Integration:

```
x = input('Enter xi values: ');
y = input('Enter yi values: ');
h = x(2)-x(1);
n = length(x);
if mod(n-1,2) == 0
 disp("Using Simpson's 1/3rd rule");
 I = y(1)+y(n);
 for i = 2:2:n-1
 I = I+4*y(i);
 end
 for i = 3:2:n-2
 I =I+2*y(i);
 end
 I = (h/3)*I;
else
 disp("Using Trapezoidal rule");
 I = 0;
 for i = 2:(n-1)
 I = I+y(i);
 end
 I = I*2+y(1)+y(n);
 I = (h/2)*I;
end
disp(["The integral result is: ", num2str(I)]);
```

```
%forward Interpolation
x = input('Enter xi values as an array: ');
y = input('Enter yi values as an array: ');
X = input('Enter X value: ');
h = x(2) - x(1);
n = length(x);
D = zeros(n);
for i = 1:n
 D(i,1) = y(i);
end
for j = 2:n
 for i = 1:(n - j + 1)
 D(i,j) = D(i + 1, j - 1) - D(i, j - 1);
 end
end
Y = y(1);
product = 1;
fact = 1;
for i = 1:(n - 1)
 product = product * (X - x(i)) / h;
 fact = fact * (1 / i);
 Y = Y + (product * fact * D(1, i + 1));
end
fprintf('Interpolated value Y = %f\n', Y);
```

```
%lagrange interpolation
x = input('Enter x values : ');
y = input('Enter y values : ');
X = input('Enter the y value to interpolate: ');
n = length(x);
Y = 0;
for i = 1:n;
 L=1;
 for j = 1:n;
 if i~=j
 L = L*(X - x(j)) / (x(i) - x(j));
 end
 end
 Y = Y + L * y(i);
end
disp(['Interpolated value is: ', num2str(Y)]);
```

```
%backward Interpolation
x = input('Enter xi values as an array: ');
y = input('Enter yi values as an array: ');
X = input('Enter X value: ');
h = x(2) - x(1);
n = length(x);
D = zeros(n);
for i = 1:n
 D(i,1) = y(i);
end
for j = 2:n
 for i = n:-1:j
 D(i,j) = D(i,j-1) - D(i-1,j-1);
 end
end
Y = y(n);
product = 1;
fact = 1;
for i = 1:(n - 1)
 product = product * (X - x(n - i + 1)) / h;
 fact = fact * (1 / i);
 Y = Y + (product * fact * D(n, i + 1));
end
fprintf('Interpolated value Y = %f\n', Y);
```

```
%Euler's Method
syms x
syms y
f(x,y) = input("Enter f(x,y)")
y1 = input("Enter y1 value")
x1 = input("Enter x1 value")
h = input("Enter Step Size")
X = input("Enter X value")
n = (X - x1)/h
x = x1:h:X
y = zeros(1, n+1)
y(1) = y1
for i = 2 : (n+1)
 y(i) = y(i-1) + h * f(x(i-1),y(i-1))
end
disp(y(n+1))
```

```
%Runge-Kutta 4th-order
syms x
syms y
f(x, y) = input("Enter f(x,y): ");
y1 = input("Enter y1 value: ");
x1 = input("Enter x1 value: ");
h = input("Enter Step Size: ");
X = input("Enter X value: ");
n = (X - x1)/h;
x = x1:h:X;
y = zeros(1, n+1);
y(1) = y1;
for i = 2:(n+1)
 k1 = h * f(x(i-1), y(i-1));
 k2 = h * f(x(i-1) + h/2, y(i-1) + k1/2);
 k3 = h * f(x(i-1) + h/2, y(i-1) + k2/2);
 k4 = h * f(x(i-1) + h, y(i-1) + k3);
 y(i) = y(i-1) + (k1 + 2*k2 + 2*k3 + k4) / 6;
end
disp(y(n+1));
```

Curve Fitting

```
A = [-6 -4 -2 0 2 4 6]
B = [-0.2 0.8 -1.6 3.5 3.1 4.9 3.9]
P = polyfit(A,B,6)
plot(A,B,'k')
hold on
A = min(A):2:max(A)
B = polyval(P,A)
plot(A,B,'r')
hold off
```