BACK

Using DERIVE in Analytic Geometry

Klaus Aspetsberger

Stiftsgymnasium Wilhering
Linzerstraße 8, 4073 Wilhering

Beside the classical use of Computer Algebra Systems for symbolic differentiation and integration **Derive** can be used for analytic geometry, too. Here one can use **Derive** for doing the symbolic calculation e.g. solving equations as well as for demonstrating geometrical objects in 2D-plot windows. Functions for manipulating vectors can be defined by the user.

1. Analytic Geometry for 2-dim Objects

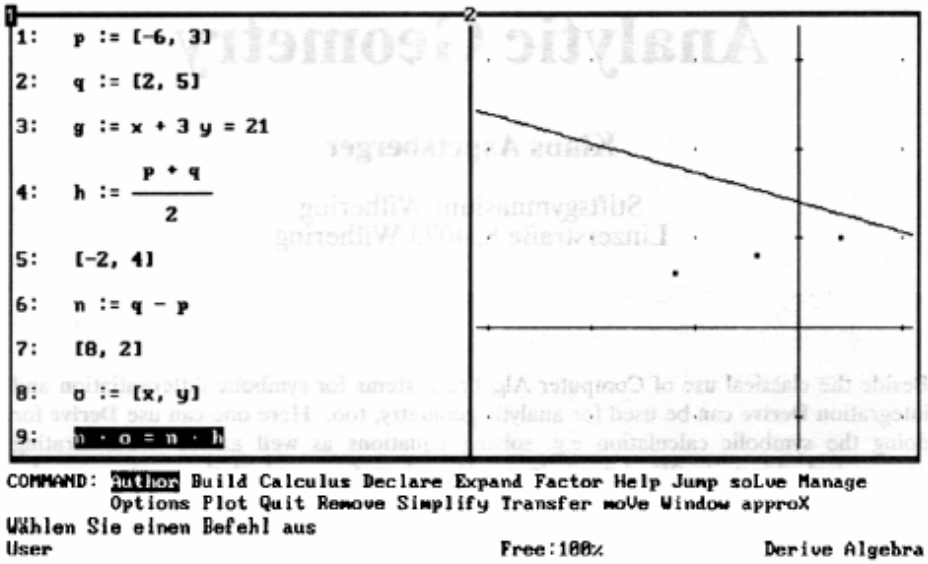
Vectors can be entered in **Derive** using the brackets [and]. Vectors are allowed to have arbitrary dimension and the elements can be symbolic expressions. In **Derive** there are several operators for vectors predefined. Use the + operator for adding two vectors, the - operator for subtracting vectors, the infix operator . to compute the inner product and finally the prefix operator CROSS to compute the cross product of two vectors. Thus, it is easy to manipulate vectors in **Derive**.

Furthermore 2-dim vectors can be plotted in 2D-plot windows. This feature is very usefull for the use of **Derive** in math classes. Thus, it is possible to visualize graphically the effects of algebraic manipulations.

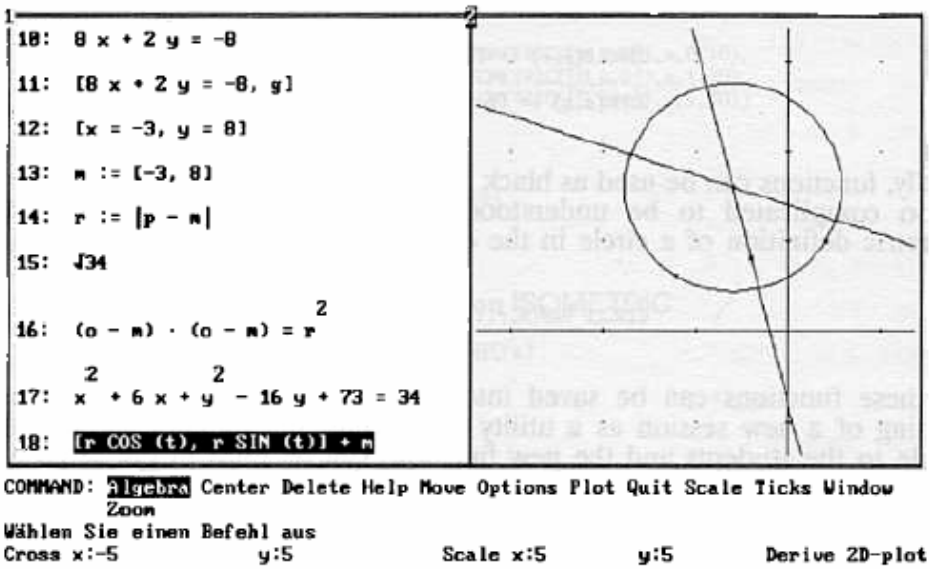
It is preferable to change the input mode to Word using the **Options Input** command and to split the screen into two windows. The first one should be designated to be a algebra window. Here we will do all the algebraic manipulations. The second one should be designated to be a 2D-plot window for visualising the geometric objects.

Example:

Find the equation of a circle having a center lying on the straight line g: x+3y = 21. The points P(-6 / 3) and Q(2 / 5) should lie on this circle.



First we assign the given values to the variables *p* and *q*. It is also possible to assign symbolic expressions to variables. Thus, we assign the equation of the straight line to the variable *g*. Both point and the straight line can be plotted in the 2D-plot window. For finding the center of the circle we first compute the axis of symmetry for the two points P and Q. For this purpose we enter the formula for the middle of line PQ in line 4 using the **Author** command and simplifying this expression we obtain expression 5. In the same way we enter the formula for the vector connecting P and Q (line 6 and line 7). This vector lies perpendicular to the axis of symmetry, whoose normal form is entered in line 9. Simplifying this expression we obtain the equation for the axis of symmetry. Expression 11 is a system of the linear equations describing the intersection of the axis of symmetry and straigth line g. This system of equations can be solved with the **soLve** command obtaining the center of the circle M (line 12 and line 13). The radius of the circle can be computed applying the absolute value function ABS to the vector P - M (expressions #14 and #15). Line 16 presents a general formulation of the equation of a circle. Simplifying this expression we obtain the actual equation of the circle. For easier plotting the circle is formulated in parameter depending form.



Notice: lines 4, 6, 8, 9 etc. do not depend on actual values. These expressions can be interpreted being a program.

2. Defining Functions

In **Derive** exist a lot of predefined functions. However, it is very easy for a user to define some functions by himself. Some of the following examples are so easy, that they can be defined by pupils. The more complicated ones should be defined by the teacher and can be used as black boxes.

There are different reasons for defining functions. Functions can be used for structuring the problem solving process. For example, one can use the following functions for defining straight lines

```
LINE_PAR(p,r,t) := p+t*r

LINE_NORM(p,n) := n . [x,y] = n . p
```

Furthermore it is easy to rename predefined functions to suit them to special situation in a math course. For example, one can introduce a function LENGTH computing the length of a vector *a* in the following way:

```
LENGTH(a) := ABS(a)
```

Functions can be used as abreviations. This is the most usual way of using functions. In the example above one might introduce functions for defining a circle or compute the axis of symmetry in the following way:

```
CIRCLE(m,r) := ([x,y]-m).([x,y]-m) = r^2

SYMM(p,q) := (p-q) . [x,y] = (p-q) . (p+q)/2
```

Finally, functions can be used as black boxes. The definitions of these functions are too complicated to be understood by the students. For example, the parametric definition of a circle in the example above could be used as a black box:

```
CIRCLE_PAR(m,r,t) := [r cos t, r sin t] + m
```

All these functions can be saved into a file, which could be loaded at the beginning of a new session as a utility file. Thus, the function definitions are invisible to the students and the new functions can be used in the same way as predefined functions.

3. 3-Dimensional Geometry

It is possible to plot three dimensonal objects. Expressions containing two variables can be visualized in a 3D-plot window. These objects are surfaces embedded in a three dimensional space. Thus, it is possible to visualize planes in 3D-plot windows. However, only one expression can be plotted in a 3D-plot window. Therefore the intersection of two or more planes can be only visualized very makeshift. It is also impossible to plot points and straight lines in a 3D-plot window.

Since Version 2 the **Derive** diskette contains several utility files. One of them is GRAPHICS.MTH. The file GRAPHICS.MTH defines functions for the isometric projection. Using this projection three dimensional objects are projected into a two dimensional plane and can be plotted in a 2D-plot window. Thus, three dimensional points, space curves and parametric surfaces can be plotted in a 2D-plot window. In 2D-plot windows several objects can be plotted at the same time.

At the beginning of a session use the **Transfer Load Utility** command to load the utility file GRAPHICS.MTH. The definitions of the several functions are loaded into the algebra window, however they are not displayed on the screen. Next use the **Window Split Vertical** command to split the **Derive** screen vertically into two windows and declare one of them as a 2D-plot window using the **Window Designate** command. In the 2D-plot window, use the **Options Color Plot** command to make the axes and the cross the same color as the background color. This makes the irrelevant 2D axes and cross invisible. For obtaining 3D axes enter in the algebra window

```
axes
```

and plot this expression in the 2D-plot window choosing a parameter going from a minimum value 0 to a maximum value 10. If there should be scaling points on the axes enter the following function:

```
pkte := [VECTOR(ISOMETRIC([a,0,0]),a,0,10),
         VECTOR(ISOMETRIC([0,a,0]),a,1,10),
         VECTOR(ISOMETRIC([0,0,a]),a,1,10)]
```

Simplify

```
kreuz := [axes,pkte]
```

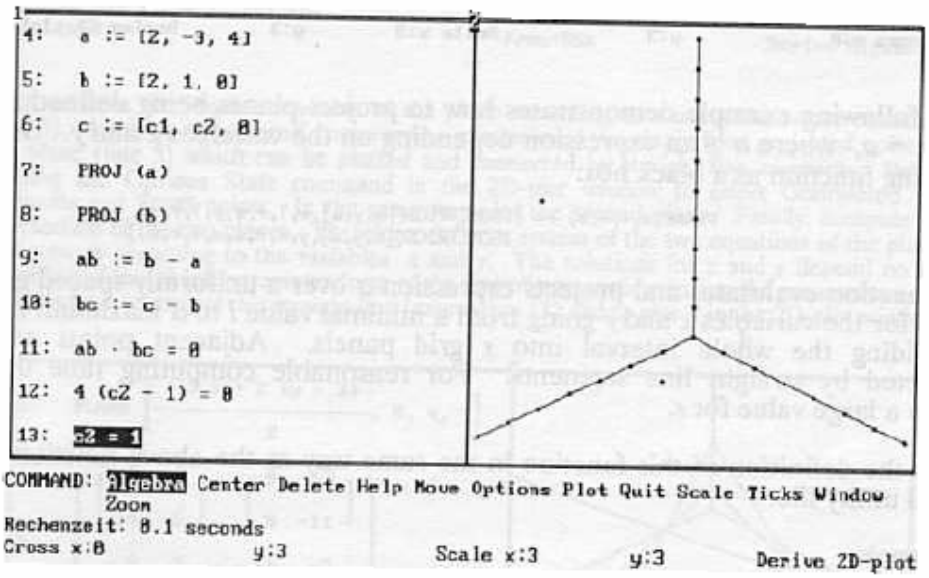
for obtaining an expression realizing both.

For an easier notation rename the function ISOMETRIC

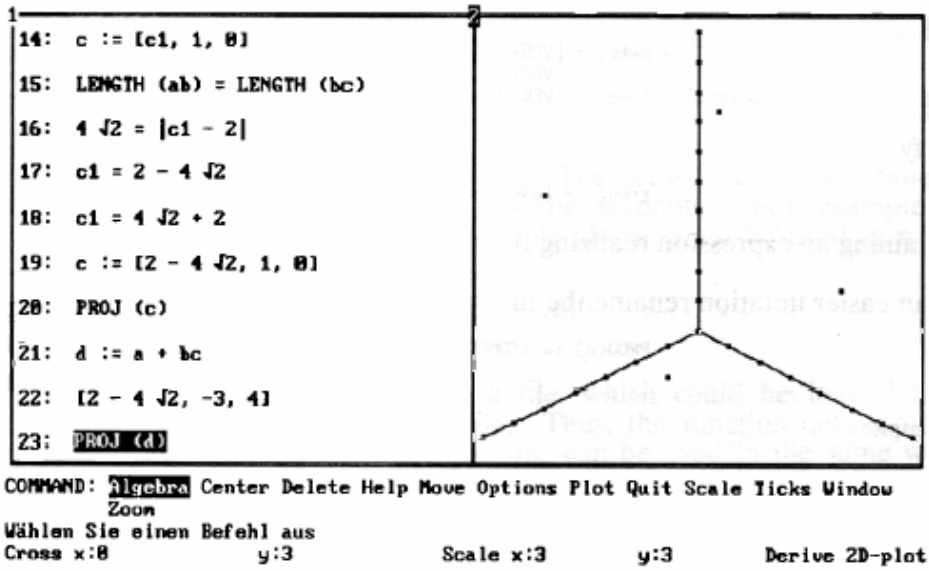
```
PROJ(a) := ISOMETRIC(a
```

Example:

A square is defined by the points A(2,-3,4) and B(2,1,0). Find corner C(c1<0,c2,0) and D!



First enter points *A*, *B* and *C*. For entering the elements *c1* and *c2* of *C* select input mode Word using the **Options Input** command. In line 7 and line 8 *A* and *B* are projected, respectively, and can be plotted in the 2D-plot window. *ab* and *bc* are the connecting vectors of *A* and *B* and of *B* and *C*, respectively. *A*, *B* and *C* are the corners of the square. Therefore, *ab* is perpendicular to *bc*. Thus, the scalar product of *ab* and *bc* is equal 0 (line 11). This is the first equation for determing *c1* and *c2*. Using the **Simplify** command we obtain an equation (line 12) for computing the value of *c2* using the **soLve** command. This result can substituted in line 6 using the **Manage Substitute** command. All sides of a square have equal length. Thus, line 15 presents the second equation for calculating *c1* and *c2*. Solving equation 16 using the **soLve** command, we obtain two solutions for *c1*. We select the first one being less than 0. Using **Manage Substitute** we substitute this result in line 14. We project *C* and plot it. In line 21 and 22 *D* will be computed and projected, respectively.



The following example demonstrates how to project planes being defined in the form $z = a$, where a is an expression depending on the variables x and y. Use the following function as a black box:

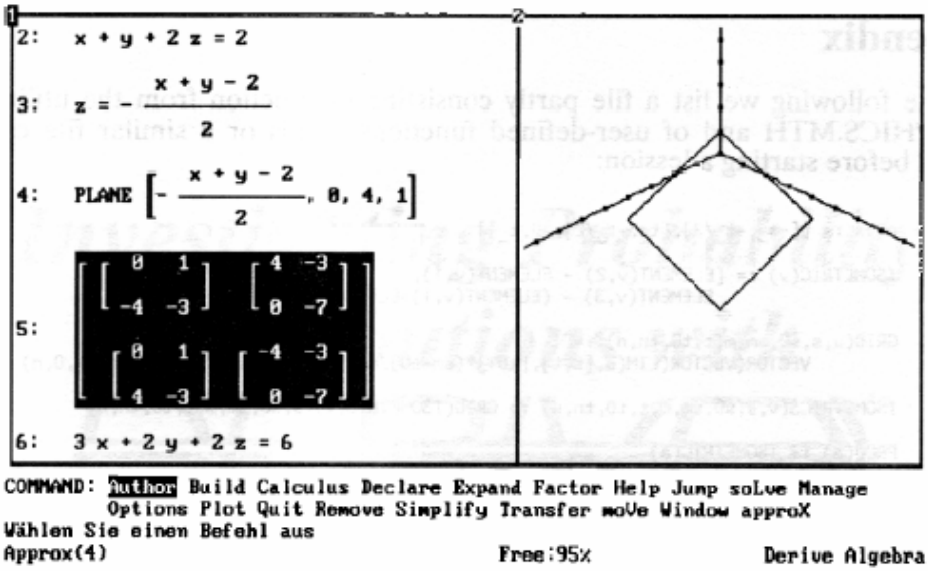
```
PLANE(a,l,r,s) := [ISOMETRICS([x,y,a],x,l,r,s,y,l,r,s),
                   ISOMETRICS([x,y,a],y,l,r,s,x,l,r,s)]
```

This function evalutates and projects expression a over a uniformly spaced grid of values for the variables x and y going from a minimal value l to a maximum value r subdividing the whole interval into s grid panels. Adjacent points will be connected by straight line segments. For reasonable computing time do not choose a large value for s.

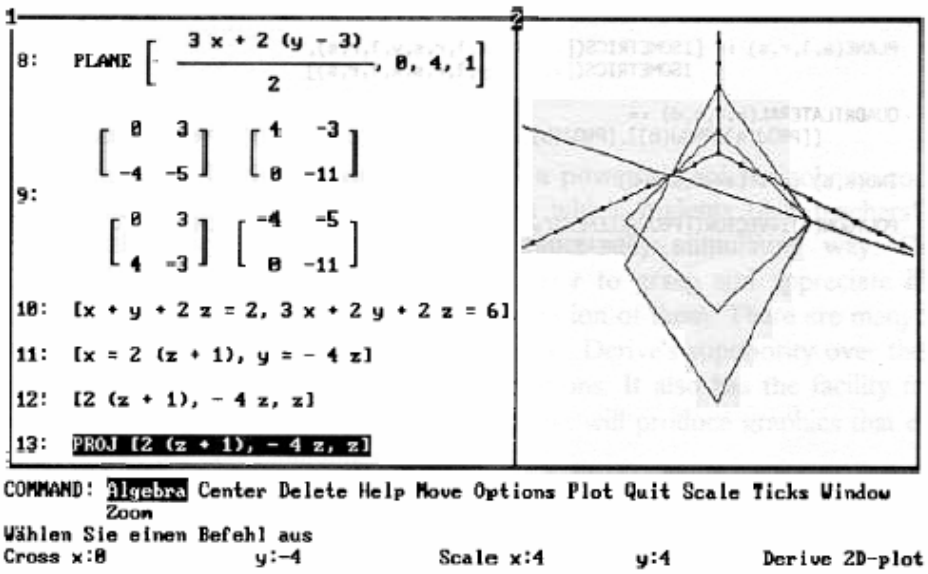
Save the definition of this function in the same way as the above functions in a special utility file.

Example:

Find the line of intersection of the planes $x + y + 2z = 2$ and $3x + 2y + 2z = 6$.



First we solve the implicit equation of the plane according to z using the **soLve** command (line 3). Applying the **approX** command to expression 4 we obtain four projected points of the plane (line 5) which can be plotted and connected by straight line segments. Before plotting use **Options State** command in the 2D-plot window to select Connected line segments and Small points. In the same way plot the second plane. Finally, compute the intersection of the two planes. Therefore, define a system of the two equations of the planes and solve it according to the variables x and y. The solutions for x and y depend on the variable z. Line 12 is a parameter form of the line of intersection having the parameter z. At least we have to project this straight line (expression 13) and to plot it in the 2D-plot window.



Appendix

In the following we list a file partly consisting of function from the utility file GRAPHICS.MTH and of user-defined functions. This or a similar file can be loaded before starting a lession:

```
axes := [[-t_,-t_/2],[t_,-t_/2],[0,t_]]

ISOMETRIC(v) := [ELEMENT(v,2) - ELEMENT(v,1),
                 ELEMENT(v,3) - (ELEMENT(v,1)+ELEMENT(v,2))/2]

GRID(u,s,s0,sm,m,t,t0,tn,n) :=
       VECTOR(VECTOR(LIM(u,[s,t],[s0+j*(sm-s0)/m, t0+k*(tn-t0)/n]),j,0,m),k,0,n)

ISOMETRICS(v,s,s0,sm,m,t,t0,tn,n) := GRID(ISOMETRIC(v),s,s0,sm,m,t,t0,tn,n)

PROJ(a) := ISOMETRIC(a)

pkte := [VECTOR(PROJ([a,0,0]),a,0,10),
         VECTOR(PROJ([0,a,0]),a,1,10),
         VECTOR(PROJ([0,0,a]),a,1,10)]

LINE_PAR(p,r,t) := p+t*r

LINE_NORM(p,n) := n . [x,y,z] = n . p

LENGTH(a) := ABS(a)

EQ(a,b) := VECTOR(ELEMENT(a,i)=ELEMENT(b,i),i,1,DIMENSION(a))

kreuz := [[[-t_,-t_/2],[t_,-t_/2],[0,t_]],
          [[[0,0],[-1,-1/2],[-2,-1],[-3,-3/2],[-4,-2],[-5,-5/2],
            [-6,-3],[-7,-7/2],[-8,-4],[-9,-9/2],[-10,-5]],
           [[1,-1/2],[2,-1],[3,-3/2],[4,-2],[5,-5/2],[6,-3],
            [7,-7/2],[8,-4],[9,-9/2],[10,-5]],
           [[0,1],[0,2],[0,3],[0,4],[0,5],[0,6],[0,7],[0,8],[0,9],[0,10]]]]

PLANE(a,l,r,s) := [ISOMETRICS([x,y,a],x,l,r,s,y,l,r,s),
                   ISOMETRICS([x,y,a],y,l,r,s,x,l,r,s)]

QUADRILATERAL(a,b,c,d) :=
      [[PROJ(a),PROJ(b)],[PROJ(b),PROJ(c)],[PROJ(c),PROJ(d)],[PROJ(d),PROJ(a)]]

IND(a,b) := IF(a<=b,a,a-b)

POLYGON(v):=VECTOR([PROJ(ELEMENT(v,i)),PROJ(ELEMENT(v,IND(i+1,DIMENSION(v))))],
                   i,1,DIMENSION(v))
```