Algorithms for finding roots.

1) Bisection Method -

1. Read a, b such that $a < b$ & $f(a)f(b) <$
2. $m = (a+b)/2$
3. If $f(m) = 0$ then m is an exact root, else if $f(a) \cdot f(m) < 0$ then $b = m$ else if $f(m) \cdot f(b) < 0$ then $b = m$
4. Repeat steps 2 & 3 until $f(m_i) = 0$
5. End else if
6. End

2) Regula Falsi Method -

1. Read values of x_0, x_1 & $E = 0.001$
2. Function value $f(x_0)$ & $f(x_1)$
3. check whether product of $f(x_0)$ & $f(x_1)$ is negative or not.
4. Determine $m = \dfrac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$
5. Check product x_1 & x_0 is negativ or not if (-ve) $x_0 = x$, (+ve) $x_1 = x$

6. check the value of $f(m)$ is greater than 0.0001 or not
7. display the root as x
8. stop

3) Algorithm for Newton's Method
1. Read b, $E = 0.001$
2. $m = b - \frac{f(b)}{f'(b)}$
3. $i = 1$
4. while $(|f(m)| > E)$
5. print $(i, b, m, f(m))$
6. $b \leftarrow m$
7. $m = b - \frac{f(b)}{f'(b)}$
8. $i \leftarrow i + 1$
9. end of while
10. end

4) Secant Method
1. Read value x_0, x_1 & $E = 0.001$
2. compute $f(x_0)$ & $f(x_1)$

(3) $m = \dfrac{[x_0 * f(x_1) - x_1 * f(x_0)]}{f(x_1) - f(x_0)}$

(4) $f[(x_2 - x_1)/m] > E$

(5) Display the required root as m

(6) Stop

5) Fixed Point

(1) Read b, $E = 0.0001$

(2) $m = g(b)$

(3) $i = 1$

(4) while ($|b - m| >= E$) do

print i, b, m, abs (m-b)

b = m

m = ~~f~~g(b)

i = i+1

end while

(5) print i, b, m, abs (m - b)

(6) end of algorithm

Algorithm for applying interpolatio

1) Lagrange's Interpolation

(1) start

(2) Read no. of data (n)
(3) Read data x_i & y_i for $i=1$ to n
(4) Read value of independent variable x_p & corresponding value y_p
(5) $i=0$
(6) for $i=1$ to n
$p=1$
for $j=1$ to n
if $i \neq j$ then
calculate $p = p * (x_p - x_j) / (x_i - x_j)$
end if
next j
calculate $y_p = y_p + p * y_i$
Next i
(7) Display value of y_p
(8) Stop

2) Hermite Interpolation formula

$$n_m(x) = \sum_{i=0}^{n} \sum_{j=0}^{a_i - 1} \sum_{k=0}^{x_i - j - 1}$$

$$f^{(j)}(x_i) \frac{1}{k!} \frac{1}{j!} \left[\frac{(x - x_i)^{x_i}}{\Omega(x)} \right] 1k$$

3) Divide difference Table

1. Start
2. Read divided difference
   $F_{0,0} \dots F_{n,n}$
3. for $i = 0, \dots n$
   set $F_{i,0} = f(x_i)$
4. for $i = 1 \dots n$
   for $j = 1 \dots i$
   set $f_{i,j} = \dfrac{f_{i,j-1} - f_{i-1,j-1}}{x_i - x_{i-j}}$
   end
   end
   Output $(F_{0,0} \dots F_{i,i} \dots F_{n,n})$

4) Newton Interpolation (Forward)

$$f(a+hu) = f(a) + \frac{u \Delta f(a)}{\lfloor 1} + \frac{u(u-1)}{\lfloor 2} \Delta^2 f(a) + \dots$$

1. Start
2. Read n
3. for i=0 to n-1
   Read x_i, $y_i[0]$
   end i
4. for j=1 to n-1

for i = 0 to n - 1 - j
yi[j] = y[i+1][j-1] - y[i][j-1]
end i
end j

(5) for i = 0 to n - 1
for j = 0 to n - 1 - i
print yi[j]
end j
end i

(6) Read a
(7) Assign h = x[1] - x[0]
(8) Assign u = (a - x[0])/h
(9) Assign sum = y0[0] & p = 1.0
(10) for j = 1 to n - 1
p = p * (u - j + 1)/j
sum = sum + p * y0[j]
end j
(11) Display a & sum
(12) Stop

Algorithm for direct method

1) Gauss Elimination method -
(i) Red n

(2) for i = 1 to n steps of 1 do
(3) for i = 1 to (n+1) steps of 1 do
(4) Read a_{ij}
end for
end for
(5) for k = 1 to (n-1) in steps of 1 do
(6) for i = (k+1) to n in steps of 1 do
(7) $u \leftarrow a_{ik} / a_{kk}$
(8) for j = k to (n+1) in steps of 1 do
$a_{ij} \leftarrow a_{ij} - a_{kj} * u$
end for
end for
(9) $x_n \leftarrow \dfrac{a_{n(n+1)}}{a_{nn}}$
(10) for i = (n-1) to 1 in step of 1
(11) sum $\leftarrow$ 0.1
(12) for j = i+1 to n in steps of 1 do
(13) sum $\leftarrow$ sum + $a_{ij} * x_j$
end for
(14) $x_i \leftarrow 1/a_{ii} [a_{i(n+1)} - \text{sum}]$

2) Gauss Jacobi Method

(1) Read n

(2) for i=1 to n steps of 1 do
(3) for j=1 to (n+1) steps of 1 do
Read a_{ij}
end for
end for
(4) for k=1 to ~~(n-1)~~ matrix by 1 do
big - e ← 0
for i=1 to n by 1 do
(5) sum ← 0.0 → (Intially zero)
(6) for j=1 to n steps of 1
if $i \neq j$
end for
(7) new - x_j = [$a_j(n+1)$ - sum] / a_{jj}
$E = \left| \frac{new - x_j - old\ x_j}{new . x_j} \right|$
if (error > big.e)
big - e = e
end if
end for
(8) for i=1 to n by 1 do
set old - x_i = new - x_i
end for
printf " sol^n does not converges in matrix "
end for algorithm

Gauss Seidel Method

1. Read n
2. for i = 1 to n 1 to do
   for j = 1 to n 1 to do
   Read = aij
   end for
   end for
3. Read max itr, ϵ
4. for i = 1 to n by 1 do
   set $x_i \leftarrow 0.0$
   end for
5. for k = 1 to max. itr by 1 do
   big - e $\leftarrow$ 0
   for i = 1 to n by 1 do
   sum $\leftarrow$ 0.0
6. for j = 1 to n by 1 do
   if ($i \neq j$)
   sum $\leftarrow$ sum + $a_{ij} * x_j$
   endif
   end for
7. temp $\leftarrow$ [$a_{i(n+1)}$ - sum] / a_{ii}
8. error = $\dfrac{temp - x_i}{tem}$

(9) xi ← temp
if (error > big e)
big _e = error
endif
end for
(10) if (big - e ≤ e) then
print " sol" converges", k," iteration";
for i = 1 to n by 1 to do
print xj
end for
exit
end if
end for
print " sol" don't converges in matrix'