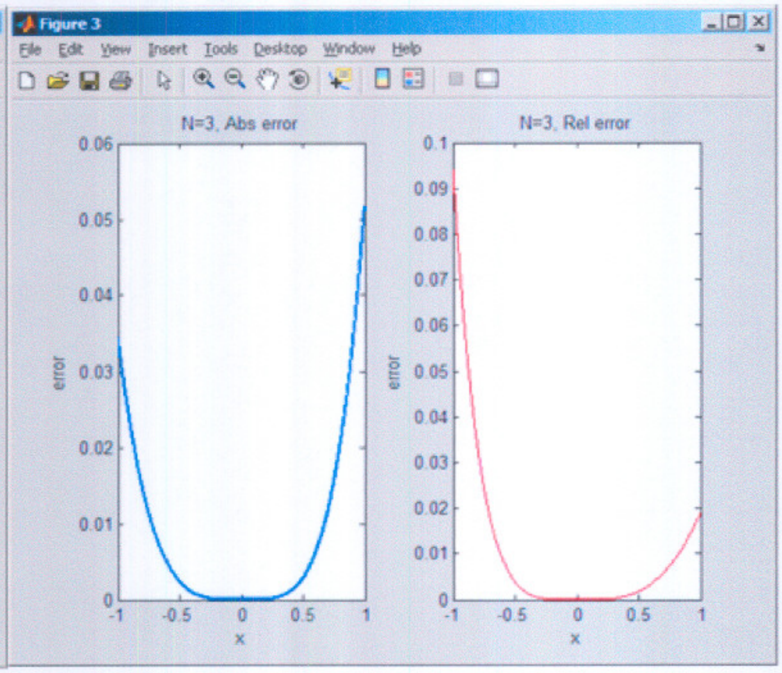


Figure 3
File Edit View Insert Tools Desktop Window Help
N=3, Abs error
N=3, Rel error
error
x

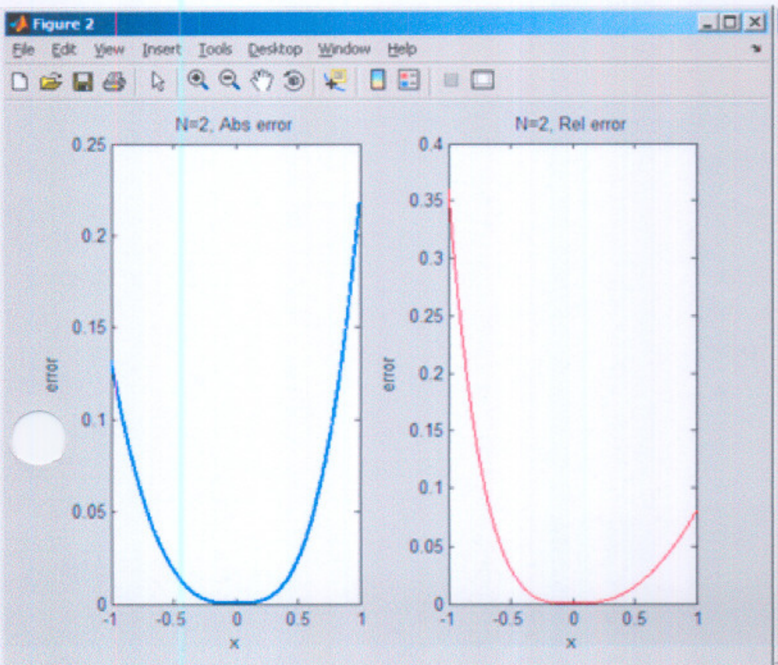


Figure 2
File Edit View Insert Tools Desktop Window Help
N=2, Abs error
N=2, Rel error
error
x

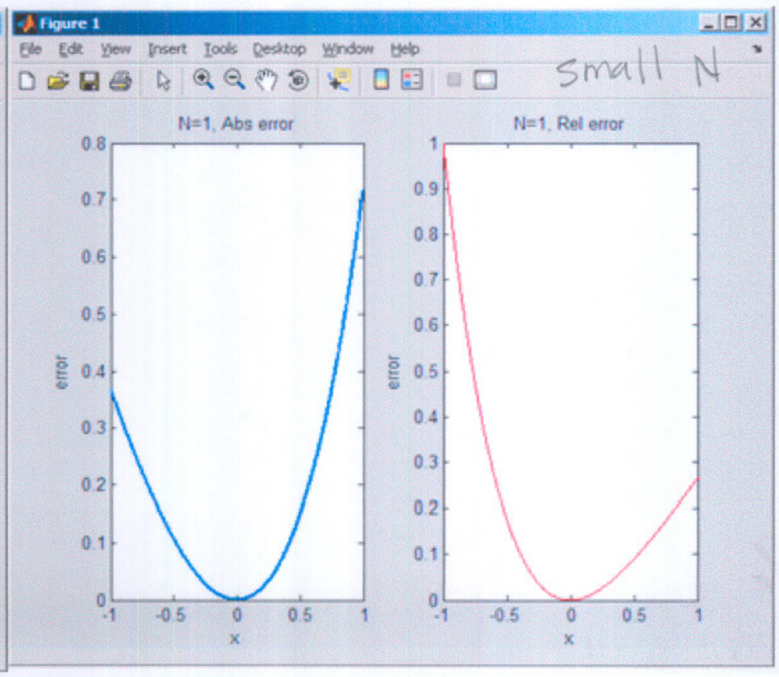


Figure 1
File Edit View Insert Tools Desktop Window Help
Small N
N=1, Abs error
N=1, Rel error
error
x

Second Computer Assignment. bisection

```
function [estimatedRoot, yAtEstimatedRoot, numIterationsUsed]= ...
    nma_bisection(theFunc, leftPoint, rightPoint, xErrorTol, yErrorTol)
%
% function [estimatedRoot, yAtEstimatedRoot, numIterationsUsed]=
%    nma_bisection(theFunc, leftPoint, rightPoint, xErrorTol, yErrorTol)
%
% 1-D bisection method
%
% This functions tries to find a root for a 1-D function bracketed between
% 2 points using the bisection method.

% INPUT:
%   theFunc:    Handle to the function whose root to be found
%   leftPoint:  value of the left point of the interval (the 'a' point)
%   rightPoint: value of the right point of the interval (the 'b' point)
%   xErrorTol : x-tolerance
%   yErrorTol : y-tolerance
%
%OUTPUT:
%   estimatedRoot: value of the estimate of the root
%   yAtEstimatedRoot  : value of the function at the estimated root
%   numIterationsUsed : number of iterations used
%
%Written by: Nasser Abbasi, feb 13,2007.
%            Part of HW3. Math 501, CSUF.
%
%
%

% EXAMPLE RUNS
%
% EXAMPLE 1:
% >> [c,y,n]=nma_bisection(@sin , -0.1, 0.3, 0.001, 0.001)
% c =
%    -6.938893903907228e-018
% y =
%    -6.938893903907228e-018
% n =
%      2
%
% EXAMPLE 2:
% >> [c,y,n]=nma_bisection(@(x) x^2+2*x-1 , 0.1, 0.5, 0.001, 0.001)
% c =
%    0.41562500000000
% y =
%    0.00399414062500
% n =
%      8
%
% compare above answer c to fzero answer:
% >> fzero(@(x) x^2+2*x-1 , -.1)
% ans =
%    0.41421356237310
```

example runs

```
TRUE  = 1;
FALSE = 0;

maxIterations = (log(  (rightPoint - leftPoint)/ xErrorTol ))/log(2) -1;
maxIterations = ceil(maxIterations);

n = 1;
rootFound = FALSE;

while n < maxIterations  &&  ~rootFound

   c  = (leftPoint+rightPoint)/2;
   fc = theFunc(c);

   if abs(fc) < yErrorTol
      rootFound = TRUE;
   else
      if sign(fc) == sign(theFunc(leftPoint))
         leftPoint = c;
      else
         rightPoint = c;
      end

      n = n + 1;
   end

end

estimatedRoot     = c;
yAtEstimatedRoot  = theFunc(c);
numIterationsUsed = n;
```

second computer Assignment. Secant

```
function [estimatedRoot, yAtEstimatedRoot, numIterationsUsed]= ...
    nma_secant(theFunc, a, b, xErrorTol, yErrorTol, maxIterations)
%
% function [estimatedRoot, yAtEstimatedRoot, numIterationsUsed]= ...
%     nma_secant(theFunc, a, b, xErrorTol, yErrorTol, maxIterations)
%
% 1-D secant method
%
% This functions tries to find a root for a 1-D function using the secant
% method.

% INPUT:
%   theFunc:    Handle to the function whose root to be found
%   a:  value of first of the initial point that secant method requires
%   b:  value of second of the initial point that secant method requires
%   xErrorTol : x-tolerance
%   yErrorTol : y-tolerance
%   maxIterations: max iterations allowed
%
%OUTPUT:
%   estimatedRoot: value of the estimate of the root
%   yAtEstimatedRoot  : value of the function at the estimated root
%   numIterationsUsed : number of iterations used
%
%Written by: Nasser Abbasi, feb 13,2007.
%             Part of HW3. Math 501, CSUF.
%
%
%

% EXAMPLE RUNS
%
% EXAMPLE 1:
% >> [c,y,n]=nma_secant(@(x) x^3-sinh(x)+4*x^2+6*x+9 ,7,8,0.0001,0.0001,10)
% c =
%    7.11306342932610
% y =
%    -2.875063387364207e-008
% n =
%       6

%EXAMPLE 2
%
% >> [c,y,n]=nma_secant(@(x) x^2+2*x-1 , 0, 1, 0.0001, 0.0001,10)
% c =
%    0.41421143847487
% y =
%    -6.007286838860537e-006
% n =
%       5
```

example Runs

```
TRUE  = 1;
FALSE = 0;

n = 1;
rootFound = FALSE;

fa = theFunc(a);
fb = theFunc(b);

while n < maxIterations  &&  ~rootFound

    if abs(fa)>abs(fb)
        tmp = a;
        a = b;
        b = tmp;

        tmp = fa;
        fa = fb;
        fb = tmp;
    end

    s  = (b-a)/(fb-fa);
    b  = a;
    fb = fa;
    a  = a - fa*s;
    fa = theFunc(a);

    if abs(fa) < yErrorTol |  abs(b-a)<xErrorTol
       rootFound = TRUE;
    end

    n = n + 1;
end

estimatedRoot     = a;
yAtEstimatedRoot  = fa;
numIterationsUsed = n;
```

Name: Nasser Abbasi
Math 501 – Numerical Analysis & Computation – Dr. Lee – Spring 2007

Computer Assignment 02/12/2007

1) Write a MATLAB program that takes in the coefficients of a polynomial $P(z)$ and a specific point z_0 and outputs the values $P(z_0)$ and $P'(z_0)$.
(Print out the matlab code with your name on it)

2) Test the program with the polynomial $P(z) = 9z^4 - 7z^3 + z^2 - 2z + 5$ and $z_0 = 2$.
(Write the executable command along with the output)

```
function [pz,pzd]=nma_horner(a,z0)
%
% function [pz,pdz]=nma_horner(a,z0)
%
% evaluate P(z0) and P'(z0) from coefficients of polynomial a, at the
% point z0

%INPUT:
%  a: vector that contains the polynomial coeff in this order
%     [a0 a1 ..... an]
%  z0: the value where to evaluate the Polynomial at.
%
% by Nasser Abbasi. Computer assignment 02/12/07
% Math 501. CSUF
%
%
% EXAMPLE RUN:
% Test program on P(z)=9*z^4-7*z3+z^2-2*z+5 at z=2
%
% >> [pz0,pdz0]=nma_horner([5,-2,1,-7,9],2)
%
% pz0 =
%     93
% pdz0 =
%    206
% >>
```

Part #2
executable command with the output.

```
if nargin < 2
   error 'number of arguments must be 2'
end

if length(a)==0
   error 'coefficients array is empty'
end

if ~ ( isnumeric(z0) && isnumeric(a) )
    error 'input parameters must be numeric'
end

%first call to find P(z0);
b=myHorner(a,z0);
pz=b(1);

%Call again call to find P'(z0);
b=myHorner(b(2:end),z0);
pzd=b(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% internal function to evaluate
% a Horner row
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function b=myHorner(a,z0)
n = length(a);
b = zeros(n,1);
b(n) = a(n);
for k = n-1:-1:1
    b(k) = a(k)+b(k+1)*z0;
end
```

Verification using CAS:

```
In[1]:= p[z_] := 9z^4 - 7z^3 + z^2 - 2z + 5;
        p[z] /. z→2

Out[2]= 93

In[3]:= D[p[z], z] /. z→2

Out[3]= 206
```