

```

1:
2: *****
3: /ql_ans/roots.f90
4:
5: PROGRAM roots
6: !***** Main program for root finding
7: !*
8: !* Uses bisection and newton-raphson from the
9: !* module "roots_mod"
10:
11: USE roots_mod !** Use the required module
12:
13: REAL :: a=-20,b=0 !** Equation One
14: REAL :: a=-1,b=1 !** Equation Two
15: REAL :: a=1,b=2 !** Equation Three
16: REAL :: x=12 !** Newton Raphson
17:
18: REAL :: root
19: REAL, PARAMETER :: tol=0.0001
20:
21: INTEGER :: i, iters
22: INTEGER, PARAMETER :: maxiters=200
23: LOGICAL, PARAMETER :: output=.TRUE.
24: LOGICAL, DIMENSION(2) :: conv
25:
26: root=newt_raph(x,tol,maxiters,iters,conv,output) !** Return root in x
27:
28:
29:
30: IF (conv(1)) THEN
31: PRINT/("The root is estimated as : ",F10.6), root !** Bisection
32: PRINT/("The number of iterations taken was",i5),iters
33: ELSE
34: PRINT*, "Unsuccessful Convergence"
35: ENDF
36:
37: IF (conv(2)) PRINT*, "Max iters exceeded : Unsuccessful Convergence"
38:
39: END PROGRAM roots
40:
41: *****
42:
43:
44: *****
45: /ql_ans/res.f90
46:
47: ===== RESULTS =====
48:
49: Equation One
50: =====
51:

```

Iters	a	b	root	f(root)
1	-20.000000	0.000000	-10.000000	0.767387
2	-20.000000	-10.000000	-15.000000	0.414978
3	-20.000000	-15.000000	-17.500000	0.188819
4	-20.000000	-17.500000	-18.750000	0.068040
5	-20.000000	-18.750000	-19.375000	0.006490
6	-20.000000	-19.375000	-19.687500	-0.024472
7	-19.687500	-19.375000	-19.531250	-0.006979
8	-19.531250	-19.375000	-19.453125	-0.001241
9	-19.453125	-19.375000	-19.414062	0.002626
10	-19.453125	-19.414062	-19.433594	0.000693
11	-19.453125	-19.433594	-19.443359	-0.000274
12	-19.443359	-19.433594	-19.438477	-0.000209
13	-19.443359	-19.438477	-19.440918	-0.000032
14	-19.440918	-19.438477	-19.439697	0.000089
15	-19.440918	-19.439697	-19.440308	0.000028

```

68:
69:
70:
71:
72: The root is estimated as : -19.440536
73: The number of iterations taken was 18
74:
75: Equation Two
76: =====
77:
78: Iters a b root f(root)
79: 1 -1.000000 1.000000 0.000000 -1.000000
80: 2 0.000000 1.000000 0.500000 -0.500000
81: 3 0.500000 1.000000 0.750000 0.312500
82: 4 0.500000 0.750000 0.625000 -0.277344
83: 5 0.625000 0.750000 0.687500 -0.135254
84: 6 0.687500 0.750000 0.718750 -0.055237
85: 7 0.718750 0.750000 0.734375 -0.012825
86: 8 0.734375 0.750000 0.742188 0.009062
87: 9 0.734375 0.742188 0.738281 -0.001964
88: 10 0.738281 0.742188 0.740234 0.003506
89: 11 0.738281 0.740234 0.739258 0.000768
90: 12 0.738281 0.739258 0.738770 -0.000599
91: 13 0.738770 0.739258 0.739014 0.000084
92: 14 0.738770 0.739014 0.738892 -0.000258
93: 15 0.738892 0.739014 0.738953 -0.000087
94:
95: The root is estimated as : 0.738953
96: The number of iterations taken was 15
97:
98:
99: Equation Three
100: =====
101:
102: Iters a b root f(root)
103: 1 1.000000 2.000000 1.500000 2.375000
104: 2 1.000000 1.500000 1.250000 -1.796875
105: 3 1.250000 1.500000 1.375000 0.162109
106: 4 1.250000 1.375000 1.312500 -0.848389
107: 5 1.312500 1.375000 1.343750 0.350983
108: 6 1.343750 1.375000 1.359375 -0.096409
109: 7 1.359375 1.375000 1.367188 0.032356
110: 8 1.359375 1.367188 1.363281 -0.032150
111: 9 1.363281 1.367188 1.365234 0.000072
112: 10 1.363281 1.365234 1.364258 -0.016047
113: 11 1.364258 1.365234 1.364746 0.007989
114: 12 1.364746 1.365234 1.364990 -0.003960
115: 13 1.364990 1.365234 1.365112 -0.001944
116: 14 1.365112 1.365234 1.365173 -0.000936
117:
118: The root is estimated as : 1.365173
119: The number of iterations taken was 14
120:
121: Newton-Raphson
122: =====
123:
124: 1 6.0833335 35.0069466
125: 2 3.2060504 8.2787590
126: 3 1.9149355 1.6669779
127: 4 1.4796785 0.1894486
128: 5 1.4156617 0.0040979
129: 6 1.4142144 0.000024
130: 7 1.4142135 -0.0000001
131:
132: The root is estimated as : 1.414214
133: The number of iterations taken was 7
134:

```

```

135: *****
136:
137:
138: *****
139: ./q1_ans/roots_mod.f90
140:
141: MODULE roots_mod
142:
143: IMPLICIT NONE
144:
145: CONTAINS
146:
147: FUNCTION newt_raph(x,tol,maxiters,itors,conv,output)
148: !** Returns the root estimate using bisection method
149:
150: !** Dummy declaration
151: REAL, INTENT(INOUT) :: x
152: REAL, INTENT(IN) :: tol
153: INTEGER, INTENT(IN) :: maxiters
154: INTEGER, INTENT(OUT) :: iters
155: LOGICAL, DIMENSION(2), INTENT(OUT) :: conv
156: LOGICAL, INTENT(IN) :: output
157:
158: REAL :: old_root !** Local variable declarations
159: REAL :: newt_raph !** Function return declaration
160:
161: iters=0
162: DO !** Start looping.
163:   old_root=x
164:   x=x-f(x)/df(x)
165:   conv=end_loop(x,old_root,tol/2,maxiters,itors)
166:   iters=itors+1
167: IF (output) PRINT '(I4,2F12.7)',iters,x,f(x) !** Output results
168: IF (conv(1) .OR. conv(2)) THEN
169:   newt_raph=x
170: EXIT !** EXIT loop if either flag is true
171: ENDF
172: END DO
173:
174: END FUNCTION newt_raph
175:
176: ! *****
177:
178: FUNCTION bisection(a,b,tol,maxiters,itors,conv,output)
179: !** Returns the root estimate using bisection method
180:
181: !** Dummy declarations
182: REAL, INTENT(INOUT) :: a,b
183: REAL, INTENT(IN) :: tol
184: INTEGER, INTENT(IN) :: maxiters
185: INTEGER, INTENT(OUT) :: iters
186: LOGICAL, DIMENSION(2), INTENT(OUT) :: conv
187: LOGICAL, INTENT(IN) :: output
188:
189: !** Function return declaration
190: REAL :: bisection
191: !** Local variable declarations
192: LOGICAL :: outputlocal
193:
194: iters=0
195: IF (output) PRINT*, " Iters a b root f(root)"
196: DO
197:   bisection=(a+b)/2
198: conv=end_loop(a,b,tol,maxiters,itors)
199: iters=itors+1
200: IF (f(bisection) .EQ. 0) conv(1)=.TRUE.
201: !** Output results

```

```

202: IF (output) PRINT '(I4,4F12.6)',iters,a,b,bisection,f(bisection)
203: IF (conv(1) .OR. conv(2)) EXIT !** EXIT loop if either flag is true
204: IF (f(a)*f(bisection) > 0) THEN !** IF f(a) & f(p) same sign
205:   a=bisection
206: ELSE
207:   b=bisection
208: ENDF
209: END DO
210: END FUNCTION bisection
211:
212: ! *****
213:
214: FUNCTION f(x)
215: !** Returns the value of the function "f" at "x"
216:
217: REAL, INTENT(IN) :: x !** Dummy declaration
218: REAL :: f !** Function return declaration
219:
220: f=COS(x/20)+SIN(x/20)**3 !** Equation One
221: f=2*x**3-x**2+x-1 !** Equation Two
222: f=x**3+4*x**2-10 !** Equation Three
223: f=x**2-2 !** Newton-Raph
224:
225: END FUNCTION f
226:
227: ! *****
228:
229: FUNCTION df(x)
230: !** Returns the value of the derivative of the function "f" at "x"
231:
232: REAL, INTENT(IN) :: x !** Dummy declaration
233: REAL :: df !** Function return declaration
234:
235: df=2*x
236:
237: END FUNCTION df
238:
239: ! *****
240:
241: FUNCTION end_loop(num1,num2,tol,maxiters,itors)
242: !** Returns a logical type. If .TRUE. then the tolerance has been met or
243: !** the maximum number of iterations has been exceeded.
244:
245: !** Dummy variables
246: REAL, INTENT(IN) :: num1,num2
247: REAL, INTENT(IN) :: tol
248: INTEGER, INTENT(IN) :: maxiters
249: INTEGER, INTENT(INOUT) :: iters
250:
251: !** Function return declaration
252: LOGICAL, DIMENSION(2) :: end_loop
253:
254: end_loop(1)=(ABS(num1-num2)/2<tol)
255: end_loop(2)=(iters>maxiters)
256:
257: END FUNCTION end_loop
258:
259: END MODULE roots_mod
260: *****
261:

```