Bisection Method

1. Decide initial values for x1 and x2 and stopping criterion, E
2. Compute f1=f(x1) & f2=f(x2)
3. If f1\*f2>0 , x1 and x2 do not bracket the root & goto 7
4. Compute x0=(x1+x2)/2 and compute f0=f(x0)
5. If f1\*f2<0 then set x2=x0 , f(x1)=f(x0)

Else set x1=x0 , f1=f0

1. If absolute value of (x2-x1)/x2 is less than error E, then
   1. Root=(x1+x2)/2
   2. Write the value of root
   3. Go to step 7

Else

* 1. Goto 4

1. Stop

Newton-Rap son

1. Assign an initial guess to x, say x0 & precision E
2. Evaluate f(x0) & f’(x0)
3. Find the improved estimate of x0
   1. X1=x0-f(x0)/f’(x0)
4. Check for the accuracy of latest estimate
   1. Compare the relative approximate error with predefined value E
   2. If |(x1-x0)/x1|<E stop, otherwise continue
5. Replace x0 by x1 & repeat step 3 & 4

Secant Method

1. Input two initial guesses (say x0 & x1 ) & precision E
2. Evaluate f(x0) & f(x1)
3. Estimate the new value of the root as
   1. X2=x1-{f(x1)(x1-x0)}/{f(x1)-f(x0)}
4. Find the absolute relative approximation error Ea as
   1. Ea= |(x2-x1)/x2|
5. Compare the approximate relative approximate error |Ea| with pre-specified relative error tolerance E
   1. If Ea>E then x0=x1, f(x0)=f(x1) , x1=x2 go to step 3
   2. Else go to step 6
6. Stop

Fixed Point Iteration

1. Input initial guess (say x0) & error estimate (say P)
2. Convert f(x)=0 to the form x=g(x)
3. Estimate now value of the root x1 as
   1. X1=g(x0)
4. Find the absolute relative approximate error |Ea| as
   1. |Ea|=|(x1-x0)/x1|
5. Compare absolute relative approximate error Ea with the pre-specified relation error tolerance E
   1. If |Ea|>E then x0=x1 go to step 3
   2. else go to step 6
6. Stop

Multiple Root Using Newton Rap son Method

1. Input the degree & coefficient of polynomial
2. Input initial guess x0 & error limit E
3. While n>1
   1. Find the root using newton-Rap son algorithm say nth root is xr
   2. Divide the polynomial by x-x1 to get the polynomial of degree n-1
   3. Set x0=x1
   4. N=n-1
4. End while
5. 1st root = -a0/a1
6. Stop

Horners Method for Polynomial Evaluation

1. Enter the coefficients of the polynomial
2. Enter the value at when the polynomial need to evaluate say x0
3. Enter the degree of polynomial say n & set bn=an
4. While n>1
   1. Bn-1=an-1+x0\*bn
5. End while
6. Display the value of b0, which is the value of P(x0)
7. Stop

Langrange interpolation polynomial

1. Read the number of points say n
2. Read the point at which we need to compute the value x0
3. Read interpolating points
4. Calculate Li value as
   1. For i=0 to n-1
      1. For j=0 to n-1
      2. If (i!=j) then Li=Li\*((x-xj)/(xi-xj))
5. Compute the interpolation value as
   1. For i=0 to n-1
      1. V=v+fi\*Li
6. Display the value of v
7. Stop

Newton’s dividend difference Interpolation

1. Read the number of interpolating points say n
2. Read a value where we need to calculate interpolation say x
3. Read the given points
4. Assign the value of fx[xi] to dd[xi]
   1. For i=0 to n-1
      1. Dd[i]=fx[i]
5. Compute the dividend difference
   1. For i=0 to n-1
      1. For j=n-1 to i+1
      2. dd[j]=(dd[j]-dd[j-1])/(x[j]-x[j-i-1])
6. set v=0 & p=1
7. calculate the interpolation value
   1. for i=0 to n-1
      1. for j=0 to i-1 { p=p\*(x-x[j])}
   2. v=v+dd[i]\*p
   3. reset P=1
8. Display the value of v
9. Stop

Newton’s Forward Interpolation Polynomial

1. Read the number of points say n
2. Read the interpolating point at which we need to calculate the value say x
3. Read the given points
4. Calculate the first forward difference
   1. For i=0 to n-1
      1. Fd[i]=fx[i]
5. Compute other differences as
   1. For i=0 to n-1
      1. For j=n-1 to i+1
      2. Fd[j]=fx[j]-fx[j-1]
6. Set xv=fx[0], p=1
7. Calculate the interpolating values as
   1. For i=1 to n-1
   2. P=1
      1. For k=1 to i-1 { p=p\*(s-k+1)}
   3. Xv=xv+(p\*fd[i])/i!
8. Display the interpolating value xv

Newton’s backward interpolation Polynomial

1. Read the number of points say n
2. Read the value at which we need to calculate interpolation value say xp
3. Read the n data points say x[i] & fx[i]
4. Compute h=x[1]-x[0] & s=(xp-x[n-1])/h
5. Compute first backward difference as
   1. For i=0 to n-1
      1. Bd[i]=fx[i]
6. Compute 2nd to nth backward difference as
   1. For i=n-1 to 1
      1. For j=0 to i-1
      2. Bd[j]=bd[j+1]-bd[j]
7. Set v=bd[n-1] , set p=1
8. Compute interpolation value
   1. For i=1 to n-1
      1. For j=1 to i
      2. P=p\*(s+j-1)
      3. End for
   2. V=v+(bd[n-i-1]\*p)/i!
   3. Reset p=1
   4. End for
9. Display v
10. Stop

Linear Regression

1. Read the number of data points, say n
2. Read data points say x[i] & y[i]
3. Calculate summation value
   1. For i=0 to n-1
      1. Sx=sx+x[i]
      2. Sy=sy+y[i]
      3. Sxy=sxy+x[i]\*y[i]
      4. Sx2=sx2+x[i]\*x[i]
   2. End for
4. Calculate the values of parameter
   1. b=((n\*sxy)-(sx+sy))/((n\*sx2)-(sx)^2)
   2. a=(sy/n)-b\*(sx/n)
5. Display the equation a+bx
6. Stop

Exponential Regression Analysis

1. Read the number of points, say n
2. Read the data points, say x[i] & y[i]
3. Calculate needed summation as below
   1. For i=0 to n-1
      1. Sx=sx+x[i]
      2. Slogy=slogy+logy[i]
      3. Sxy=sxy+s[i]\*logy[i]
      4. Sx2=sx2+x[i]\*x[i]
   2. End for
4. Calculate a & b by using formula as
   1. B=((n\*sxy)-(sx\*slogy))/((n\*sx2)-(sx\*sx))
   2. R=(slogy/n)-b\*(sx/n)
   3. A=e^r
5. Display the equation ae^bx
6. Stop

Fitting the polynomial

1. Read the number of points, say n
2. Read the order of polynomial say m
3. Read the data points, say x[i] fx[i]
4. Calculate the required summations as below
   1. For i=1 to 2m
      1. For j=0 to n-1 Sx[i]=sx[i]+pow(x[j],i)
      2. End for
   2. End for
   3. For i=1 to m
      1. For j=0 to n-1
      2. Sxy[i]=sxy[i]+y[j]\*pow(x[j],i)
      3. End for
   4. End for
5. Compute the lHS coefficient matrix of order (m+1)\*(m+1)
6. Construct the RHS matrix of order (m+1)\*1
7. Solve for coefficients a0, a1, a2…..am using Gauss Elimination Method
8. Display the equation a0+a1x+a2x^2+a3x^3+……….+amk^m

Gauss Elimination Method

1. Read the dimension of the system of equations say n
2. Read the coefficient matrix row-wise
3. Read RHS vector
4. Perform forward elimination
   1. For k=1 to n-1
      1. Pivot=a[k][k]
      2. If (pivot<0.000001)
         1. Display “Method Failed”
      3. Else
         1. For i=k+1 to n
            1. Term = a[i][k]/pivot
            2. Term = a[i][k] – term\*a[k][k]
            3. Multiply row k of b matrix by term & subtract if from row i
         2. End for
      4. End for
   2. End for
5. Perform back substation as follow
   1. X[n]=b[n]/a[n][n]
   2. For i=n-1 to 1
   3. Sum=0
      1. For j=i+1 to n
         1. Sum=sum+a[i][j]\*x[j]
      2. End for
   4. X[i]=(b[i]-sum)/a[i][i]
   5. End for
6. Display solution vector
7. Stop

Gauss Elimination with partial pivoting

1. Read the dimension of the system of equations say n
2. Read the coefficient matrix row-wise
3. Read RHS vector
4. Perform forward elimination
   1. For k=1 to n-1
      1. Find the largest of a[p][k] for p=k,k+1,..n
      2. Swap row k & row p in coefficient matrix
      3. Swap row k & row p in RHS vector
      4. Pivot=a[k][k]
         1. For i=k+1 to n
            1. Term = a[i][k]/pivot
            2. Term = a[i][k] – term\*a[k][k]
            3. Multiply row k of b matrix by term & subtract if from row i
         2. End for
   2. End for
5. Perform back substation as follow
   1. X[n]=b[n]/a[n][n]
   2. For i=n-1 to 1
   3. Sum=0
      1. For j=i+1 to n
         1. Sum=sum+a[i][j]\*x[j]
      2. End for
   4. X[i]=(b[i]-sum)/a[i][i]
   5. End for
6. Display solution vector
7. Stop

Gauss Jordan Method

1. Normalize the 1st equation by dividing it by its pivot element
2. Eliminate x1 terms from all the other equations
3. Now normalize the second eqn by dividing its by pivot element
4. Eliminate x2 from all other eqns above and below the normalize pivotal equation
5. Repeat this process until xn is eliminated from all but the last equation
6. The resultant b vector is the solution vector