A p p e n d i x

Frustum of a Cone

1. The basic FRUSTUM model

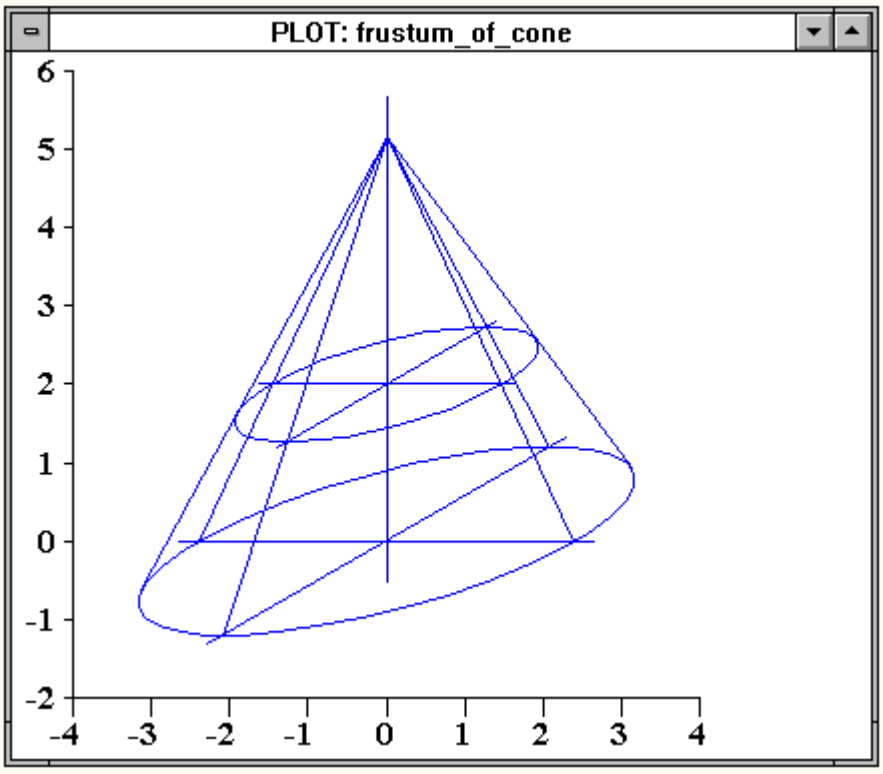
The popular cone model is too simple to illustrate the "laws of TK". The model of a frustum of a cone may do a better job. It has been derived by considering two cones: a cut-off upper cone (variables with suffix 1) and a complete cone (variables with suffix 2). The angle of opening *th* is common for both. A few other variables (also without suffix) apply specifically to the frustum. There are the following 15 independent equations in the Rule Sheet:

```
=================== RULE SHEET ============================================
S Rule-------------------------------------------------------------------------
  s1^2 = r1^2 + h1^2       ; equations for the cut-off top cone
  r1/h1 = tand(th/2)
  AB1 = pi() * r1^2
  AC1 = pi() * r1 * s1
  V1 = AB1 * h1 / 3
  s2^2 = r2^2 + h2^2       ; equations for the complete cone
  r2/h2 = tand(th/2)
  AB2 = pi() * r2^2
  AC2 = pi() * r2 * s2
  V2 = AB2 * h2 / 3
  h = h2 - h1              ; frustum of a cone
  s = s2 - s1
  AC = AC2 - AC1
  AT = AB1 + AB2 + AC
  V = V2 - V1
```

There are 18 variables. According to the 1st law, 18-15=3 variables must be assigned input values in order to determine the solution of a particular problem. Here is the Variable Sheet with given *th*=50, *h*=2 and *r2*=2.4, and 15 output values resulting from the resolution of 15 independent equations:

```
=================== VARIABLE SHEET ========================================
St Input---- Name--- Output--- Unit---- Comment------------------------------
    50.      th                deg      angle of opening at vertex
             s          2.2068          distance from base to top perimeter
     2.      h                          distance between base and top
             AC        26.8115          area of curved surface of frustum
             AT        51.6716          total surface area of frustum
             V         23.9493          frustum volume
             s1         3.4721          distance from top perimeter to vertex
             r1         1.4674          radius of top
             h1         3.1468          distance of top of frustum to vertex
             AB1        6.7645          area of top
             AC1       16.0062          area of curved surface of cut-off cone
             V1         7.0956          volume of cut-off cone
             s2         5.6789          dist from frustum base perim to vertex
     2.4     r2                         frustum base radius
             h2         5.1468          distance from base center to vertex
             AB2       18.0956          area of base
             AC2       42.8178          area of curved area of complete cone
             V2        31.0449          volume of complete cone
```

Here is a cavalier-projection of the complete cone and frustum according to the above solution:



2. Combinations of input and output variables

There are, theoretically, 18!/((18-3)! 3!)=816 combinations of 3 input variables out of 18. Practically the number of combinations may be reduced, sometimes by half or more, because many combinations happen to be infeasible. It is obvious that, for instance, *r2*, *h2* and *th* cannot be given arbitrary input values because they would make the 7th equation **`r2/h2=tand(th/2)`** inconsistent. It is less obvious, that assigning input values to *th*, *s1*, *AB1* leads to the same trouble: it will make the first 4 equations with unknowns *r1*, *h1*, *AC1* overdetermined, and the 2nd equation **`r1/h1=tand(th/2)`** will be marked as inconsistent. On the other hand, even after substituting given values of *th*, *s1*, *AB1* and evaluated *r1*, *h1*, *AC1* into the remaining 11 equations, there still will be 12 unknowns*: s, h, AC, AT, V, V1, s2, r2, h2, AB2, AC2, V2*; i.e. that part of the model will be left unresolved because it is underdetermined.

There will be more discussion about feasible and infeasible combinations of the input and output variables later.

3. VEC matrices

The next three tables illustrate the concept of VEC matrices and the third law of TK. First, there is a complete VEC matrix of the model FRUSTUM. The x's indicate which variables appear in which equations. There are no symbols "#" which would indicate that the particular equations is not resolvable with respect to the coincident variable.

The next is a reduced VEC matrix which may be obtained by excluding the rows with input variables for a particular problem (it is *th*, *h* and *r2* in our example). Finally, the third table shows the reduced matrix triangulized by reshuffling rows and columns in such a way that cells below the NW-SE diagonal become empty.

FRUSTUM.TKW, complete VEC-matrix

Eq-ns:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
th		x					x								
s												x			
h											x				
AC													x	x	
AT														x	
V															x
s1	x			x								x			
r1	x	x	x	x											
h1	x	x			x						x				
AB1			x		x									x	
AC1				x									x		
V1					x										x
s2						x			x			x			
r2						x	x	x	x						
h2						x	x			x	x				
AB2								x		x				x	
AC2									x				x		
V2										x					x

FRUSTUM.TKW, given th, h, r2; reduced VEC-matrix

Eq-ns:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s												x			
AC													x	x	
AT														x	
V															x
s1	x			x								x			
r1	x	x	x	x											
h1	x	x			x						x				
AB1			x		x									x	
AC1				x									x		
V1					x										x
s2						x			x			x			
h2						x	x			x	x				
AB2								x		x				x	
AC2									x				x		
V2										x					x

FRUSTUM.TKW, given th, h, r2; triangulized VEC-matrix

Eq-ns:	7	8	10	11	2	3	5	6	9	15	1	4	12	13	14
h2	x		x	x				x							
AB2		x	x												x
V2			x							x					
h1				x	x		x				x				
r1					x	x					x	x			
AB1						x	x								x
V1							x			x					
s2								x	x				x		
AC2									x					x	
V										x					
s1											x	x	x		
AC1												x		x	
s													x		
AC														x	x
AT															x

The diagonal elements show what variables are evaluated from which equations. The column heading shows how many times the Direct Solver cycles through the Rule Sheet; it is three times in our example. The Direct Solver solves first the equations 7,8,10,11, then 2,3,5,6,9,15, and finally 1,4,12,13,14.

4. Head, body and tail

Let's remove the constraint on h and impose one on the curved surface of the frustum by assigning an input value AC=25. The Direct Solver cannot cope with the problem any more. Here is how far we can get by trying to triangulize the VEC matrix:

FRUSTUM.TKW, given th, AC, r2; partially triangulized VEC-matrix

Eq-ns:	7	8	10	6	9	13	1	2	4	3	5	11	12	14	15
h2	x		x	x								x			
AB2		x	x											x	
V2			x												x
s2				x	x								x		
AC2					x	x									
AC1						x			x						
s1							x		x				x		
r1							x	x	x	x					
h1							x	x			x	x			
AB1										x	x			x	
V1											x				x
h												x			
s													x		
AT														x	
V															x

The head, body and tail of the VEC matrix are clearly defined. Here is the 3x3 VEC matrix of the body (or 3 simultaneous equations in 3 unknowns):

Eq-ns:	1	2	4
s1	x		x
r1	x	x	x
h1	x	x	

Selecting any of the three unknowns as a guess variable makes it possible, in the given example, to triangulize the VEC matrix and proceed. We will see this exemplified by the entire model in the next subsections.

There may be several bodies $p\times p$, $q\times q$, $r\times r$, . . . ordered along the diagonal like beads on a string. It is possible in principle (and practically, for instance, using macros), to handle one partial system of simultaneous equations after another using a minimum number of guess variables for each (except for 1×1 beads that can be handled by the Direct Solver, providing there is an x rather than a # in the diagonal cell). The present TK Solver does not do this automatically. Consequently, all the bodies have to be solved simultaneously with whatever number of guess variables it takes.

The tail is presented in the partially triangulized VEC matrix in a way in which it would be handled by the Direct Solver after evaluating the head variables and solving the body of the simultaneous system. It is not the way of reversed triangulization described in the commentary to the fifth law of TK as a means for identification of the remaining body of the simultaneous system. If we follow the description (as does the TK utility procedure performing the reverse triangulization), and reshuffle the rows and columns as we go, the tail looks as follows:

Eq-ns:	12	11	14	15	5	3
s	x					
h		x				
AT			x			
V				x		
V1				x	x	
AB1			x		x	x

Using the reverse triangulization resulted in making the upper right triangle blank, which is just another way to ascertain that after all other unknown variables are evaluated, the remaining ones (6 in our case) could be taken care of by the Direct Solver.

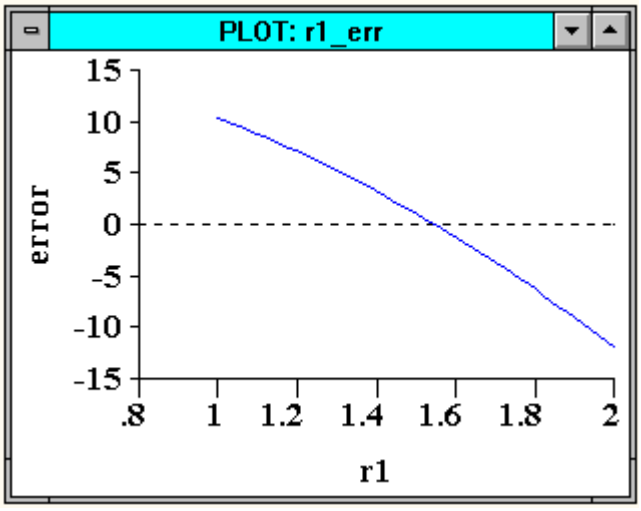
5. Direct solution of a model with error terms

Following the sixth law of TK we break the impasse by making *r1* an input variable without changing its value of 1.4674. Once *r1* is considered known, we remove its row from the VEC matrix. An attempt to solve pressing F9 leads to inconsistency in the equation on line 13 (3rd from the bottom). We can edit in an error term, i.e. **`err+AC=AC1-AC2`** . The problem becomes solvable by the Direct Solver as may be seen from the triangulized VEC matrix on the next page.

FRUSTUM.TKW, given th, AC, r2; r1 assigned input value and an error term err added to eq-n 13; triangulized VEC-matrix

Eq-ns:	2	3	5	7	8	10	11	14	15	1	4	6	9	12	13
h1	x		x				x			x					
AB1		x	x					x							
V1			x						x						
h2				x		x	x					x			
AB2					x	x		x							
V2						x			x						
h							x								
AT								x							
V									x						
s1										x	x			x	
AC1											x				x
s2												x	x	x	
AC2													x		x
s														x	
err															x

Introduction of error terms and their evaluation by the Direct Solver as stipulated by the sixth law of TK is quite helpful in studying their dependency on the extra input variables or guess-candidates in the process of looking for solutions of complicated and large systems of nonlinear simultaneous equations. What we present here is a simplistic example because the simplicity of the FRUSTUM model does not justify using this strategy. Yet, in order to complete the example, we used list solving to generate data for the graph of the function *err* = f(*r1*) in the interval $1 \le r1 \le 2$. It is clear from the graph that choosing a guess value of *r1* quite far from the final solution (*r1*=1.5482) will still ensure fast convergence:



6. Iterative solution

We constrain the error term by assigning an input value *err*=0, make *r1* a guess variable (its previous value 1.4674 is a good guess), and solve.

7. *Redundant equations*

It is possible to avoid using the Iterative Solver for solving problems with given *th*, *AC* and *r2* by adding to the system of equations an additional one, which could be derived from the original 15 equations by algebraic manipulation. The purpose: making it possible to triangulize the VEC matrix or, in other words, to solve the problem by the Direct Solver. Once the equation is derived from the other ones, it is not independent; it does not affect the determinacy (i.e. overdeterminacy or underdeterminacy) of the system of equations, hence the term "redundant".

A useful redundant equation may be derived from the system of three equations (the 1st, 2nd and 4th rule):

$$s_1^{\,2} = r_1^2 + h_1^2$$
$$r_1 \,/\, h1_1 = \tan(\theta / 2)$$
$$AC_1 = \pi \cdot r_1 \cdot s_1$$

by eliminating *s1* and *r1*. The resulting equation, **`AC1/pi()=h1^2*tand(th/2)/cos(th/2)`** , is then added to the Rule Sheet as equation 16.

FRUSTUM1.TKW, given th, AC, r2 and redundant eq-n 16; triangulized VEC-matrix

Eq-ns:	7	8	10	6	9	13	16	2	3	4	5	11	12	14	15	1
h2	x		x	x								x				
AB2		x	x											x		
V2			x												x	
s2				x	x								x			
AC2					x	x										
AC1						x	x			x						
h1							x	x			x	x				x
r1								x	x	x						x
AB1									x		x			x		
s1										x			x			x
V1											x				x	
h												x				
s													x			
AT														x		
V															x	

We can see that the redundant equation 16 is used for evaluation of *h1* whereas the equation 1 doesn't play any active role; the Direct solver just checks that it holds.

There was an avalanche of VEC matrices on the last five pages. One would not need them and worry about them during routine use of TK. The general purpose of showing so many of them here is to illustrate the inner workings of TK and, specifically, the flexibility with which the Direct Solver changes the order of evaluation depending on the distribution of given (input) and unknown (output) variables.

8. Symbolic manipulation

The equations in the Rule Sheet may be streamlined and the redundant equations described in previous section may be derived with help of the computer algebra programs, such as Macsyma, Maple or Mathematica. However, it is easier and faster, in simple situations, to manipulate the equations manually using the TK Rule Sheet as a scratchpad and the Direct Solver as an audit tool. You find some particular solution (e.g. using the Iterative Solver) and overdetermine the problem by moving some or all output values to input so that the Direct Solver always works upon pressing F9. Then you copy the set of rules to work on and use the cut/copy/paste and other edit commands for substitutions, simplifications, etc.

You keep the intermediate equations until done, and you press F9 after each step so that TK checks whether the equalities still hold. If there was any mistake or typo, TK will flash the inconsistency message and you inspect your work, fix the problem and press F9 again. It is also a good idea to watch the end of the Variable Sheet to make sure that no "new variable" shows up due to a typo when modifying the equations. That would render the integrity checks inoperative.

Here is an annotated portion of the Rule Sheet shoving the sequence of actions needed to derive the redundant equations in the previous section:

```
; original equations:
      s1^2 = r1^2 + h1^2
      r1/h1 = tand(th/2)
      AC1 = pi() * r1 * s1
; substituting for s1 in the 3rd equation from the first
      AC1 = pi() * r1 * sqrt(r1^2+h1^2)
; dividing both sides by h1^2
      AC1/h1^2 = pi() * r1/h1 * sqrt(r1^2/h1^2+1)
; substituting for r1/h1 from 3
      AC1/h1^2 = pi() * tand(th/2) * sqrt(tand(th/2)^2+1)
; applying trigonometric identity:
      AC1/h1^2 = pi() * tand(th/2) / cosd(th/2)
```

9. Invertible cone model

Adding one or even two redundant equations to the FRUSTUM model enhanced the direct resolvability, but more half of the feasible combinations of input and output values still requires the Iterative Solver. Let us, on one hand, simplify the model by handling the almost identical equations for the top and complete cones in a single rule function. On the other hand, let us revise the single cone model or function and supplement it with redundant equations as to make it fully invertible or directly resolvable for any feasible distribution of input and output variables.

We start with taking the FRUSTUM model and focusing at the 5 independent equations with 7 variables related to one of the cone (we choose the top one). The rest of the model could be deleted or canceled. Next, we analyze the reduced model with the VECOMB utility. There are 21 combinations of 2 input and 5 output variables out of which one is infeasible (given *r1,AB1*), 14 can be solved by the Direct Solver and 6 require the Iterative Solver. The latter group include the combinations with the following pairs of given variables:

th,s1 *th,AC1* *th,V1* *s1,V1* *h1,AC1* *AC1,V1*

For each these pairs of given variables there must be at least one redundant equations which contains that pair plus another single variable; the redundant equation must be resolvable with respect to that variable so that when the unknown variable is evaluated, the solution propagates freely through the rest of equations. A partial Rule Sheet demonstrates the course of development of the redundant equations:

```
================= RULE SHEET ==========================================
S Rule-------------------------------------------------------------------
  ; Development of the invertible function cone1:

  s1^2 = r1^2 + h1^2       ; eq-ns for the cut-off top cone
  r1/h1 = tand(th/2)
  AB1 = pi() * r1^2
  AC1 = pi() * r1 * s1
  V1 = AB1 * h1 / 3

  s1^2 = r1^2 + h1^2       ; 1
  r1/h1 = tand(th/2)       ; 2
  AC1 = pi() * r1 * s1     ; 4
  V1 = pi()*r1^2*h1/3      ; 6  AB1 eliminated from 3 and 5

  h1^2 * (1+tand(th/2)^2) = s1^2              ; 7  r1 eliminated from 1,2,4,6
  AC1^2 / s1^2 / pi()^2 = h1^2 * tand(th/2)^2 ; 8
  3*V1*s1^2 = AC1^2 * h1 / pi()               ; 9

  AC1 = sind(th/2) * s1^2 * pi()              ; 10  h1 eliminated from 7,8,9
  AC1^3 / pi()^2 = 3* V1 * s1^3 * tand(th/2)  ; 11

  sind(th/2)^2*cosd(th/2) * s1^3 * pi() = 3* V1   ; 12: AC1 elim. from 10,11
  cosd(th/2)^2*sind(th/2) * AC1^3 = 9*pi() * V1^2 ; 13: s1 elim. from 10,11
  AC1 * cosd(th/2) / tand(th/2)  = pi() * h1^2    ; 14: s1 eliminated from 7,8

  ; Last three are the redundant equations
```

The first group represents the original 5 equations of the top cone. The variable *AB1* doesn't appear in any of the six pairs and thus it is the first candidate for elimination which reduces the set of 5 equations to 4 in the second group. Next, we eliminate *r1* and then *h1* obtaining the third and fourth group of 3 and 2 equations respectively. Eliminating *AC1* from the fourth group produces a redundant equation that takes care of the pairs *th,s1* and *th,V1*. Similarly, eliminating *s1* takes care of *th,V1* again, and also of *th,AC1*. That still leaves the last three pairs unattended. One problem is the fact that neither of the derived redundant equations can be resolved with respect to *th*. If that were possible, the pairs *s1,V1* and *AC1,V1* would be covered, but the *h1,AC1* wouldn't. That points to the need of another, third redundant equation.

What to do about the invertibility of the redundant equations with respect to the variable *th*? The answer is simple: the expressions containing *th* make for a smooth monotonous functions in the interval of *th*/2 from 0 through 90 degrees, that can be handled by the invertible TK list functions with cubic interpolation. Generating the range list by listsolving from the domain list with 1 degree increments provides sufficient accuracy commensurable with the precision of the solution propagation through the whole model.

Here is the resulting custom-built rule function **`cone1`**:

```
 ==================== RULE FUNCTION: cone1 ==================================
Comment:              Cone geometry, 7 vbls, fully invertible
Parameter Variables:
Argument Variables:   th,s,r,h
Result Variables:     AB,AC,V
S Rule-------------------------------------------------------------------------
  ; Supporting list functions: s2cd(x) = sind(x)^2*cosd(x)
  ;                           c2osd(x) = cosd(x)^2/sind(x)
  s^2 = r^2 + h^2
  r/h = tand(th/2)
  AB = pi() * r^2
  AC = pi() * r * s
  V = AB * h / 3
  s2cd(th/2) * s^3 * pi() = 3 * V       ; redundant eq-n
  s2cd(90-th/2) * AC^3 = 9*pi() * V^2   ;     "      "
  AC * c2osd(th/2) = pi() * h^2         ;     "      "
```

Now, the groups of equations dealing with the top and complete cones can be replaced by two calls to the rule function **cone1** . The five equations specifically for frustum remain the same. We just threw in "for good measure" another two redundant equations which may improve the direct resolvability of the model for some "popular" combinations of inputs and outputs. These equations could be derived from the first two frustum equations and some of the top and complete cone equations; they also can be derived directly from the frustum geometry. Here is the resulting Rule Sheet:

```
==================== RULE SHEET ======================================
S Rule-----------------------------------------------------------------
  call cone1(th,s1,r1,h1;AB1,AC1,V1)  ; eq-ns for the cut-off top cone
  call cone1(th,s2,r2,h2;AB2,AC2,V2)  ; eq-ns for the complete cone
  h = h2 - h1               ; frustum of a cone
  s = s2 - s1
  AC = AC2 - AC1
  AT = AB1 + AB2 + AC
  V = V2 - V1
  s^2 = (r2-r1)^2 + h^2     ; redundant equation
  h/s = cosd(th/2)          ;     "        "
```

The VECOMB analysis of the simplified model FRUSTUM2, which is backed-up by the fully-invertible function **cone1,** shows not only further increases the number of directly resolvable combinations of input and output variables (from the original 111 out of 816 minus 92 infeasible, to 426 out of 816 minus 97 infeasible). The invertibility of **cone1** and the additional redundant equation also helps with the remaining 293 cases which still need the Iterative Solver. All of them (with a handful of exceptions) now require just one guess variable instead of two before.

It may be of interest to notice that the reference-book formulas for the frustum of a cone, such as

$$AC = \pi\left(r_1 + r_2\right) \cdot s$$

$$AT = \pi\left(r_1 + r_2\right) \cdot s + \pi r_1^2 + \pi r_2^2$$

$$V = \frac{\pi}{3} h\left(r_1^2 + r_2^2 + r_1 r_2\right)$$

are all implicitly available for computation by the Direct Solver in the model FRUSTUM2.