[11.9] *nSolve()* may return "Questionable Accuracy" warning even with good solutions

nSolve() is a reasonably robust numeric solver, but some equations give it difficulty, even when there are no obvious problems such as singularities. Futher, *nSolve()* may give the "Questionable Accuracy" warning in the status line, for solutions that seem valid. In these cases, *nSolve()* will take longer to return the answer, even when it returns nearby solutions quickly.

These two programs illustrate the problem:

```
solvebug(vf)
func
local vfs,tapx

©Find estimate tapx for solution
ln(1000*(vf-.015))→vfs
polyeval({3.618919682914,-31.003830334444,76.737472978603,-68.237201523917,262.4
6139741751,84.916629306139},vfs)+polyeval({-3.9287348339733E-7,5.9179552041553E-
5,-0.0036896155610467,0.12308990642018,-2.7560332337098,0},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


vfts120(ts)
func

polyeval({6.177102005557E-14,-3.724921779392E-11,9.3224938547231E-9,-1.239459227
407E-6,9.2348545475962E-5,-.00365424005205 54,.075999519225692},ts)

Endfunc
```

solvebug() is a program that tries to find a solution *t* to the function vfts120(t) = vf, where *vf* is supplied as an argument. *vfts()* calculates a 6th-order polynomial polynomial function of its argument *ts*. Some typical values of vfts120(ts) near ts = 100 are

ts	vfts120(ts)
100	0.01612996896145
100.5	0.01613163512705
101	0.01613331366868

One way to test *solvebug()* solutions is to use a call like this:

```
solvebug(vfts120(t))
```

In general this expression should return a value near *t.* For example, if use set t = 100, this expression quickly returns 100. But if we set t = 100.17, it takes longer to find the solution, and the "Questionable Accuracy" warning appears in the display status line. However, the residual error is only about -6.6E-9, so the solution is as good as can be expected with *nSolve().*

vfts120() is well-behaved in this area, and very nearly linear. I emailed TI-cares about this problem, and here is their response:

> *"The evaluation of solvebug(vfts120(100.17)) results in the computation of nSolve(vfts120(t) = vfts120(100.17),t) with computed bounds that keep the search in the neighborhood of t = 100.17. On the Y= screen define y1(x) = vfts120(x) - vfts120(100.17). On the Window screen set the following window dimensions*

xmin = 100.17 - 1.2E-7
xmax = 100.17 + 1.2E-7
xscl = 1.E-8
ymin = -4.E-13
ymax = 4.E-13
yscl = 1.E-14
xres = 1.

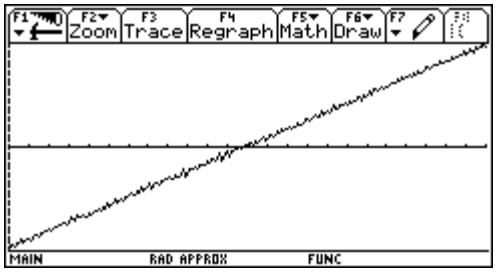
Graph y1(x). The graph shows a greal deal of roundoff noise due to catastrophic cancelation. This roundoff noise can also be seen by generating the Table corresponding to the Graph.

The roundoff error causes several apparent local maximums in the neighborhood of the reported solution. These local maximums get very close to the x-axis; that is, they almost, but don't quite, produce sign changes. To the nSolve algorithm these apparent local maximums so close to zero appear to be "candidate" even-order solutions.

Floating point roundoff error can perturb a legitimate even-order solution so that the residual doesn't change sign and doesn't quite reach zero. When such a residual is very-nearly zero, nSolve reports the position as a "candidate" solution but also issues the "Questionable accuracy" warning. The computed bounds affect the sequence of sample values, so they can strongly affect which solution or "candidate" solution is approached and the number of steps (length of time) to settle on that solution. Since nSolve only seeks one solution, it may report a "candidate" solution even when there is a nearby solution that DOES have a sign change.

To summarize, the computed bounds cause the sequence of nSolve steps to take a somewhat longer time to settle on a "candidate" solution that is produced by roundoff noise due to catastrophic cancelation. While this "candidate" solution is not the best available, it is a good approximation with a very small relative error. Given the catastrophic cancelation and the fact that the slope of the curve is extremely small in the neighborhood of the solution, the reported "candidate" solution is a very good result despite the conservative "Questionable Accuracy" warning. Moreover, the computing time is quite modest for such an example."

I followed TI's advice and plotted the function, which looks like this:



As TI wrote and this plot shows, the difference between vfts120(x) and vfts120(100.17) is not a smooth curve. The 'catastrophic cancellation' to which TI refers is also called destructive cancellation, and refers to computation errors that result from subtracting two nearly equal numbers. This results in a loss of significant digits during the calculation.

Note also that the y-scale for this plot is very small, ±4E-13, so the plot shows effects that are not usually evident with the display resolution of 12 significant digits.

In summary, be aware that *nSolve()* may return the "Questionable Accuracy" warning even for solutions that are fairly good. And, in situations like this, *nSolve()* will take slightly longer to find the solution.