Case Study: Automating Linkage Plots

This paper describes the process in building a TK Solver model to solve linkage equations for both a single position and repeatedly through a range of motion. The equations are solved once with iteration on the rule sheet and repeatedly within a procedure function using a root-finding tool from the TK Library.

Step 1 – Enter equations to solve for a single position.

Rule
(x1,y1) = ptord(a,phi)
b^2 = (x2-x1)^2 + (y2-y1)^2
(x2-x3,y2-y3) = ptord(c,psi)
(x4-x1,y4-y1) = ptord(sqrt(m^2+n^2),atan2d(y2-y1,x2-x1)+atan2d(n,m))
d = sqrt((x4-x1)^2 + (y4-y1)^2)
e = sqrt((x4-x2)^2 + (y4-y2)^2)

The variables appear as shown below, with sample inputs and solution.

St	Input	Name	Output	Unit	Comment
	.5	a			Length of Input Crank a
	1	b			Length of Coupler Link b
	1.5	c			Length of Output Crank c
	1.2	m			Coordinates of \ u
	-.8	n			Coupler Point / v
	1.8	x3			Length of stationary link
	.5	y3			Elevation of stationary link
	45	phi		deg	Angular displacement of input crank
		x1	.353553391		End coordinate x of input crank
		y1	.353553391		End coordinate y of input crank
		x2	.552882839		End coordinate x of output crank
		y2	1.33348592		End coordinate y of output crank
		psi	146.244001	deg	Angular displacement of output crank
		x4	1.37669476		Coordinate x of coupler point
		y4	1.37000887		Coordinate y of coupler point
		d	1.44222051		Distance from point 1 to point 4
		e	.824621125		Distance from point 2 to point 4

The solution requires a single guess for the variable psi. In this case, I used 90 as the initial guess. Another solution is possible. If I had guessed 270, the resulting solution for psi would have been 225.318. Here are the outputs under the second guess condition.

St	Input	Name	Output	Unit	Comment
		x1	.353553391		End coordinate x of input crank
		y1	.353553391		End coordinate y of input crank
		x2	.745252438		End coordinate x of output crank
		y2	-.56654		End coordinate y of output crank
		psi	225.31851	deg	Angular displacement of output crank
		x4	.087517532		Coordinate x of coupler point
		y4	-1.0639179		Coordinate y of coupler point
		d	1.44222051		Distance from point 1 to point 4
		e	.824621125		Distance from point 2 to point 4

The user must decide which configuration serves their purposes and guess accordingly.

Step 2 – Create a procedure function to loop through the input crank angles (0 to 360 degrees in steps of 3), solving for the resulting linkage positions.

Here is the procedure function. We will describe it, line by line, below.

Statement
i = 0
'phix[1] = 0
psi = NewtonN('f,90,.001,1e-6)
for phi = 0 to 360 step 3
'phix[1] = phi
i = i + 1
psi = NewtonN('f,psi,.001,1e-6)
(x1,y1) = ptord(a,phi)
(x2,y2) = ptord(c,psi) + (x3,y3)
(x4,y4) = ptord(sqrt(m^2+n^2),atan2d(y2-y1,x2-x1)+atan2d(n,m)) + (x1,y1)
('xg[i+7],'yg4[i+7]) = (x4,y4)
next

We start by declaring an index variable “i” and set it equal to 0. Next, we place a 0 into the 1st element of the phix list.

The third statement references the NewtonN function to compute the value of psi when the input crank angle is 0. The first input to NewtonN is ‘f, indicating that the function f contains the equations to be solved. The second input is the initial guess for the unknown. The third variable is the differentiation increment (a relatively small number used in computing the slope of the error function at a point). The fourth input is the solution tolerance (usually 1E-6). Here is function f.

Rule
phi = 'phix[1]
(x1,y1) = ptord(a,phi)
b^2 = (x2-x1)^2 + (y2-y1)^2 + err
(x2-x3,y2-y3) = ptord(c,psi)
(x4-x1,y4-y1) = ptord(sqrt(m^2+n^2),atan2d(y2-y1,x2-x1)+atan2d(n,m))

Function f is a rule function which returns a single value given a single input. The input is psi and the output err. NewtonN will repeatedly pass values of psi into the function until err is within the given tolerance. Variables a,b,c,m,n,x3,y3 are declared as parameter variables so their values come directly from the variable sheet. The equations in this rule function are those required to compute the value of err. These are some of the same rules as are found on the rule sheet. To determine the rule requiring the error term, the model is overdefined on the variable sheet by making the guess variable, psi, an input. Upon solving, whichever rule is marked as “Inconsistent” is the one the Iterative Solver was working on and the one which NewtonN must now solve. An error term, err, is added to that rule and all the rules required to compute a value for err are then copied into the rule function f. In this case, four of the six equations on the rule sheet are required.

The first rule in the function gets the current value in the 1st element of the list phix and uses it as the value of phi. As the procedure function loops through the input crank angles, ‘phi[1] gets updated.

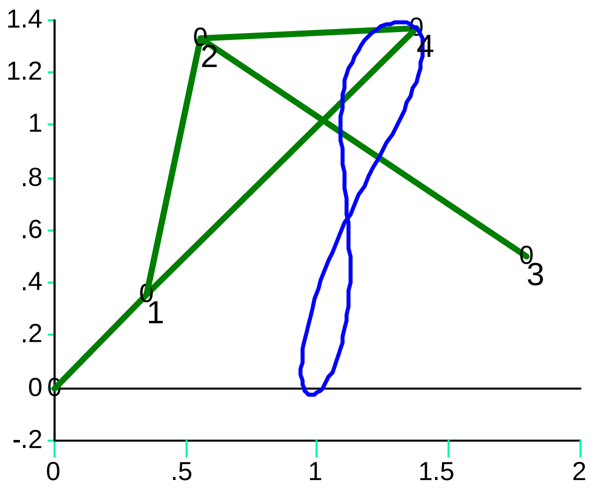
The first reference to NewtonN computes the initial linkage coordinates. These will then be used as initial conditions as the crank angle is rotated.

The loop is then defined in line 4 of the procedure, with the angles going from 0 to 360 in steps of 3. The first statement inside the loop places the crank angle into the first element of the phix list so that it can be accessed elsewhere in the model – in rule function f in particular. Next the index variable is incremented by 1.

NewtonN is used next, inside the loop, to compute the current value of psi. Then the remaining equations required to compute the plot coordinates at each input angle are added before closing the loop. The resulting lists, xg and yg4, contain the coordinates of the coupler point as it gets moved. The model also includes some procedure statements prior to the call to the looping function. Those statements produce a plot of the coordinates at the single position defined on the variable sheet. Here is the complete graphing procedure.

Statement
call blankm('xg,'yg,'yg4)
'xg[1]:= 0
'yg[1]:= 0
'xg[2]:= x1
'yg[2]:= y1
'xg[3]:= x2
'yg[3]:= y2
'xg[4]:= x4
'yg[4]:= y4
'xg[5]:= x1
'yg[5]:= y1
'xg[6]:= x2
'yg[6]:= y2
'xg[7]:= x3
'yg[7]:= y3
call looper()

All the coordinate values in this procedure come directly from the variable sheet. Here is the resulting plot.



1.4
1.2
1
.8
.6
.4
.2
0
-.2
0
.5
1
1.5
2
0
1
2
3
4