Case Study – LOTS of Topics All in One Example

A customer sent me a TK5 model which wasn't loading into TK6 properly. I was getting "Incomplete Load" errors in TK6. I looked at the rule sheet in TK5 and saw that an old TK Library function named Simpson was being used to do numerical integration. The topic of the model was beam deflection using the "Generalized Castigliano Theorem".

I checked the Function Sheet and the Simpson function was displayed in red, indicating that it was an "Included" function that was not actually merged into the model. I suspect that the TK6 model conversion process cannot handle TK5 models with included objects. A simple remedy is merge the included objects via the File Menu – File, Include, Merge Included Objects. Then I saved the file and TK6 had no trouble loading it.

I'm not sure why the Simpson function was being used. The built-in INTEGRAL function is faster, more accurate, and more robust. I noticed that the model was generating a plot by list solving the integrals so speed could be a factor. So here is the old rule and the new rule.

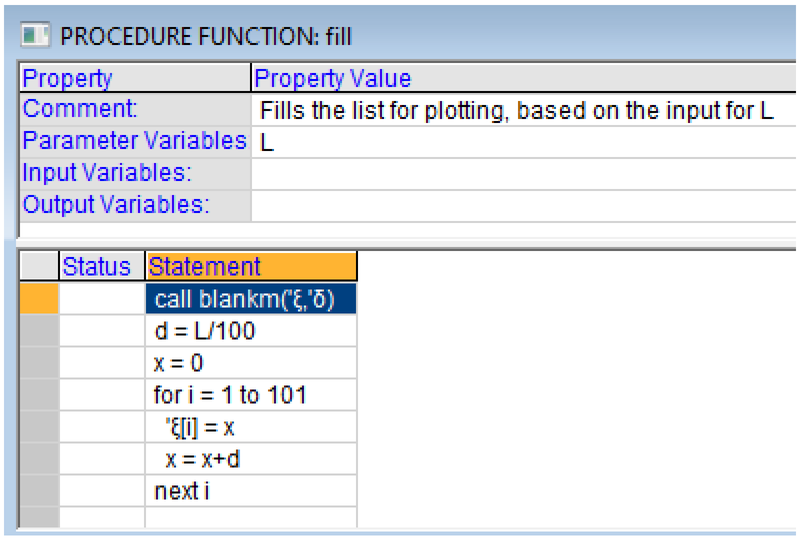
$$0 = \mathrm{Simpson}('\mathrm{INTEGRANDR}, 0, L, n) + \lambda \cdot L$$
$$0 = \mathrm{INTEGRAL}('\mathrm{INTEGRANDR}, 0, L) + \lambda \cdot L$$

The Simpson function required one extra input variable, n, representing the number of integration slices to use in processing the integrand.

The next thing I noticed was that the variable sheet had no units in the Units column. Instead, the units were listed in the Comment column as part of the description for each variable. It was simple to add the units to the Unit column and merge in a Unit Sheet from the TK Library. I also could have used the Units Import Wizard to pull in the conversion factors I wanted.

At this point, the model could be solved on the Variable Sheet to find a solution at a single point. If a plot of the deflections along the length of the beam was required, list solving could be used. That approach of F9 then F10 always seems clumsy to me so I always try to replace the list solving calculations with a procedure function if possible. For example, the model was sent to me with an input of 20 for the beam length. The model included a list of values for the list solver to work through, going from one end of the beam to the other. But what if the user changes the length of the beam? They would need to fill the input list with new values going from 0 to the new length. It's better to automate such things.

I created a procedure function called fill, with the purpose of blanking the plot lists and then filling the input list from 0 to the beam length, L.



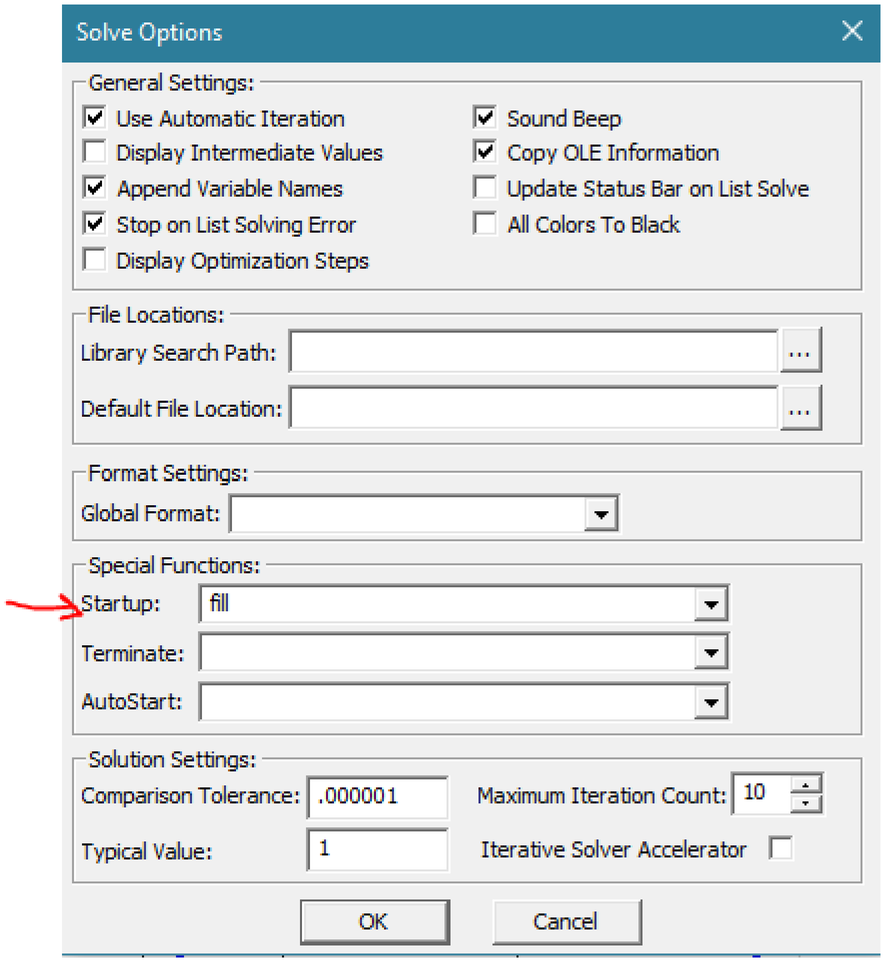
PROCEDURE FUNCTION: fill

Property	Property Value
Comment:	Fills the list for plotting, based on the input for L
Parameter Variables	L
Input Variables:	
Output Variables:	

Status	Statement
	call blankm('ξ,'δ)
	d = L/100
	x = 0
	for i = 1 to 101
	'ξ[i] = x
	x = x+d
	next i

The procedure requires one parameter variable, L, and generates 101 values in the input list locations along the beam.

Next, I declared fill as the Startup function for the model in the Solve Options form. This forces the model to run the fill procedure automatically at the start of any solution (F9 or F10).



Solve Options

General Settings:

- ☑ Use Automatic Iteration
- ☐ Display Intermediate Values
- ☑ Append Variable Names
- ☑ Stop on List Solving Error
- ☐ Display Optimization Steps
- ☑ Sound Beep
- ☑ Copy OLE Information
- ☐ Update Status Bar on List Solve
- ☐ All Colors To Black

File Locations:

Library Search Path:	
Default File Location:	

Format Settings:

Global Format:	

Special Functions:

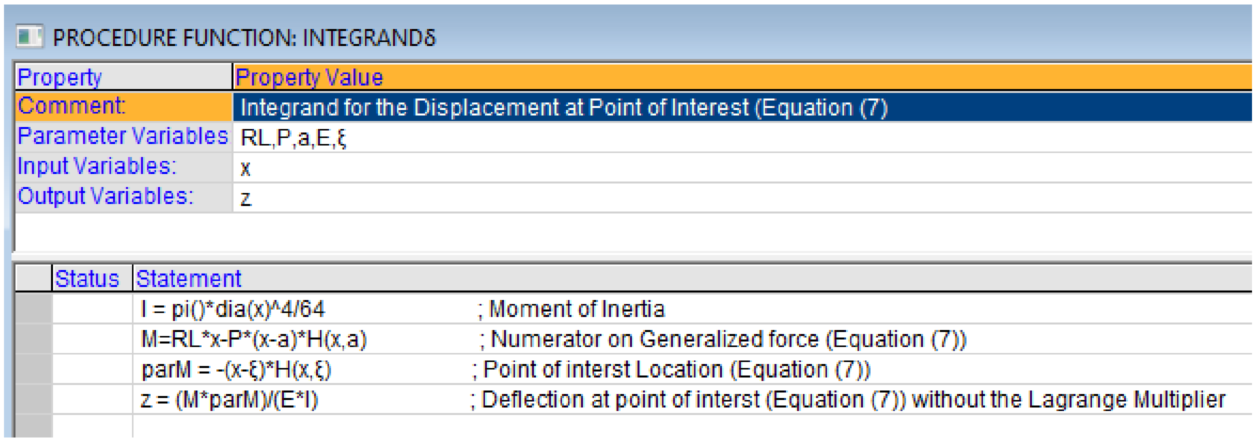
Startup:	fill
Terminate:	
AutoStart:	

Solution Settings:

Comparison Tolerance:	.000001	Maximum Iteration Count:	10
Typical Value:	1	Iterative Solver Accelerator	☐

I tested the model with a list solve and it worked nicely as I changed the beam length from 20 to 15 and watched the plot update.

Next, I looked into what was required to generate the deflection values at each location along the beam. The entire model didn't need to be solved at each location – just a single integration. Here is the integrand function in the original model sent by the customer.



PROCEDURE FUNCTION: INTEGRANDδ

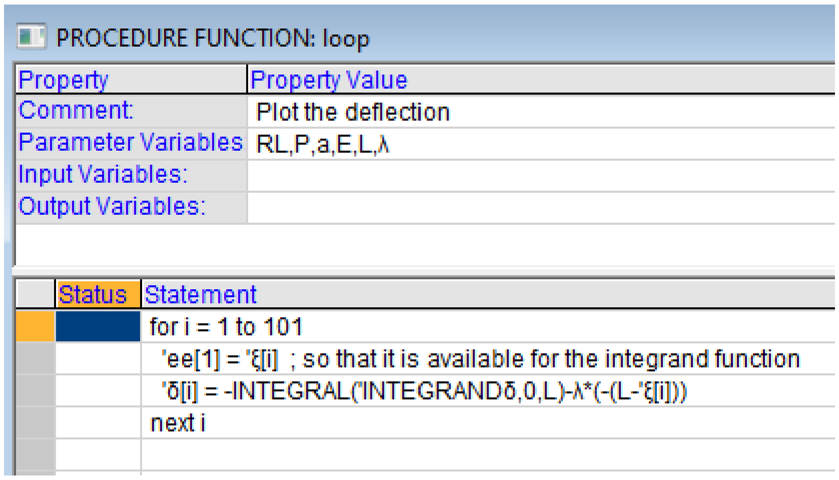
Property	Property Value
Comment:	Integrand for the Displacement at Point of Interest (Equation (7)
Parameter Variables	RL,P,a,E,ξ
Input Variables:	x
Output Variables:	z

Status	Statement
	I = pi()*dia(x)^4/64 ; Moment of Inertia
	M=RL*x-P*(x-a)*H(x,a) ; Numerator on Generalized force (Equation (7))
	parM = -(x-ξ)*H(x,ξ) ; Point of interst Location (Equation (7))
	z = (M*parM)/(E*I) ; Deflection at point of interst (Equation (7)) without the Lagrange Multiplier

And here is the equation to integrate it.

δ = -INTEGRAL('INTEGRANDδ , 0 , L) - λ· (-(L - ξ)) ;Deflection equation at point of interest (Equation (7))

I created a procedure function called "loop" to step from one end of the beam to the other, computing the deflection at each point.

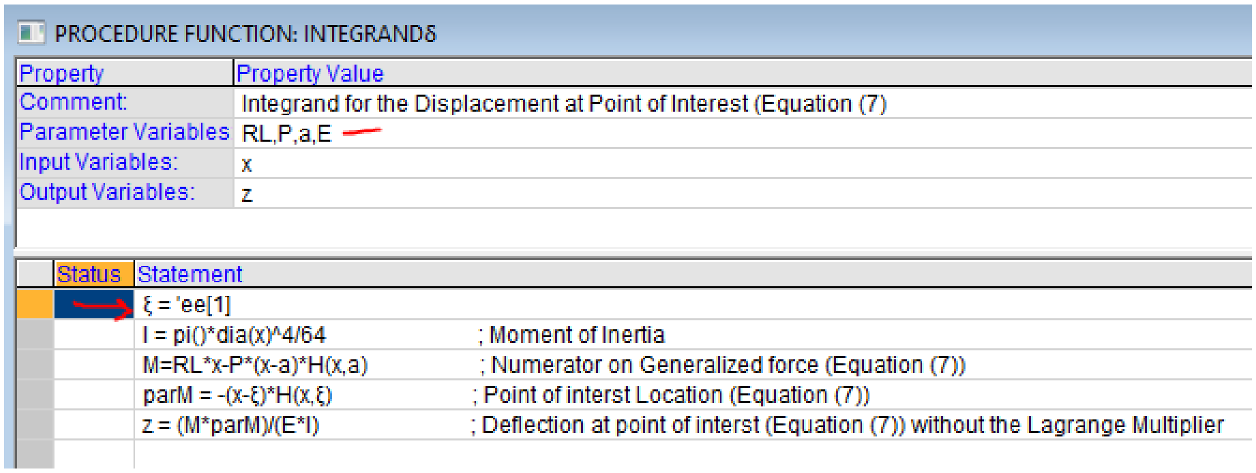


PROCEDURE FUNCTION: loop

Property	Property Value
Comment:	Plot the deflection
Parameter Variables	RL,P,a,E,L,λ
Input Variables:	
Output Variables:	

```
for i = 1 to 101
  'ee[1] = 'ξ[i]  ; so that it is available for the integrand function
  'δ[i] = -INTEGRAL('INTEGRANDδ,0,L)-λ*(-(L-'ξ[i]))
next i
```

The loop function requires several parameter variables from the Variable Sheet. Those parameters remain constant throughout the integration process. I noted that the current beam location was required by the integrand as a parameter variable but that value was changing from one step of the loop function to the next. Instead of passing the value as a variable, I passed it as a list element because list elements can be overwritten. The current beam location was placed into the first element of a new list I decided to name ee for no special reason. I modified the integrand function to get that value from the list.



PROCEDURE FUNCTION: INTEGRANDδ

Property	Property Value
Comment:	Integrand for the Displacement at Point of Interest (Equation (7)
Parameter Variables	RL,P,a,E
Input Variables:	x
Output Variables:	z

Status	Statement
	ξ = 'ee[1]
	I = pi()*dia(x)^4/64 ; Moment of Inertia
	M=RL*x-P*(x-a)*H(x,a) ; Numerator on Generalized force (Equation (7))
	parM = -(x-ξ)*H(x,ξ) ; Point of interst Location (Equation (7))
	z = (M*parM)/(E*I) ; Deflection at point of interst (Equation (7)) without the Lagrange Multiplier

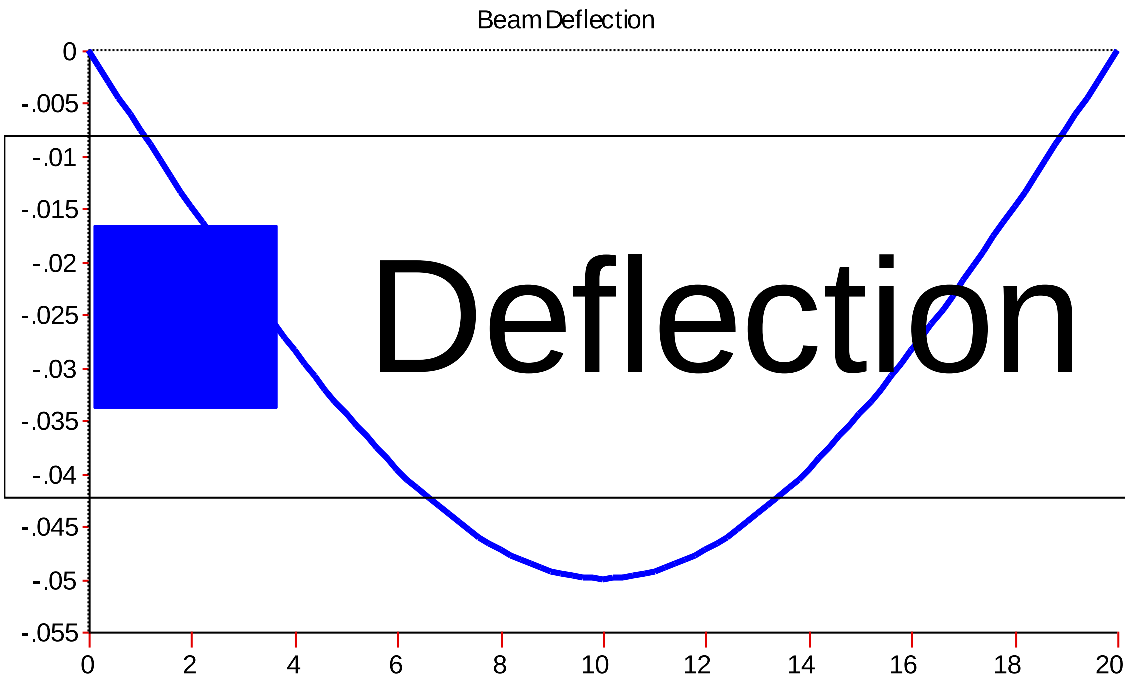
As you can see there, I removed the location variable from the list of Parameter Variables and inserted the new statement in the first line of the integrand function to get the current value from the ee list.

This integrand function also needs to get a value from the ee list when solving for the single point on the variable sheet. A rule is added to place the location value from the variable sheet into the list.

place('ee , 1) = ξ ; solving for deflection at the point of interest

Finally, a rule is added to call the loop function from the rule sheet.

call loop()



Success!!!