Case Study: Systems of Linear Equations, Math Languages and TK Solver

There are numerous mathematical programming languages available which include the ability to process systems of linear equations. The key to working with those languages is an understanding of matrix notation and how that notation translates into rows and columns of vectors and matrices. College courses spend weeks on this issue so it must be worthy of a case study document on how it relates to building mathematical models in TK Solver.

Here is a set of three linear equations set up as rules in TK Solver.

3*x1 + 4*x2 + 5*x3 = 4
1*x1 + 3*x2 + 1*x3 = 9
3*x1 + 5*x2 + 9*x3 = 2

Then we provide a couple of guesses to trigger the iterative solver.

Variables

Status	Input	Name	Output
Guess	1	x1	
Guess	1	x2	
		x3	

And we get the solutions.

Name	Output
x1	-1.5
x2	4
x3	-1.5

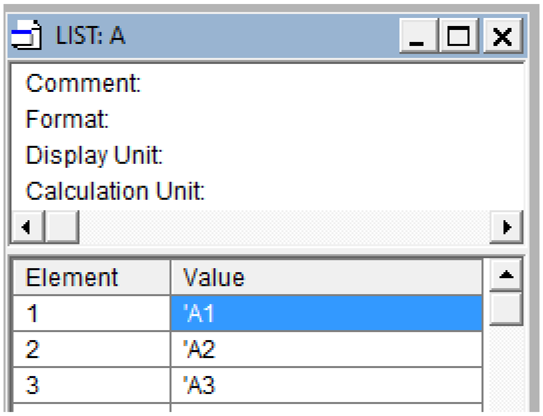
Those equations can easily be included as part of a complete mathematical model in TK Solver. The resulting model can include units and conversions, plots, and other TK features. And of course that model will be backsolvable for any variables of interest. The TK model becomes a versatile app.

Here is the matrix notation typically required to represent these linear equations in math languages. The semicolons indicate the end of a matrix row. The "\" operator is not the same as a division operator in algebraic notation. It actually can be read as "inverse of the left matrix then multiplied times the right matrix". It's efficient.

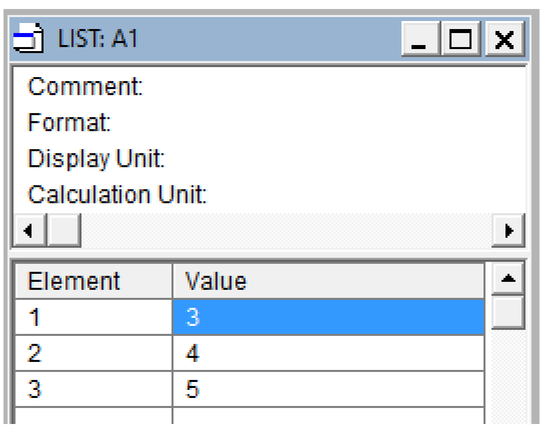
```
A = [3 4 5; 1 3 1; 3 5 9]
b = [4; 9; 2]
x = A \ b
```

TK Solver also includes matrix processing functions but does not include any special operators. For the above example, the following steps would be required, with much of it done on the List Sheet and related subsheets.

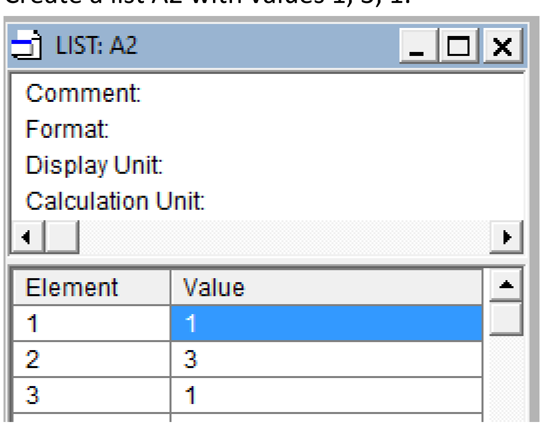
Create a list called A with elements ‘A1, ‘A2, ‘A3.



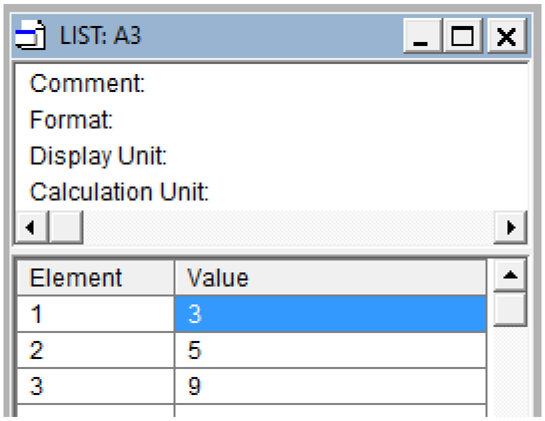
Create a list A1 with values 3, 4, 5.



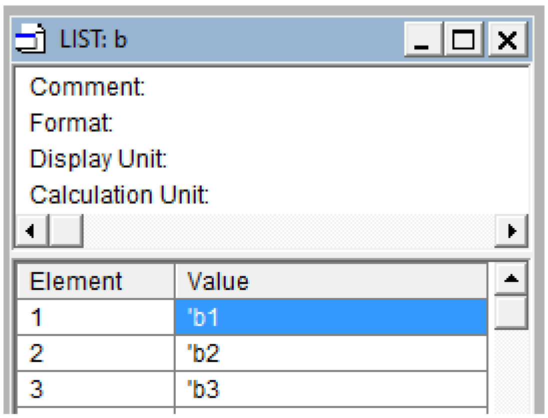
Create a list A2 with values 1, 3, 1.



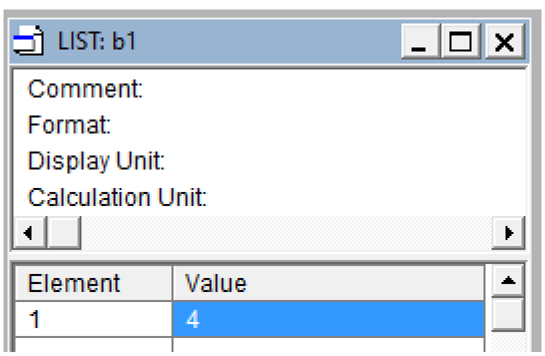
Create a list A3 with values 3, 5, 9.



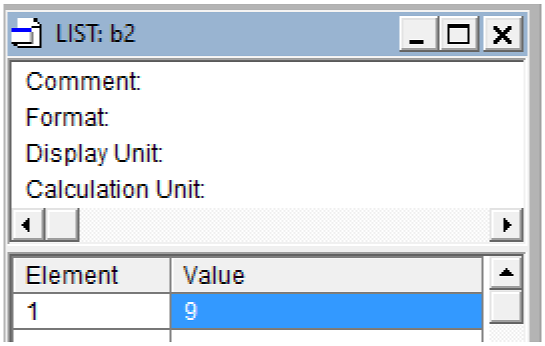
Create a list b with elements ‘b1, ‘b2, ‘b3.



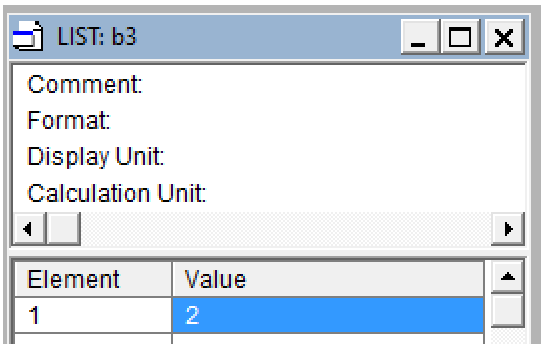
Create a list named b1 with value 4.



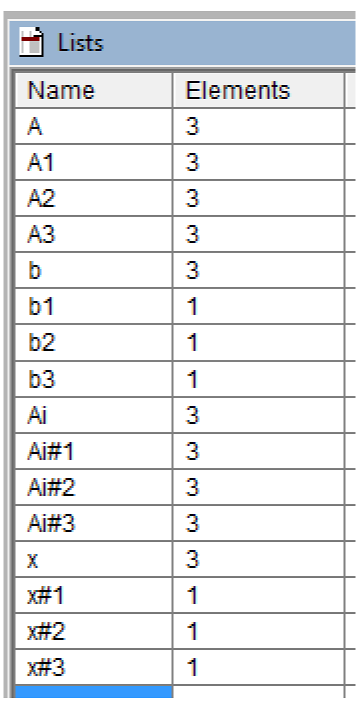
Create a list named b2 with value 9.

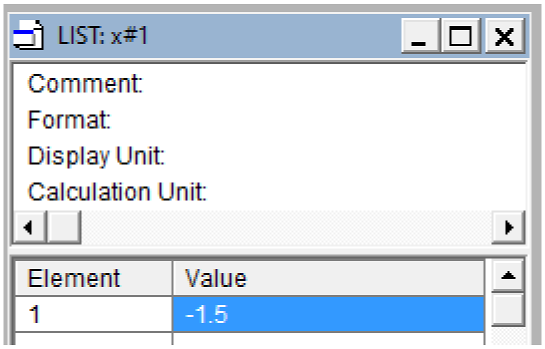


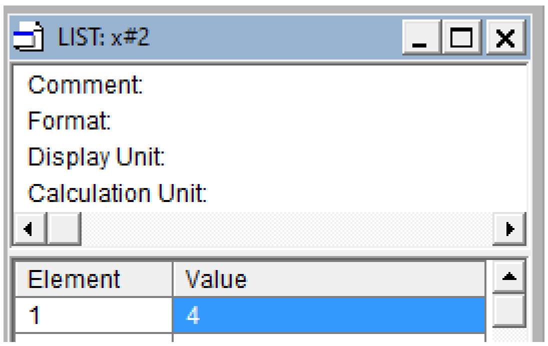
Create a list named b3 with value 2.

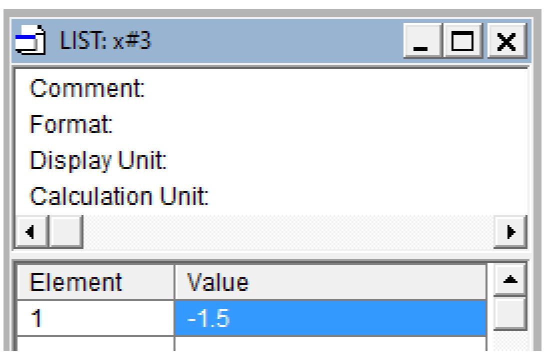


Create a rule to compute the inverse of the A matrix. Call $INV('A,'Ai)
Create a rule to multiply Ai times the b matrix to produce an x matrix. Call $MMPRODUCT('Ai,'b,'x)
Here is the List Sheet after solving. The lists Ai and below were automatically generated by the $INV and $MMPRODUCT functions. The solution values are found in lists x#1, x#2, and x#3.









Anyone who goes through those steps in TK once will likely never do it again. It's so much easier to simply type the three equations on the rule sheet and let the iterative solver find the solutions.

There are many scenarios where processing of data lists is best handled with matrix functions. The TK Solver Library includes applications that include matrix processing algorithms implemented as TK Procedure Functions. The curve-fitting applications in the Statistics section of the Library are excellent examples. Those applications only require you to supply the test data in the form of lists for the independent and dependent variables. Here is the function sheet for the Polynomial Regression app. The Comment column describes each of the procedures.

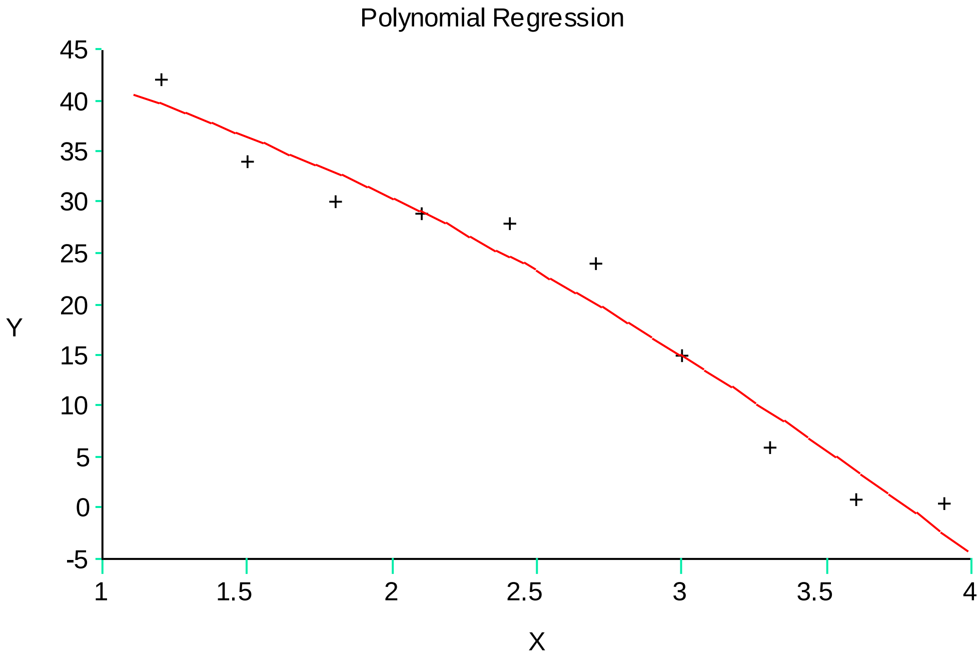
Name	Type	Arguments	Comment
matrix	Procedure	2;0	Matrix generation for polynomial regression analysis
lineq2	Procedure	3;0	No-frills linear system solver, single coefficient list
standerr	Procedure	3;3	Standard error evaluation
residual	Procedure	2;1	Generates residual values for plots and tables
statmaker	Procedure	4;0	Puts statistics into the summary table
curvemaker	Procedure	1;0	Generates the fitted polynomial
anova	Procedure	2;0	Generates Anova Table
FCDF	Rule	3;1	Snedecor's F CDF

All of that takes place behind the scenes. Here is the interface (table) for the app. It's very easy to use and you don't have to know anything about matrix operations in TK.

INT TABLE: works

Element	X	Y	residuals	SUMMARY	STATS
1	1.2	42	+2.355E0	order	2
2	1.5	34	-2.525E0	N	10
3	1.8	30	-3.015E0	Syx	3.30976647
4	2.1	29	-1.155E-1	adj R2	.948171138
5	2.4	28	+3.173E0	p	.000104221
6	2.7	24	+3.852E0		
7	3	15	-8.091E-2	b0	48.2339394
8	3.3	6	-3.624E0	b1	-4.56111111
9	3.6	.8	-2.978E0	b2	-2.16329966
10	3.9	.5	+2.958E0	b3	0
11				b4	0

Here is the resulting plot of the curve through the data points.



Summary

If you need to solve a system of linear equations as part of a TK model, it may be best to simply type the equations as is and let the iterative solver do the work. If you need to do curve-fitting or some other task that requires matrix processing, it's possible that the TK Library includes ready-made tools for the task with no programming required. If you need to prove to your college instructor that you know how to translate a system of linear equations into algebraic notation in order to get a good grade, use a mathematics language.