UNIVERSITY OF VESZPRÉM

DEPARTMENT OF MATHEMATICS AND COMPUTING

SALT3DIM.exe - A PROGRAM FOR HANDLING 4 COMPONENT MIXTURES

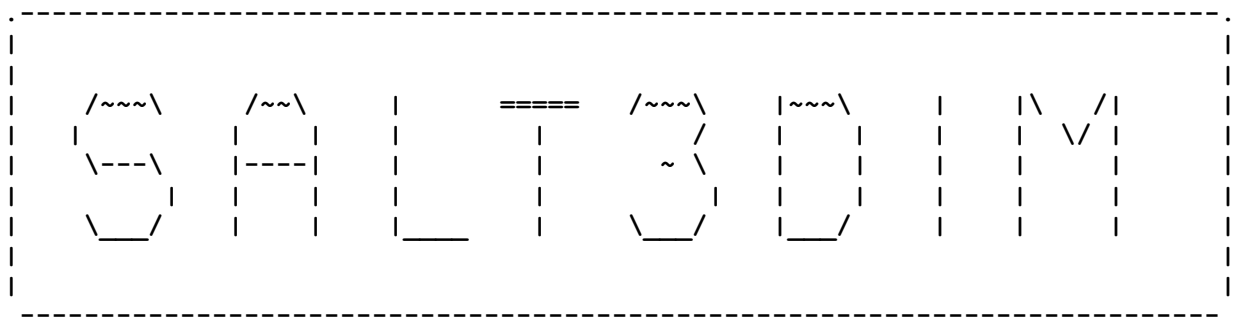
by

ISTVÁN SZALKAI
Department of Mathematics and Computing, University of Veszprém,
P.O. Box. 158, 8201 Veszprém, Hungary

PREPRINT No. 047

March 1996

VESZPRÉM
EGYETEM U. 10.
P. O. Box: 158.
H-8201



SALT3DIM

version 1.0 , 1995.dec.6.

A program for handling 4 component systems

User's Manual

Copyright by **Dr.István SZALKAI**
Dept.Math.,University of Veszprém,Hungary, H-8200
tel./fax.: (36) 88 - 423 239

email: *szalkai@almos.vein.hu*

This file (Salt3dim.hlp) contains general information and detailed User's Manual as well, of the program SALT3DIM.EXE (1995.dec.) . This file contains no extra ASCII characters and has length of 889 lines.

CONTENTS:
~~~~~~~~~

0 GENERAL PURPOSE and THEORY

This program offers a graphical and computer computational tool (first of all for chemics) for investigating more component systems. The present version is for quaterner systems.

In short, quaterner systems are mixtures, containing of, let us say, the components A,B,C and H, and if x_A, x_B, x_C and x_H denote the percent of the corresponding component, clearly we have

$$x_A + x_B + x_C + x_H = 1$$

and not to forget: these real numbers are between 0 and 1.

The usual graphical illustration of these mixtures in 3-dimension is the following: the inner points and the surface of a unit regular 3-dimensional simplex (tetrahedron) represent the different possibilities for mixing these four compounds, ie.the different possible values for the quadruples (x_A, x_B, x_C, x_H). Translating to the languagage of geometry: if the vectors, pointing from the origin to the vertices of the tetrahedron are denoted by a,b,c and h, then the mixture-phase (x_A, x_B, x_C, x_H) corresponds to the geometric point

$$\underline{p} = x_A * \underline{a} + x_B * \underline{b} + x_C * \underline{c} + x_H * \underline{h} \quad .$$

In this case mathematician call the numbers (x_A, x_B, x_C, x_H) the 'barycentric' coordinates of $\underline{p}$.

Well, since this stuff is wellknown (for chemics), or it is terrible (for non-mathematicians), LET'S START THE PROGRAM, and see what happens.

The General Purposes of the Program are:

(1) to Display Any (1,2 or 3) Sets of Given Data (given in %), in Cartesian Coordinate System,

(2) Approximate the Data Sets by Surfaces,

(3) Finding the Eutektic Lines and Point (ie. Finding the Pairwise Intersections of these Surfaces and with the Sides of the Tetrahedron as Well),

(4) Making Intersection of the Above Results With Planes, Representing Fixed (constant) %-Values of One of the Components,

(5) To Display the Projection of the Above Results to the [xy] Coordinate Plane

(6) Saving/Loading Your Computation Results to Continue Your Work after a break (be aware of blackouts) or Next Midnight.

This version is for computations and for education purposes only, so sometimes its appearance and limitations are not as brilliant as I could do! (Eg.there are not always warnings or 'try again's whenever You are mistaken!) Most of the error messages do not affect your computations.

1 THE FILES CONTAINED IN THE SYSTEM

are:

SALT3DIM.EXE	-	the exe file
*.BGI	-	Turbo Pascal graphic interface files
SALT3DIM.HLP	-	this file
SGOMB3_1.DAT	-	demo data file
SGOMB3_2.DAT	-	- " -
SGOMB3_3.DAT	-	- " -
SGOMB3-.3DI	-	demo final result from the above data sets for loading and peeping (a pentium PC at 150MHz worked 8 hrs for it)
SSIK3_1.100	-	other demo data files to use
SGOM3_2.100	-	- " -
SGOM3_3.100	-	- " - other demo data files to use

2 SOME TECHNICAL DATA

about the present version of the program:

Approx. 1945-1956 (!) lines source code in Turbo Pascal 6, runs even on an 286 AT with 4 MB RAM. The program can handle at most three (3) different sets of data, each of them may contain at most 1000 entries, approximating each data set can be refined up to 30x30 grid. The program can store and display up to 3x500 points of pairwise intersections of the approximating surfaces, and up to 3x2x x300 points of intersections of these surfaces with the sides of the tetrahedron (see function 'e' = Eutectic intersections in 5.6, these points are called 'eutectic'). Furthermore, when making a 2-dimensional intersection of the 3-dimensional picture with function 'Alt-i' (intersection, see 5.18), the program can store and display up to 300 points of each surface (and all of the points falling in the neighbourhood of the intersecting plane of each data set and previously found eutectic points). Displaying and approximating requires some seconds on a 486 PC with 80 MHz, but getting intersections of both kind with fine resolution needs some hours. These and other limitations of the present quick-written version are because of the strict warnings of the Turbo Pascal system ('Structure too large', 'Code segment too large', 'Stack/heap overflow',etc.). Each improvement suggestions or other comments or bug reports from salt- or

dimension-fan users are appreciated and are honored with a newer version of the program, since improvements in all respects are in progress!

3 THE GRAPHICAL SCREENS

Hitting any key, you (almost) always will see the MAIN SCREEN: the coordinate axes, the tetrahedron, the cross-wires (denoted by +), and at the bottom: the coordinates of + and your eye. This is the single 3-dimensional picture in our program. For understanding some functions of the program, we shortly have to learn how this picture is built up, this will be described next.

The main picture is built as: we projected everything with lines from Eye to the plane, which is orthogonal to and halves the line segment between the fixed 'cross-wire' and Eye. Unfortunately, the cross-wire is not displayed on this screen, the sign + only shows the place you want to put next the cross-wire itself. (For how to move the sign + , see the function '1/Alt-1' in 5.14.) But don't worry: after redrawing the picture with one of the functions 'p' (picture redraw) or 'o' (original view) in 5.15 and in 5.16 resp., the sign + is placed exactly to where the cross-wire is, in fact!

After reading in data set(s) this screen also shows these data set(s) read in (in different colours), his/their domain(s) as smallest rectangle(s) containing the orthogonal projections of the data points. Moreover, after some computations you will see the interpolating surface(s), also in different colours, and after some more computations the eutektic lines and surfaces (in white).

The other two screens can be get by the functions 'Alt-p' (project to [xy] plane) and 'Alt-i' (intersection), they are described at those functions in 5.17 and 5.18, resp. These are 2-dimensional pictures.

4 THE MENU, ie. THE HELP SCREEN

Here you are the possible keys you should hit as you see in the program's built-in help screen (version 1.0, 1995.dec.6), which can be activated simply pressing the '?' key (whenever the main screen is displayed).

NOTES: do not mix lower and upper case letters!
- means hitting two buttoms at the same time
/ separates alternatives.

```
-------------------------------------------------------------------------------
f       = new dataFile read           |    Alt-f   = as f,but Cartesian coord.
b       = Bulb-size data plot on/off  |
d/D     = modify approximation Domain |
a       = Approximate last datafile   |
e       = find Eutektic intersections |    Alt-l   = draw eutectic Lines on/off
g/G     = Grids on/off                |
m/M     = Move to/from +              |    <-/->   = rotate around z axis
z/Z     = Zoom                        | Curs up/dn = rotate around "x=-y"
E       = input new position for Eye  |
1/Alt-1 = move + along the x axis     |

            (similar with 2 and 3)    |
p       = Picture redraw in 3D        |
            (centered around + )      |
o       = Original view               |    Alt-p   = Project to [xy] plane
s       = Save picture & computation  |    Alt-i   = Intersect with const.data
l       = Load picture & computation  |    Ctr-End = End program
                                      |
?       = This help screen            |
-------------------------------------------------------------------------------
```

To choose a function, just hit the buttom(s) listed above. One letter in each function's above short description is always in UPPER CASE to remind you for the key(s) of the function (usually the same letter, in fact).

You are allowed to use these functions when the main 3-dimensional picture is displayed, and this picture is always redrawn in seconds (exceptions are the functions 'Alt-p' (projection to the [xy] plane,see 5.17) and 'Alt-i' (intersection,see 5.18)).

You might want to start with the last but one function 'l' (load, see 5.20) to see one final result of my work, then function 'Alt-p' (project to the [xy] plane, see in 5.17), and then with 'Alt-i' (intersection, in 5.18, choosing the preloaded 'previous' intersection screen), and then with ... well, try all of them a bit please before switching the computer off.

5 THE FUNCTIONS in the MENU

Now we separately describe each function below. You are warmly advised to read all of my warnings before trying them out, but after it, you are warmly recommended to try out all variants in details of that function. Once again note that do not mix lower and upper case letters !

Have a nice time!

5.0 ? = The help screen

When a function does not pop in your mind, just hit this key whenever the program displays the 3-dimensional main screen, and you'll see the above help screen. After hitting any key you return again for the main screen.

5.1 f = new dataFile read

The program can handle at most 3 (three) different sets of data. Choosing the 'f' (file read) function, you can read the next set of data, given in % quantities (mathematicians call them 'barycentric coordinates'). The (ordering) numbers of the data sets are counted and displaying automatically for your convenience, you do not have to bother them! (The similar is for their colors.)

The program offers you the forthcoming one from the three demo data files SGOMB3_1.DAT , SGOMB3_2.DAT and SGOMB3_3.DAT . So, hitting a pure Enter, you will read the next of them in order in an eyeblick. In this case be sure that these file are in the current directory.

If you want to read another file, you must give full filename with the extension, moreover, you should give it with full pathname in case your file is in other directory.

In case you give a nonexistent filename, after a red color warning you are asked 'Do you try another file(y/n)?'. Choosing 'y' (and hitting Enter) you get the next chance, while choosing 'n' (and Enter) the program returns to the unchanged 3-dim.main screen.

The dataset MUST be in the format below (otherwise the pogram halts with runtime error):

```
n               <= the number of data points, the first number in the file
a1 b1 c1 h1     <= four positive numbers between 0 and 1, sum of which is
a2 b2 c2 h2           1.0000  (% values of the components a,b,c and water),
a3 b3 c3 h3           separated by one or more spaces, Enter after each line
.  .  .  .                -"-
an bn cn hn     <= you need at least  n  many data lines
 ...            <= all the remainder lines are unread, may contain anything
```

The maximum size (n) of each data set may not exceed 1000, otherwise the program halts with Runtime error. The present version of the program does not check whether there are exactly n many data in the file, it simply runs and reads.

In case the sum of the data in any line is not exactly 1.000 you receive a red error message, but hopefully the program continues reading,you simply should hit any key.

After succesful reading of the first (1.) data set, the program asks you a name for an output file. This is a kind of report file (plain ASCII), all your computations can be read in it. The filename is up to you, but do not forget: the program ultimately overwrites the existing file with the same name in that directory! Again the program offers you a default name for this file. In case it suits you, you may just hit 'Enter'. This report file will be stored in the active directory unless you specify other pathname.

You also are asked whether you want to store in this output file the transformed data set or not. This means, you can see (and store in a file) among others the Cartesian (!) coordinates of your data. (Recall, that your data was originally given in % values (barycentric coordinates)!)

Now, after an OK sign, the program returns to the 3-dim.main screen and you can see your data set in the forthcoming color, along the previously read data sets in Cartesian coordinate system in the unit tetrahedron.

The smallest rectangle whose sides are parallel to the axes x and y , and which contains all the projections (ie.the (x,y)-coordinates) of the given data set, is drawn with the same color. The size of this rectangle can be adjusted with the function 'd/D' (domain, see 5.4), and plays role when approximating the (last read) data set (see 5.5 for approximating).

5.2 Alt-f = as f, but Cartesian coord.

This function is exactly the same as the above function 'f' , but the data to be read are supposeb to be given in (x,y,z) Cartesian coordinates. That means, the first three numbers of each line of the datafile will be read, and will not be transformed.

When the last dataset is read in with the function 'Alt-f', the tetrahedron is not displayed.

This function allows you to use this program for approximating ANY 2-variable function with the built-in approximation method of Shepard. (Well, this method is not the best, but do not forget: I had to choose a method for (randomly) scattered data points, since the chemics'dataset are such.)

5.3 b = Bulb-size data plot on/off

Hitting this buttom once (or more), all the data points are displayed with little bulbs (or with points again). So this buttom is a 'toggle', you can use it at any time.

This function do has effect for the projected picture (see 'Alt-p' in 5.17) but not for the intersection picture constructed by the help of the function 'Alt-i' (see in 5.18), and can only be adjusted when the 3 -dimensional main picture is displayed. (That is, use the bulb-function when the 3-dim picture is displayed and use 'Alt-p' afterwards.)

5.4 d/D = modify approximation Domain

Hitting 'd' or 'D' the size of the domain-rectangle for the last read in dataset (that is, the smallest rectangle whose sides are parallel to the axes x and y and which contains all the projections (the (x,y)-coordinates) of the given data set, drawn with the same color) can be adjusted, ie. it can be enlarged/make smaller.

Notify, that only the last-read dataset-rectangle can be modified, the others not any more! (Well, restart the program.) Moreover, there is no possibility for getting back to the original domain -- so be careful modifying the domain, since what is even more important: approximation of the last-read dataset will be done on this domain set, actually displayed in the screen (see 5.5 for approximating).

Though, resizing this domain at any time and after it a new approximation (only for the last-read dataset!), choosing the approximating function 'a' (see below) is always available.

5.5/0 Shepard's approximating method

The program's built-in approximating method (for the functions 'a'=approximation, 'e'=eutectic points and 'Alt-i'=intersection with constant data) i . Shepard's method we shortly describe here, to explain the program's query on 'Shepards exponent'. (R is for the set of real numbers.)

You are qiven a set of data points

$P_i = (x_i, y_i, z_i)$ for i=1,2,...,M

and you are looking for a two variable continuous function **U**:RxR -> R such that

(*) $\mathbf{U}(x_i, y_i) = z_i$ for i=1,2,...,M .

When the values (x_i, y_i) lie on a rectangle- grid, many approximating methods are (well) known. However, when these values are randomly chosen in the plane RxR, there are not so many. Our's is Shepard's Method: for any point P=(x,y) in RxR let

$$U(P) := \frac{\sum_{i=1}^{M} z_i \cdot \left(\prod_{j \neq i} r(P, P_j) \right)}{\sum_{i=1}^{M} \left(\prod_{j \neq i} r(P, P_j) \right)} = \frac{\sum_{i=1}^{M} z_i \cdot \frac{1}{r(P, P_i)}}{\sum_{i=1}^{M} \frac{1}{r(P, P_i)}}$$

where

$r(P, P_i)$:= the alpha 'th power of the Eucledian distance of the points P and P_i

and this alpha exponent can be any real number greater or equal than 1,of course differrent exponents give different approximating surfaces, all of them are continuous and give exact approximations (ie.satisfying (*)).

Using our program you are able to investigate and to compare these different surfaces for different values of the exponent 'alpha', as described below. We got best results with exponents around 10 , too small or too big exponents were not so good.

For further details see eg. **W.J.Gordon & J.A.Wixom's** paper *Shepard's Method of "Metric Interpolation" to Bivariate and Multivariate Interpolation* in the journal: Mathematics of Computation vol.32.(1978),pp. 253-264.

Though the functions 'a', 'e' and 'Alt-i' (approximation/eutectics/intersection) allow you to input different values for alpha, you are NOT advised to do so in the same session! Moreover, saving with the function 's' (=save,see 5.19) saves only the last value of alpha!

5.5 a = Approximate last datafile

This function combined with 'Alt-f' (reading dataFile, see.5.2) allows you to use this program for approximating ANY 2- variable function with the built-in approximation method of Shepard. Hitting the key 'a' the program starts

approximating the last read (!) datafile, approximating the others in not possible any more. (Well,restart the program.)

The approximating is done on the 'domain'-rectangle, so before starting approximating please adjust the size of this rectangle with the 'd/D'(=domain) function as described in 5.4.

So, after you hit the key 'a' the first you are asked for is Shepard's exponent ('alpha'). In case you give a value smaller than 1, you'll asked for this exponent again. (Recall from 5.5/0 that Shepard's exponent (alpha) is the parameter you can vary for our built-in approximation method.) Though the functions 'a', 'e' and 'Alt-i' (approximation/eutectics/intersection) allow you to input different values for alpha, you are NOT advised to do so in the same session! Moreover, saving with the function 's' (=save, see 5.19) saves only the last value of alpha! We got best results with exponents around 10 , too small or too big exponents were not so good.

The next question is 'Consider points only in the simplex? (y/n)'. Since some parts of our (infinite!) approximating surfaces are out of the tetrahedron, especially for large domains (whether you see it or not on the main screen) you may wish to deal only with the inner points of the tetrahedron (and with the points on its sides). Since checking this for every point makes the computation a bit slower, the answer is up to you (try out both).

Then you can input the (integer) numbers for the division of the sides of the domain-rectangle to make an equidistant grid on which the approximation will be carried out. Sorry, each of these numbers must not exceed 30 in the present version of the program. (Try to input higher numbers...)

The graphical file which name is asked for next will be a pure datafile containing the approximated values for your further use *of the next format*:

```
n                          <=  number of divisions along the  x  axis
m                          <=          - " -       along the  y  axis
x1   x2    .  .  .    xn   <=  the x-coordinates of the grid-points
y1   y2    .  .  .    ym   <=      y-coordinates     - " -
z11  z12   .  .  .    z1n
z21  z22   .  .  .    z2n
 .                         <=  the approximated values
 .
 .
zm1  zm2   .  .  .    zmn
```

The program offers you a default filename automatically, these names are different for the three data set read in! So, you can even simply press Enter at this prompt. Note, that any file (if exists) with the same name will be overwritten without any warning!

And finally, the computation begins, the numbers of the grid points and the approximated values are displayed in the forthcoming approximating color and the final picture appears on the main screen!

Some concluding remarks:

The approximation for the latest-read dataset can be repeated with other parameters and other domain-rectangle at any time, but remember: the previously computed values will be lost (for the latest-read dataset only).

The approximation results (for all datasets), displayed on the main screen can be stored and loaded again with the functions 's' (save,see 5.19) and 'l' (load,see 5.20). For example, you may wish to save before another approximation process.

The running time linearly depends on the size of the data set and on the number of the the grid points and slightly on alpha if the latest is an integer, but not for other alphas. (Why?) Displaying and approximating requires some seconds on a 486 PC with 80 MHz for the data sets containing 100 entries and for a 30x30 grid.

5.6 e = find Eutektic intersections
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

(Lower case letter 'e', please!)

This function searches the intersections of the approximating surfaces with the sides of the tetrahedron and their pairwise intersections, too. Well, if you choose this function when no data set is read in (with the 'f' or 'Alt-f' = read data File in functions, see 5.1 and 5.2), then a red message warns you to this fact, and a gray message gives the hope to you that the system is not halted.

The behaviour of the program while computing this function is very similar to the function 'Alt-i' (intersection,see 5.18) - you have to learn hard only one of them.

In case you have already made such an eutectic picture (or, equivalently, you loaded a saving file (see 5.20) containing such a picture as our demo file SGOMB3-.3DI does), that picture is already displayed on the main screen. Be careful: using this function again, no exit possibility exists: the program ultimately deletes that previous picture and begins to construct a new one (on top of all, the running time is always very long)!

Saving your 'eutectic'-pictures is possible with the 's' function (see 5.19) and can be loaded again withe the 'l' function (see 5.20).

To be more precise: the computation starts on the old picture - this makes possible for you to compare the two pictures during the computation, but not when it is finished, since the old picture is kept destroying). But before you have to input some parameter!

One more preliminary WARNING: there is no exit possibility after choosing this function while the computation even for reasonable parameters can be very long (hours), only at your socket! (I mean, switch the computer off.)

Since the program searches for the intersection points (of one surface with one side of the tetrahedron, or of two surfaces) step-by-step investigating all the possible points, moving on a grid of some step-length, first you are asked for this refinement step-length ('delta'). This has the important consequence that the running time linearly depends on (1/delta)^2 (the square of delta's reciprocial) ! May I advice you to choose delta=0.01 or so ?

Next you can vary Shepard's exponent ('alpha') for our built-in approximation method, which is described in details at the 'a' (approximation) function in 5.5. Though in the functions 'a' , 'e' and 'Alt-i' (approximation /eutectics/intersection) the program allows to you to input different values for alpha, you are NOT advised to do so in the same session! Saving with the 's' function (see 5.19) only saves the last value of alpha! The best-useable interval of alpha's is suggested at the 'a' (approximation) function in 5.5 : in general alpha=10 is the best.

The final questuion of the program is: 'Shall we consider points only in the simplex? (y/n)'. Since some parts of our (infinite!) approximating surfaces are out of the tetrahedron (whether you see it or not on the main screen), you may wish to deal with the inner and points of the tetrahedron (and with the points of its sides). Though checking this for every point makes the computation a bit slower, in this case you should hit 'y' (and Enter).

The (bit long) computation process can be followed on the screen: a little circle shows the point the program investigates for intersecting the forthcoming surface. (The program investigates your surfaces separately, this makes the whole process a bit quicker.) When the program finds an eutectic point, this circle leaves a white coloured dot at this place. Don't worry: after the procedure the whole new picture will be re-displayed! This way of showing also slowers the computation process, but at leat you do not have to worry about frozen machines.

All moving/zooming/turning possibilities you can use afterwards, though for every modifying the picture, the present version of the program redisplays the whole picture, which (on slower machines) makes slowly to you to go far away (in R^3 I mean).

The program can store and display up to 3x500 points of pairwise intersections of the approximating surfaces, and up to 3x2x300 points of intersections of

these surfaces with the sides of the tetrahedron. The number of the points the program finds depend on delta. In case the program would find more than 500/300 points for one surface, the first 500/300 of them will be stored and the program goes to the next surface (if left). Have a look at the moving point on the screen: it shows the order of investigating (and finding and storing) intersecting points!

The running time, as was indicated above, mainly depends linearly on (1/delta)^2 (=the square of delta's reciprocial) and the size of your data sets (in each point we investigate we have to make an approximation). Eg. choosing delta=0.005 needed 8 hours on a 486 PC with 150 MHz with the demo data sets, each of them containing 100 entries.

5.7 Alt-l = draw eutectic Lines on/off

Hitting these buttome once (or more), all the eutectic points (have been computed so far) are connected with lines in their (computing) order. Hitting these buttoms once again, these points will be displayed only which the default in our program. In other words, this function is again a 'toggle', you can use it at any time.

5.8 g/G = Grids on/off

Hitting the buttom 'g' (lower case) small x signs show 0.1 -unit length on the axes (between 0 and 1) and on the edges of the tetrahedron. These might help you to realise the sizes of the picture.

The upper case 'G' buttom hiddens these signs, but you can display or hid them at any time.

5.9 m/M = Move to/from +

This function moves your Eye to/from the cross-wire (see the explanation for building the main screen in section 3) and redraws the picture.

The starting coordinates for Eye are (20,30,25), the cross-wire is the center of the tetrahedron at the beginning of the program. Using the present function (and the present version of the program), Eye is moved along the line determined by the cross-wire and the previous position of Eye, BUT it is not checked if Eye is going to the other side of this line (well, if you want so...).

Do NOT mix this function with 'z' (zoom, see in 5.10) !

However keep in mind that the sign + itself, displayed at any time on the main screen, is NOT always the cross-wire itself (though its coordinates are always displayed along the coordinates of the Eye)! For further explanation see the functions '1/Alt-1' (moving +, in 5.14), 'p' (picture redraw,in 5.15) and 'o' (original view,in 5.16).

5.10 z/Z = Zoom

These buttoms zoom in/out your picture in/out with almost no limits. (Sorry for the computer 'under/overflow' technics ...)

In case the points you are interested in are moving out from your monitor, remember to use the functions '1/Alt-1' (moving +, see in 5.14) and then 'p' (picture redraw centered around +, see in 5.15).

5.11 <- / -> = rotate around z axis

With the 'CursorLeft' and 'CursorRight' buttoms you are able to move horizontally around the vertical z axis in the full 360 degree (or more).

5.12 Curs up/dn = rotate around the line x = -y

These cursor moving buttoms rotate the picture around the line in the [xy] coordinate plane (ie.z=0) with the equation x = -y . This does not only mean that you can move your Eye up and down, it is more! Combining this function with the left/right cursor buttoms, you may walk around your dataset(s) and their approximating surfaces.

Warning: if you move too close to the vertical z axis, and then move further in the same direction you may fall down on the other side of your horse: you will not recognize your picture at the first glance!

5.13 E = input new position for Eye

(Upper case letter 'E' , please!)

With this function you can directly modify the Eye's coordinates and to have a new view. (Recall the explanation how the main screen is biult up in section 3 and the roles of the cross-wire and the sign + described in 5.14.)

May I advice you to put down the previous (good!) position of the Eye before trying the present 'E' function, or at least to remember the 'o' (original view) function in 5.16 ? (The Eye's coordinates are always displayed along the co-ordinates of the sign + at the bottom of the screen in yellow.)

The starting coordinates for Eye are (20,30,25) and the original position for the cross-wire is the center of the tetrahedron when the program starts. However keep in mind that the sign + itself, displayed at any time on the main screen, is NOT always the cross-wire itself, they coincide only after redrawing the picture with one of the functions 'p' (picture redraw, see 5.15) or 'o' (original view, see 5.16), or restarting the program.

The Eye's position can also be modified with the 'm/M' (move) function (see 5.9).

5.14 1/Alt-1 = move + along the x axis (similar with 2,3,Alt-2 and Alt-3)

Of course I mean the key hittings '1' , '2' , '3' , 'Alt-1' , 'Alt-2' and 'Alt-3'. With these keys you can move the sign + in 6 directions in the 3 - dimensional space (and nothing else). Each step is approx. 0.1 unit. The exact position of the sign + is written at the picture's bottom in yellow (next to Eye's coordinates).

The sign + only helps you to plan the next position for the cross-wire, which is (in the present version of the program) is not visible, and does not move when moving + ! (Recall the explanation for building the main screen in section 3.)

But don't worry, after redrawing the picture with one of the functions 'p' (picture redraw, see 5.15) or 'o' (original view, see 5.16) the sign + is placed to the same place where the cross-wire is, in fact! So, to avoid con-fusion, you are advised to use the function 'p' (picture redraw, see in 5.15) as soon as it is possible!

5.15 p = Picture redraw in 3D (centered around +)

Hitting this key the cross-wire is set to the position of the sign + (but not conversely!) and the picture is redrawn so that the sign + with the cross-wire are positioned into the center of your screen. (Recall the explanation for building the main screen and the roles of the cross-wire and the sign + in sec-tion 3.)

The sign + can be moved with the function '1/Alt-1' (move + along the axes, see in 5.14).

5.16 o = Original view

Hitting this key all picture -displaying data are set to original default and you see the original picture like when the program started. No your data or your computation will be lost but only the way you were looking at them!

5.17 Alt-p = Project to [xy] plane

Hitting these keys all your data (with the domain sets in their adjusted sizes) along with the eutectic points are projected to the [xy] coordinate plane in an eyeblick. No moving/zooming/turning possibilities you have, only the bulb function (see 5.3) has an effect. Sorry, the present version does not project Eye, the + sign and the cross-wire. (I just felt like that.)

The function 'b' (bulb,see 5.3) do has effect for the projected picture but not for the intersection picture constructed by the help of the function 'Alt-i' (see in 5.18), and can only be adjusted when the 3-dimensional main picture is displayed. (That is, use the bulb-function when the 3-dim picture is displayed first, and use 'Alt-p' afterwards.)

Hitting the Spacebar you will return to the main screen.

5.18 Alt-i = Intersect with constant data

This function makes a 2- dimensional intersection of the 3-dimensional picture with a plane, parallel to one of the sides of the tetrahedron.

In the behaviour of the program while computing this function there are many similarities to the function 'e' (eutectics,see 5.18) - you have to learn only one of them hard.

In case you have already made such an intersection picture (or, equivalently, you loaded a saving file (see 5.20) containing such a picture as our demo file SGOMB3-.3DI does), the present version of the program asks you 'Display the previous picture (y/n)'. Answering 'y' the programs redisplays that intersection, but be careful: answering 'n' ultimately deletes that previous picture and begins to construct a new one! (The present version of our program can handle only one intersection picture.) Saving your intersections pictures is possible with the 's' function (see 5.19) and can be loaded again withe the 'l' function (see 5.20).

One more WARNING: there is no exit possibility after choosing this function while the computation even for reasonable parameters can be very long (hours), only at your socket! (I mean, switch the computer off.)

And now, what is we are intersecting, in detail: first you are asked for the name of the component to be fixed (the components are denoted by the letters a,b and c). Sorry, there is no built in check at all in the present version of the program, so BE SURE to input only the lower case letters a,b or c ! After you are asked for the fixed value (in %) of this component, we call this quantity 'fix' in what follows. The program checks whether this quantity is between 0 and 1 but you are not advised to chose 1 itself.

The program intersects with the plane which points (in the tetrahedron) represent states of mixtures in which the component you've chosen has fixed quantity in % . (In other words, the barycentric coordinate concerning of the component you have fixed of the points, lying on this plane, have the same fixed value.) This means that this plane intersects the edges of the tetrahedron, connected with the vertex you have choosen, in the ratio of (1-fix):fix. (So, this intersection is what I'm talking about.) This triangle will be drawn in green on the screen, its size is in linear connection with its real size, you cannot adjust it, sorry.

The program first intersects the approximating surfaces fitted on your datasets with this plane. For this purpose, you are asked next for is the refinement step drawing the picture ('delta') and Shepard's exponent ('alpha').

Delta is the length of steps the program investigates the points of the intersection (green) triangle for intersection points of the surfaces on your data set(s). This has an important consequence: the running time linearly depends on (1/delta)^2 (the square of delta's reciprocial)! May I advice you to choose delta=0.01 or so ?

Shepard's exponent (alpha) is the parameter you can vary for our built-in approximation method, which is described in detail at the 'a' (approximation) function in 5.5. Though in the functions 'a' , 'e' and 'Alt-i' (approximation/eutectics/intersection) the program allows to you to input different values for alpha, you are NOT advised to do so in the same session! Saving with the 's' function (see 5.19) only saves the last value of alpha! The best-useable interval of alpha's is suggested at the 'a' (approximation) function in 5.5, in general alpha=10 is the best.

The (bit long) computation process can be followed on the screen: a little circle shows the point the program investigates for intersecting the forthcoming surface. (The program investigates your surfaces separately, this makes the whole process a bit quicker.) When the program finds a intersecting point, this moving circle leaves a dot of the colour of the surface being investigated at present at this place. Don't worry: after the procedure the whole new picture will be re-displayed!

In general you may get curly intersections of the surfaces with the plane. This is because we do not intersect with a vertical plane.

After intersecting the approximating surfaces, the program turns to the data sets (and previously found eutectic points). Points of them, falling on the plane will be displayed by bulb size dots, while other points in the neighbourhood of the plane are shown as small color points. Since these data are not stored but computed at every time instead, all of them will be displayed at every time.

No moving/zooming/turning possibilities you have, neither the bulb function (see 5.3) has an effect. The size of the triangle is adjusted automatically, depending on its real size. Sorry, the present version does not project Eye, the + sign and the cross-wire. (I just felt like that.) ... . The present version of the program can store and display up to 300 points of each surface (I mean at most 3x300 points). The number of the points the program finds depend on delta. In case the program would find more than 300 points, the first 300 of them is stored and the program goes to the next surface (if left). Have a look at the moving point on the screen: it shows the order of investigating (and finding and storing) intersecting points!

The running time,as was indicated above, mainly depends linearly on $(1/delta)^2$ (=the square of delta's reciprocial) and on the size of the intersection (green) triangle determined by 1-fix . Furthermore, the running time also depends linearly on the size of your data sets (in each point we investigate we have to make an approximation). The number of eutectic points computed so far is not so important. Eg.choosing delta=0.005 needed 1.5 hours on a 486 PC with 150 MHz with the demo data sets, each of them containing 100 entries.

Hitting the Spacebar you will return to the main screen.

5.19 s = Save picture & computation

Using this function ALL of your data and ALL of your computation along with ALL of your picture adjustement, not to forget about the intersection picture you've computed with the function 'Alt-i' (see in 5.18) ALL will be stored in a file of the name you give.

Combining this with the function 'l' (load,see below) you will be able to continue your work (after a blackout or next midnight) as you even didn't switch your computer off!

Some further WARNINGS of mine:

The present version of the program does not check if there is a file in the present directory (or in the directory you give him). I mean that the program

simply OVERWRITES the other file with the same name! The present version of the program also does not shows directories, you are supposed to keep them in mind!

There is no 'saved' flag in the program, so be sure to save your work after you have computed some complicated, and before ending the program!

In general, this function generates a large file (usually 200-400 kb!), so make sure to have enough space before using the 's' function ! The size of this file (among others) depens on the refinement of your computations you've done with functions 'e' (eutectic, see 5.6) and 'Alt-i' (intersection, see 5.18). I mean, on delta.

5.20 l = Load picture & computation

Using this function ALL of your/mine data and computation along with your / mine picture adjustement, not to forget about the intersection picture you/me have computed with the function 'Alt-i' (see in 5.18) ALL will be LOADED from a file in which you have already previously saved them with the function 's' (see above).

WARNING: the present version of the program does not warn you to save your work before loading! Since everything in computer's memory will be overwritten while loading, ALL OF YOUR WORK in the present session WILL BE LOST!. So be sure to save them with the function 's' described above before loading another one!

The program offers you to load the demo file SGOMB3-.3DI as default, in this case you should hit only Enter and to see what happens.

If you want to read another file, you must give full filename with the extension, moreover, you should give it with full pathname in case your file is in other directory.

In case you give a nonexistent filename, after a red color warning you are asked 'Do you try another file(y/n)?'. Choosing 'y' (and hitting Enter) you get the next chance while choosing 'n' (and Enter) the program returns to the 3-dim.main screen, displaying your old data sets and computations (if you had any).

After succesful loading the program returns to the 3-dimensional main picture displaying the new data sets (and the interpolating surfaces and eutectic points as well, if there are any). One 2-dimensional intersection-picture (if there was one constructed by the function 'Alt-i', see 5.18) was also saved, which can be redisplayed by using 'Alt-i' again and chosing the option 'yes, display the previous picture'.

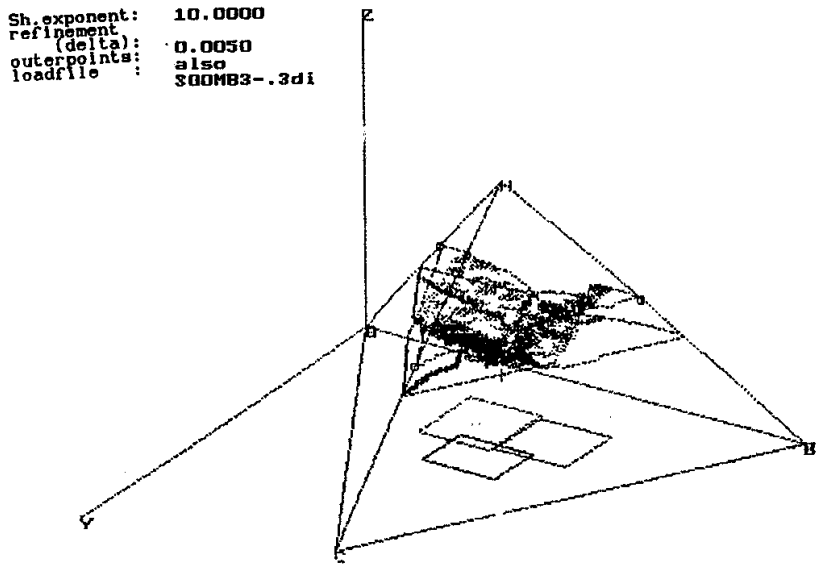
Combining this function with the function 's' (save,see above) you will be able to continue your work (after a blackout or next midnight) as you didn't even switch your computer off!

A puzzle: what is the escape method when accidentally 'l' (load) was pressed?

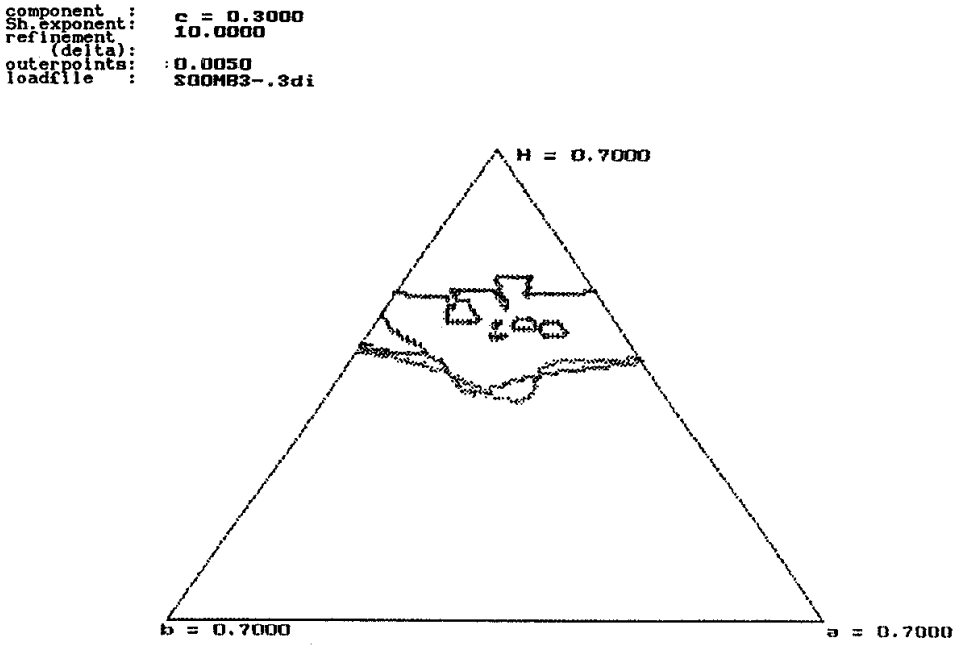
5.21 Ctr-End = End program

Before this function make sure you have saved all your computation with function 's' (save,see in 5.19) for further re-use or computations!

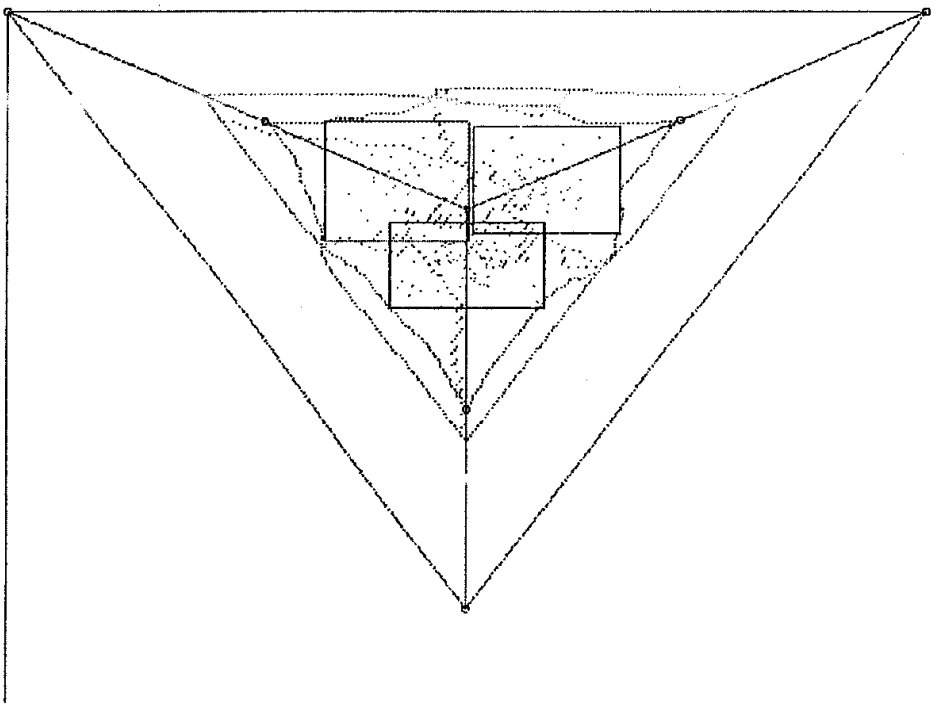
Have a nice time!



The main screen



Intersecting with c=0.3



Hit SPACEBAR to return

Projecting to the [xy] plane