Adding Constraints to Equation Solving

TK's iterative solver does not allow users to constrain the solution to be within a certain range. However the new optimizer in TK5 solves that problem. Let's use a familiar geometry problem as an example.

Here are some equations for a cone.

$$volume = \frac{1}{3} \cdot \pi \cdot radius^2 \cdot height$$

$$radius^2 + height^2 = slant^2$$

$$surface = \pi \cdot radius \cdot slant$$

$$tand\left[\frac{theta}{2}\right] = \frac{radius}{height}$$

Here is the corresponding TK Variable Sheet.

St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
		radius		in	
		height		in	
	8.5	slant		in	
		surface		in^2	
		theta		deg	

Given inputs for volume and slant, each of the four equations has at least two unknowns, so iteration is required to solve for them. Several solutions are possible, depending on the initial guess. For example, if we guess a radius value of 2, TK iterates to the solution shown below. A guess of 7 results in the second solution. Other solutions are possible for guesses such as radius = -3 and radius = -7.

St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
		radius	3.10740181	in	
		height	7.91164041	in	
	8.5	slant		in	
		surface	82.978621	in^2	
		theta	42.8861542	deg	

St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
		radius	8.43180749	in	
		height	1.07453361	in	
	8.5	slant		in	
		surface	225.159088	in^2	
		theta	165.474962	deg	

The TK optimizer provides an alternative approach to finding a solution.

The first step is to input your initial guess value on the variable sheet, as an input and not a guess. This causes the model to be overdefined. Solve, and TK displays the error message and identifies the rule in which the inconsistency is detected. In this case, the second rule is marked, along with the three variables in that rule.

St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
>	2	radius		in	
>		height	19.0985932	in	
>	8.5	slant		in	
		surface		in^2	
		theta		deg	

Status	Rule
Satisfied	volume = 1/3 * pi() * radius^2 * height
> Inconsistent	radius^2 + height^2 = slant^2
* Unsatisfied	surface = pi() * radius * slant
* Unsatisfied	tand(theta/2) = radius/height

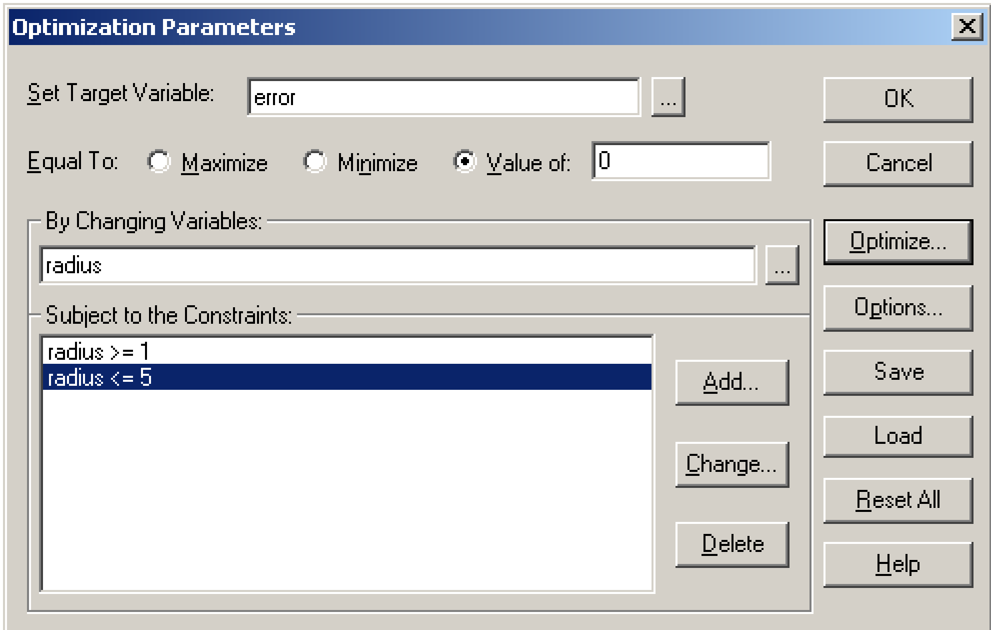
The next step is to edit the inconsistent rule, adding an error term.

Status	Rule
Satisfied	volume = 1/3 * pi() * radius^2 * height
Satisfied	radius^2 + height^2 = slant^2 + error
Satisfied	surface = pi() * radius * slant
Satisfied	tand(theta/2) = radius/height

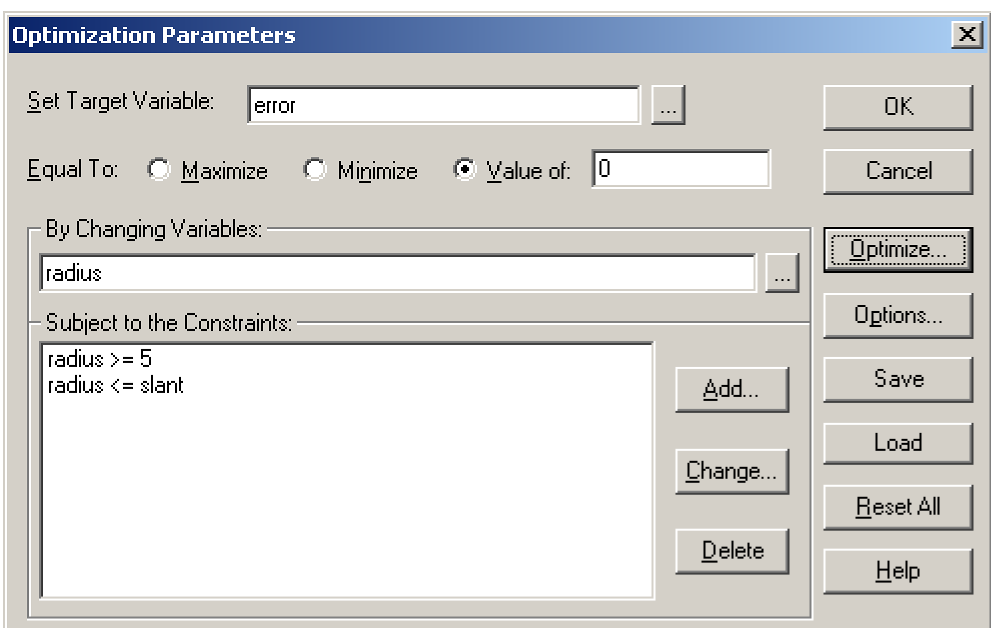
The variable sheet now includes the new variable. A solution to the equations is now found whenever the value of error is 0. The optimizer can now be used to determine the value of radius which causes this to happen.

St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
	2	radius		in	
		height	19.0985932	in	
	8.5	slant		in	
		surface	53.4070751	in^2	
		theta	11.9564215	deg	
		error	296.506261		

The optimizer is launched. The target and change variables are defined. The constraints are added. It is important to note that the initial guess for the change variables in the optimizer are the current values on the variable sheet. In this case, the change variable is radius, and it will start with the value 2. We can search for a solution between radius = 1 and radius = 5 by configuring the optimizer as shown in the first case below. The second case assumes a guess of 6 with new constraints limiting the radius to between 5 and 8.5 (the value of slant).



St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
	3.1074	radius		in	
		height	7.91164041	in	
	8.5	slant		in	
		surface	82.978621	in^2	
		theta	42.8861542	deg	
		error	-4.263E-14		



St	Input	Name	Output	Unit	Comment
	80	volume		in^3	
	8.43181	radius		in	
		height	1.07453361	in	
	8.5	slant		in	
		surface	225.159087	in^2	
		theta	165.474962	deg	
		error	-2.834E-7		

Summary:

Simple models like this one would not require the optimizer but this simple case demonstrates the technique to use for models of any level of complexity.

1. Overdefine the problem by inputting a value for the guess variable(s).
2. Locate the inconsistent equation(s) and edit in error terms until the inconsistency is gone.
3. Use the optimizer to minimize the error. If you have more than one error term, you should create a new rule which adds the squares of the each of the errors, forming a single error variable.