

```
FindRoots mySinc, SincCoefs
```

Igor finds the first of the roots we found previously:

```
Possible root found at 2.0708
Y value there is 2.748e-13
```

You may have noticed by now that FindRoots is reporting Y values that are merely small instead of zero. It isn't usually possible to find exact roots with a computer. And asking for very high accuracy requires more iterations of the search algorithm. If function evaluation is time-consuming and you don't need much accuracy, you may not want to find the root with high accuracy. Consequently, you can use the /T flag to alter the acceptable accuracy:

```
FindRoots/T=1e-3 mySinc, SincCoefs      // low accuracy
```

```
Possible root found at 2.07088
Y value there is -5.5659e-05
```

```
Print/D V_root, V_YatRoot
2.07088376061435 -5.56589998578327e-05
```

```
FindRoots/T=1e-15 mySinc, SincCoefs      // high accuracy
```

```
Possible root found at 2.0708
Y value there is 0
```

```
Print/D V_root, V_YatRoot
2.0707963267949 0
```

This also illustrates another point: the results of FindRoots are stored in variables. We used these variables in this case to print the results to higher accuracy than the six digits used by the report printed by FindRoots.

Roots of a System of Multidimensional Nonlinear Functions

Finding roots of a system of multidimensional nonlinear functions works very similarly to finding roots of a 1D nonlinear function. You provide user-defined functions that define your functions. These functions have nearly the same form as a 1D function, but they have a parameter for each independent variable. For instance, if you are going to find roots of a pair of 2D functions, the functions will look like this:

```
Function myFunc1(w, x1, x2)
  Wave w
  Variable x1, x2

  return <an arithmetic expression>
End

Function myFunc2(w, x1, x2)
  Wave w
  Variable x1, x2

  return <an arithmetic expression>
End
```

These function look just like the 1D function mySinc we wrote in the previous section, but they have two input X variables, one for each dimension. The number of functions must match the number of dimensions.

We will use the functions

```
f1= w[0]*sin((x-3)/w[1])*cos(y/w[2])
```

and

```
f2 = w[0]*cos(x/w[1])*tan((y+5)/w[2])
```

Enter this code into your procedure window:

```
Function myf1(w, xx, yy)
  Wave w
  Variable xx,yy
```

Chapter III-10 — Analysis of Functions

```
    return w[0]*sin(xx/w[1])*cos(yy/w[2])
End

Function myf2(w, xx, yy)
  Wave w
  Variable xx,yy

  return w[0]*cos(xx/w[1])*tan(yy/w[2])
End
```

Before starting, let's make a contour plot to see what we're up against. Here are some commands to make a convenient one:

```
Make/D/O params2D={1,5,4}          // nice set of parameters for both f1 and f2
Make/D/O/N=(50,50) f1Wave, f2Wave // matrix waves for contouring
SetScale/I x -20,20,f1Wave, f2Wave // nice range of X values
SetScale/I y -20,20,f1Wave, f2Wave // and Y values
f1Wave = myf1(params2D, x, y)      // fill f1Wave with values from f1(x,y)
f2Wave = myf2(params2D, x, y)      // fill f2Wave with values from f2(x,y)
Display /W=(5,42,399,396)          // graph window for contour plot
AppendMatrixContour f1Wave
AppendMatrixContour f2Wave
ModifyContour f2Wave labels=0       // suppress contour labels
ModifyContour f1Wave labels=0
ModifyContour f1Wave rgbLines=(65535,0,0) // make f1 red
ModifyContour f2Wave rgbLines=(0,0,65535) // make f2 blue
ModifyGraph lsize('f2Wave=0')=2      // make zero contours heavy
ModifyGraph lsize('f1Wave=0')=2
```

Places where the zero contours for the two functions cross are the roots we are looking for. In the contour plot you can see several, for instance the points (0,0) and (7.8, 6.4) are approximate roots.

The algorithm that searches for roots needs a starting point. You can specify this in the FindRoots command with the /X flag, or if you don't use /X, Igor starts by default at the origin, $X_n = 0$. You must also specify both functions and a coefficient wave for each function. In this case we will use the same coefficient wave for each. The functions and coefficient waves are specified in pairs. Since we are looking for roots of two 2D functions, we have two function-wave pairs:

```
FindRoots myf1,params2D, myf2,params2D
```

Igor finds a root at the origin, and prints the results. The X,Y coordinates of the root are stored in the wave W_Root:

```
  Root found after 4 function evaluations.
W_Root={0,0}
  Function values at root:
W_YatRoot={0,0}
```

The wave W_YatRoot holds the values of each of the functions evaluated at the root.

If that's not the root you want to find, use /X to specify a different starting point:

```
FindRoots/X={7.7,6.3} myf1,params2D, myf2,params2D
  Root found after 47 function evaluations.
W_Root={-1.10261e-14,12.5664}
  Function values at root:
W_YatRoot={2.20522e-15,-1.89882e-15}
```