Equation Solving Case Study

A customer presented us with the following problem, along with a note suggesting that they were having difficulty solving the equations. The problem involved some geometry calculations. Specifically, the solution to the problem represented the point at which two surfaces blended together. The equations forced the slopes to be equal at the solution point. Here are the equations.

$$Zc2 = Zmax - Rrad2$$

$$Yb = \frac{1}{4} \cdot C1 \cdot C2 \cdot e^{(C1 \cdot Zb)}$$

$$Yb = Yc2 + \sqrt{Rrad2^2 - (Zc2 - Zb)^2}$$

$$Yb \cdot C1 = \frac{Zc2 - Zb}{Yb - Yc2}$$

There are two more equations which define the slopes of the two surfaces.

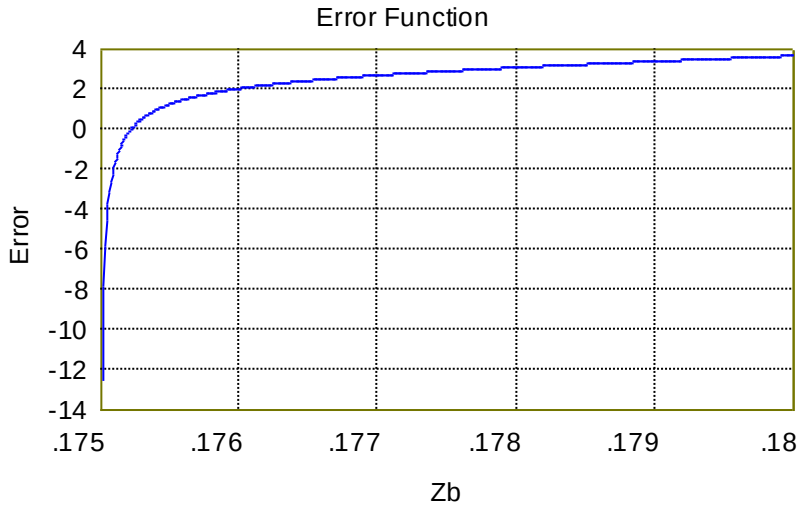
$$DY2DZb = \frac{1}{4} \cdot C1^2 \cdot C2 \cdot e^{(C1 \cdot Zb)}$$

$$DY3DZb = \frac{(Zmax - Rrad2) - Zb}{\sqrt{Rrad2^2 - ((Zmax - Rrad2) - Zb)^2}}$$

The first equation solves directly, so the iterative solver is really dealing with two equations and 2 unknowns – Zb and Yb. If you guess either Yb or Zb, the other is solved by the second equation and then Yc2 is solved by the third equation. Once Zb is known, the two slope equations are solved directly. Here are some typical inputs.

St	Input	Name	Output	Unit	Comment
	21	C1			EXP GAIN OF PORT
	.000761194	C2			CONSTANT GAIN OF 1 PORT
	.005	Rrad2			RADIUS OF EXP-CORNER INTERSECTI(
	.185	Zmax			MAX Z OF VALVE

What's a good guess value for Zb? We know from the last equation that Zb must be less than Zmax-Rrad2, otherwise we would have a square root of a negative number. We also see, from the third equation, that Zc2-Zb must be less than Rrad2. This bounds the guess between 0.175 and 0.18. We can use TK to plot the error function between those bounds. To do that, we overdefine the problem, making Zb an input and observe which equation becomes overdefined. In this case, we use .1775 as the input for Zb because it's in the center of the bounded range. TK shows that the fourth rule becomes overdefined. We edit in an error term on the right side of that rule. Then we list solve for values of Zb going from .175 to .18 in steps of 0.00001 and plot the resulting error values.



The solution is at the point where the plot crosses the 0 line – just to the right of the drop-off at 0.175.

This error plot is very revealing. If the guess is too far to the right, the slope of the error function will position the next guess at a point to the left of the bound, causing an error condition. (TK's iterative solver uses a modified Newton-Raphson method which selects the next guess based on the slope of the error function at the current guess. That is, the point at which the slope crosses the 0 line will be the value TK chooses for the next guess.) Even a guess as close as 0.176 would likely cause an error on the next iteration. TK simply "flies off the cliff."

We can avoid this problem by choosing a guess very near 0.175, in the region where the error function slope is very large. For example, 0.1751 should work nicely. This would allow TK to "climb up the cliff" slowly until reaching the solution point.

Considering all this, we might conclude that a good guess for Zb is represented by the expression

Zmax - 1.99*Rrad2

This would put the guess at a point 2% of the way across the valid domain.

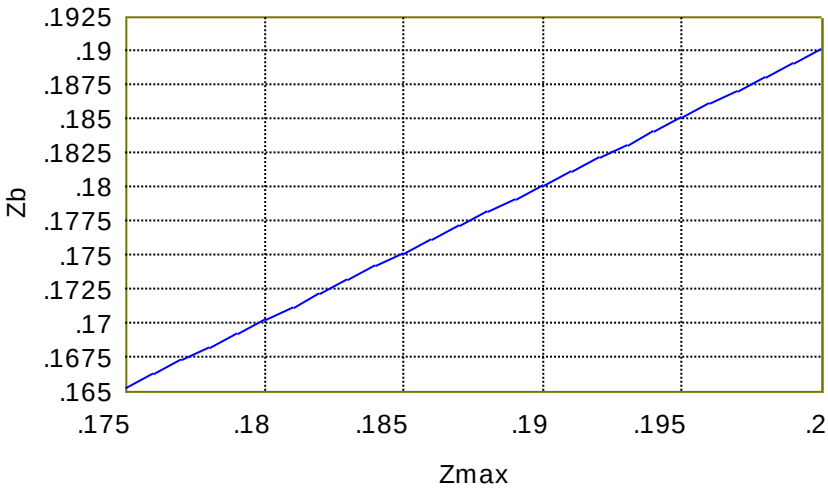
We can automate that calculation by using a guess trigger variable in a rule.

Zb = Zbg*(Zmax-1.99*Rrad2)

If Zbg is always given a first guess of 1, Zb will start where we want it. Removing the error term from the fourth equation, and giving a first guess of 1 on the variable subsheet for Zbg, TK produces the following result.

St	Input	Name	Output	Unit	Comment
		Zb	.175211818		Z POSITION OF POINT b
		Yb	.158350231		Y POSITION OF POINT b

We can now list solve to produce a plot of Zb vs. Zmax.


.1925
.19
.1875
.185
.1825
.18
.1775
.175
.1725
.17
.1675
.165
Zb
.175
.18
.185
.19
.195
.2
Zmax