The algorithm that searches for extreme points needs a starting guess which you provide using the /X flag. Here is a command to find a minimum point:

```
Optimize/X={1,5} Sinc2D,params2D
```

This command results in the following report in the history:

```
Optimize probably found a minimum. Optimize stopped because
the gradient is nearly zero.
Current best solution:
   W_Extremum={-0.000147116,8.98677}
Function gradient:
   W_OptGradient={-4.65661e-08,1.11923e-08}
Function value at solution: -0.217234
11 iterations, 36 function calls
V_min = -0.217234, V_OptTermCode = 1, V_OptNumIters = 11, V_OptNumFunctionCalls = 36
```

Note that the minimum is reported via a wave called W_extremum. Had you specified a wave using the /X flag, that wave would have been used instead. Another wave, W_OptGradient, receives the function gradient at the solution point.

There are quite a few options available to modify the workings of the Optimize operation. See **Optimize** operation on page V-725 for details, including references to reading material.

Stopping Tolerances

The Optimize operation stops refining the solution when certain criteria are met. The most desirable result is that it stops because the function gradient is very close to zero, since that is (almost) diagnostic of an extreme point. The algorithm also will take very small steps near an extreme point, so this is also a stopping criterion. You can set the values for the stopping criteria using the /T={*gradTol*, *stepTol*} flag.

Optimize stops when these conditions are met:

$$\max_{1 \le i \le n} \left\{ |g_i| \cdot \frac{max(|x_i|, typX_i)}{max(|f|, funcSize)} \right\} \le gradTol$$

or

$$\max_{1 \le i \le n} \left\{ \frac{|\Delta x_i|}{max(x_i, typX_i)} \right\} \le stepTol$$

Note that these conditions use values of the gradient (g_i) and step size (Δx_i) that are scaled by a measure of the magnitude of values encountered in the problem. In these equations, x_i is the value of a component of the solution and f is the value of the function at the solution; $typX_i$ is a "typical" value of the X component that is set by you using the /R flag and f is a typical function value magnitude which you set using the /Y flag. The values of $typX_i$ and f are one if the /R and /F flags are not present.

The default values for *gradTol* and *stepTol* are $\{8.53618 \times 10^{-6}, 7.28664 \times 10^{-11}\}$. These are the values recommended by Dennis and Schnabel (see the references in **Optimize** operation on page V-725) for well-behaved functions when the function values have full double precision resolution. These values are $(6.022 \times 10^{-16})^{1/3}$ and $(6.022 \times 10^{-16})^{2/3}$ as suggested by Dennis and Schnabel (see the references in **Optimize** operation on page V-725), where 6.022×10^{-16} is the smallest double precision floating point number that, when added to 1, is different from 1. Usually the default is pretty good.

Due to floating point truncation errors, it is possible to set *gradTol* and *stepTol* to values that can never be achieved. In that case you may get a message about "no solution was found that is better than the last iteration".

Problems with Multidimensional Optimization

Finding minima of multidimensional functions is by no means foolproof. The methods used by the Optimize operation are "globally convergent" which means that under suitable circumstances Optimize will be able to find some extreme point from just about any starting guess.

If the gradient of your function is zero, or very nearly so at the starting guess, Optimize has no information on which way to go to find an extreme point. Note that the Sinc2D function has a maximum exactly at (0,0). Here is what happens if you try to find a minimum starting at the origin:

```
Optimize/X={0,0} Sinc2D,params2D

==== The Optimize operation failed to find a minimum. ====
Optimize stopped because
The function gradient at your starting guess is too near zero, suggesting that
it is a critical point.
A different starting guess usually solves this problem.
Current best solution:
   W_Extremum={0,0}
Function gradient:
   W_OptGradient={0,0}
Function value at solution: 1
0 iterations, 3 function calls
V_min = 1, V_OptTermCode = 6, V_OptNumIters = 0, V_OptNumFunctionCalls = 3
```

In this example the function gradient is zero at the origin because there is a function maximum there. A gradient of zero could also be a minimum or a saddle point.

The algorithms used by the Optimize operation assume that your function is smooth, that is, that the first and second derivatives are continuous. Optimize may work with functions that violate this assumption, but it is not guaranteed.

Although Optimize tends to look downhill to the nearest minimum (or uphill to the nearest maximum), it is not guaranteed to find any particular minimum, especially if your starting guess is near a point where the gradient is small. Sometimes using a different method (/M=(*stepMethod*, *hessianMethod*) will result in a different answer. You can limit the maximum step size to keep progress more or less local (/S=*maxStep*). If you set the maximum step size too small, however, Optimize may stop early because the maximum step size is exceeded too many times. Here is an example using the Sinc2D function. If the starting guess is near the origin, the gradient is small and the solution shoots off into the hinterlands (only a portion of the history report is shown):

```
Optimize/X={1,1} Sinc2D,params2D
   W_Extremum={-61.1192,298.438}
Function gradient:
   W_OptGradient={-1.04095e-08,-1.19703e-08}
```

Use /S to limit the step size, and find a minimum nearer to the starting guess:

```
Optimize/X={1,1}/S=10 Sinc2D,params2D
   W_Extremum={-0.00014911,8.9868}
Function gradient:
   W_OptGradient={9.31323e-09,5.92775e-08}
```

But if the maximum step is too small, it doesn't work:

```
Optimize/X={1,1}/S=1 Sinc2D,params2D

==== The Optimize operation failed to find a minimum. ====
Optimize stopped because
the maximum step size was exceeded in five consecutive iterations.
This can happen if the function is unbounded (there is no minimum),
or the function approaches the minimum asymptotically. It may also be that the
maximum step size (1) is too small.
```

Another way to get a solution near by is to set the initial trust region to a small value. This works if you select double dogleg or More Hebdon as the step selection method. It does not apply to the default line search method. Here is an example (note that the double dogleg method is selected using /M={1,0}):

```
Optimize/X={1,1}/M={1,0}/F=1 Sinc2D,params2D
   W_Extremum={-0.000619834,8.9872}
Function gradient:
   W_OptGradient={1.09896e-07,2.9017e-08}
```