[11.12] *nSolve()* may ignore solution constraints

This tip demonstrates two cases for which *nSolve()* ignores solution constraints and returns a solution outside of the constraints. The first case involves the ordering of the conditional constraint if it uses local variables, and the second case relates to whether or not the independent variable is declared local or not. AMS 2.05 is used for the examples, but earlier AMS versions may also exhibit the problems.

Conditional constraint ordering

When used in a program or function, *nSolve()* does not always honor solution constraints if the constraints are local variables. If the constraint is expressed like this:

```
...
local low,high
value_1→low
value_2 →high
nsolve(f(x)=0,x0|x>low and x<high →result
...
```

then *nSolve()* seems to return the correct root. However, if the constraint conditional expression is reversed with respect to *x*, that is, *low* < *x* instead of *x* > *low*:

```
nsolve(f(x)=0,x0|low<x and x<high →result
```

then *nSolve()* may return a result that is a root, but is not bound by the constraints. As an example consider the function

$$f(x) = x^3 - 6 \cdot x^2 + 11 \cdot x - 6 + \sin(x)$$

with roots

x1 = 0.7137 9958 4872 36
x2 = 1.1085 5630 2169 9
x3 = 2.1404 2730 947

and we want to find *x3*. Various permutations of the constraints give these results:

x>2 and x<2.2	x3 returned
2<x and x<2.2	x3 returned
x>low and x<high	x3 returned
low<x and x<high	*x1 returned - incorrect!*
low<x and high>x	*x1 returned - incorrect!*
x>low and high>x	x3 returned

If the constraints are numbers, then the conditional ordering is immaterial; *nSolve()* returned the correct bounded root. However, if the constraints are local variables, then we must order the conditional expression such that the independent variable occurs *first*.

solve() gives similar interesting results:

x>2 and x<2.2	x3 returned
2<x and x<2.2	x3 returned
x>low and x<high	x3 returned

low<x and x<high	undef<0 and x = x2 or undef<0 and x = x1 or x = x3 (long solution time, also warning *More solutions may exist*)
low<x and high>x	undef<0 and x = x2 or undef<0 and x = x1 or x = x3 (even longer solution time, and warning *More solutions may exist*)
x>low and high>x	x3 returned

Indpendent variable local declaration

In what may be a related situation, *nSolve()* results seem to depend on whether or not the independent variable is declared local. I will use this program as an example:

```
solvebug(vf)
Func
©Demonstrate nSolve() failure with constraints
©29dec01/dburkett@infinet.com

Local vfs,tapx,t

© Find estimate tapx for solution
ln(1000*(vf-.015))→vfs
polyeval({3.618919682914,-31.003830334444,76.737472978603,-68.237201523917,262.4613974175
1,84.916629306139},vfs)+polyeval({-3.9287348339733E-7,5.9179552041553E-5,-0.0036896155610
467,0.123089906420 18,-2.756033233 7098,0},1/vfs)→tapx

© Find solution, with bounds from tapx
nSolve(vfts120(t)=vf,t)|t≥tapx-.4 and t≤min({tapx+.4,705.47})

EndFunc
```

and *vfts120()* looks like this:

```
vfts120(ts)
func

© This expression is all one program line.
polyeval({6.177102005557E-14,-3.724921779392E-11,9.3224938547231E-9,-1.239459227407E-6,
9.2348545475962E-5,-.003654240052 0554,.075999519225692},ts)

Endfunc
```

While it may appear complicated, the logic is quite simple. We want to solve the function *vfts120()* for the input argument *vf*. To speed the solution, we first find an approximate solution for *t*, called *tapx*. *tapx* is in turn the center of a desired solution range *tapx* - 0.4 to *tapx* + 0.4. Finally, we don't want to search for solutions beyond t = 705.47, so that constraint is included, too. As a test case, we try vfts120(100.17) = 0.01613053407. In this case the contraints are about 99.789 and 100.589, so the desired root of 100.17 is included in the bounds. As shown above, *solvebug()* returns about 63.673, which is a root, but outside of the constraints. Note that the constraint conditional expressions are ordered correctly: the independent variable appears first.

If we remove *t* from the *Local* list, then the correct root of 100.17 is returned, but a *Questionable accuracy* warning is shown. Regardless of whether or not *t* is declared local, the correct root is quickly returned without the warning message, if we use *tapx* as an initial guess for *t*, like this:

```
nSolve(vfts120(t)=vf,t=tapx)|t≥tapx-.4 and t≤min({tapx+.4,705.47})
```

So, it appears that the optimum fix for this problem is to declare the independent variable as *Local*, and also use an initial guess for the independent variable.