1.(a) Write three MATLAB functions to computes the coefficients of polynomial interpolant by the three methods: Vandermonde approach, Newton representation, and divided differences with input arguments x and y:

+ Vandermonde approach (InterpV.m):

function a = InterpV(x,y)

n = length(x);

V = ones(n,n);

for j=2:n

V(:,j) = x.\*V(:,j-1);

end

a = V\y;

+ Newton representation (InterpN.m):

function c = InterpN(x,y)

n = length(x);

for k=1:n-1

y(k+1:n) = (y(k+1:n) - y(k)) ./ (x(k+1:n) - x(k));

end

c=y;

+ Divided differences (InterpN2.m):

function c = InterpN2(x,y)

n = length(x);

for k=1:n-1

y(k+1:n) = (y(k+1:n) - y(k:n-1)) ./ (x(k+1:n) - x(1:n-k));

end

c=y;

(b) Write two MATLAB functions to evaluates the values of any polynomial function p(x), which is represented by the ascending form



or by the Newton's form using Horner's algorithm. The input arguments are the coefficients a and the x-coordinates z which we want to evaluate:

+ Vandermonde approach (HornerV.m):

function pVal = HornerV(a,z)

n = length(a);

pVal = a(n)\*ones(size(z));

for k=n-1:-1:1

pVal = z.\*pVal + a(k);

end

+ Newton’s form (HornerN.m):

function pVal = HornerN(c,x,z)

n = length(c);

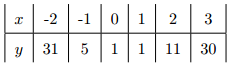
pVal = c(n)\*ones(size(z));

for k=n-1:-1:1

pVal = (z-x(k)).\*pVal + c(k);

end

(c) Write a MATLAB program (script file) to interpolates the given data to obtain



a polynomial, say p(x), by using the functions, which you have created in part (a) for the coefficients, and then to evaluate the values on [3; 5] for 65 points by using the functions, which you have created in part (b), and plot the curves by the two ways, Vandermonde approach and Newton representation. Also answer the values of p(4) and p(5):

x = [-2 -1 0 1 2 3]';

y = [35 5 1 1 11 30]';

a = InterpV(x,y);

c = InterpN(x,y);

z = linspace(-3,5,65);

pValV = HornerV(a,z);

pValN = HornerN(c,x,z);

plot(z,pValV,'--',z,pValN,'\*');

i4 = find(z==4);

i5 = find(z==5);

fprintf('p(4) = %.2f\n', pValV(i4));

fprintf('p(5) = %.2f\n', pValV(i5));

(d) Do the problem P2.2.1 (in textbook) and check your answer by the above example: Write a MATLAB function a = N2V(c,x), where c is a column n-vector, x is a column (n-1)-vector and a is a column n-vector, so that if



then



In other words, N2V converts the Newton representation to the Vandermode representation:

+ N2V function:

function a = N2V(c,x)

% Find y values correspond to x values by evaluating the Newton interpolant on z=x

y = HornerN(c,x,x);

% Computes the Vandermonde polynomial interpolant

a = InterpV(x,y);

+ Script to test example in 1.(c) (ShowN2V.m):

x = [-2 -1 0 1 2 3]';

y = [35 5 1 1 11 30]';

disp('Computes the coefficients of polynomial interpolant by the method: Newton representation');

c = InterpN(x,y)

disp('Computes the coefficients of polynomial interpolant by the method: Vandermonde approach');

a1 = InterpV(x,y)

disp('Converts the Newton representation to the Vandermode representation by using function N2V(c,x)');

a2 = N2V(c,x)

2. Plot the function y = sin(x) for x ∈ [-π,π] and their Taylor polynomials



on the same figure with different line types and different colors, also it should have the labels, title, and legend. (Note: You should write S4(x) by yourself):

n=100;

x=linspace(-pi,pi,n);

s = sin(x);

s2 = x - x.^3/6;

s3 = s2 + x.^5/120;

s4 = s3 - x.^7/5040;

plot(x,s,'b',x,s2,'\*g',x,s3,'.-r',x,s4,'+c')

title('y = sin(x), x \in [-\pi, \pi]');

xlabel('x(Radians)');

ylabel('y');

legend('y = sin(x)', 'y = x - x^3 / 3!', 'y = x - x^3 / 3! + x^5 / 5!','y = x - x^3 / 3! + x^5 / 5! - x^7 / 7!')