Lesson 22: Using the SYNOPSYS Glass Model

When you vary the properties of an optical glass in SYNOPSYS, you are asking the program to find values of the index Nd and Abbe number Vd that will correct aberrations and that are within the boundaries of the commercial glass maps. That's pretty straightforward – but you need more than just those two parameters. The program also must calculate the index of refraction at each wavelength in the lens, and you hope that the value so found will vary with wavelength in a manner that closely resembles the behavior of real glasses. That is the purpose of the glass model.

A model glass is inserted into the lens via the SpreadSheet, or more quickly, with the keyboard or WorkSheet entry. For example, the input

```
CHG
1 GLM 1.6 55
END
```

assigns a model glass to surface 1 with the Nd and Vd values specified. You can declare the glass variables in the PANT file with input such as

```
PANT
VY 1 GLM
VY 3 GBC
VY 5 GBF
VLIST GLM 1 5 8
VLIST GLM ALL
…
END
```

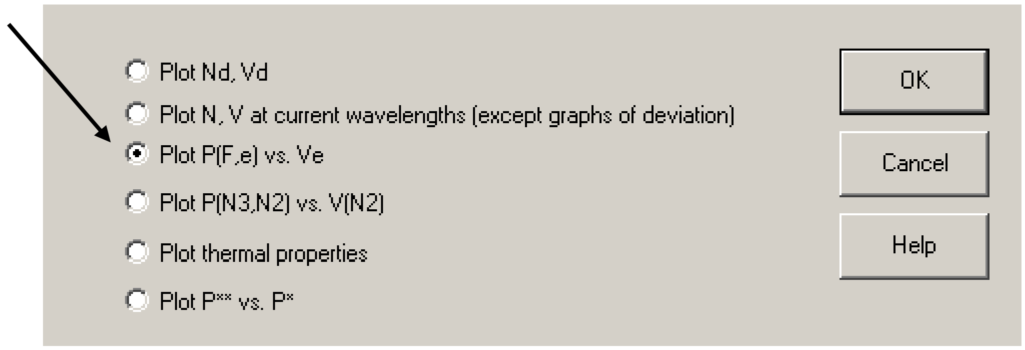
The **VLIST** form varies all glasses that are already declared model glasses, while the form **VN <u>sn</u> GLM** forces the material to be a glass model, if it is not already. In that case, the program first finds the model that most closely resembles the current glass, and then starts with that model. **GBC** and **GBF** are used to vary a glass along the crown or flint boundary.

Glass boundaries are tricky to implement. During optimization, the index often wants to become very high, and of course the dispersion of many elements would like to be infinite. That would be great mathematically, but such materials do not exist -- so the program has to constrain the glass model to the useable part of the glass map. To do so, it does something clever: when one of the glasses tries to go over the boundary on the left or right, the program first restricts the change so that the glass goes exactly to that boundary; then it redefines that variable, changing the GLM variable to a **GBC** (glass bounded, crown) or **GBF** (glass bounded, flint) variable instead. Then the glass variable will move up or down along either the crown or flint boundary. As a result, the glass remains within the glass map, and one is left with a single variable where previously there were two. If the glass tries to go over the upper or lower limit on the index, the program again reduces the change so it goes exactly to that boundary. In this way, the glass model variable always remains inside the glass map boundaries.

Once a glass has become pinned to the crown or flint boundary, it remains there for the duration of that run. It sometimes happens, however, that after a design has been much improved, one of the glasses would be better if it left the boundary. This is simple to test: just run the optimization again. The glasses will start out free to move anywhere, and they can immediately leave the boundary if it improves the lens.

Of course you do not expect the model glass after optimization to coincide exactly with any real glass in a selected vendor's catalog, but that is not a problem since you usually can find one whose properties resemble the model closely enough. Then you just substitute that glass and reoptimize. But many high-quality designs have to compensate for secondary color to some degree, and in order for the program to optimize glasses while accounting for that aberration, the partial dispersion of the model should be reasonably close to that of nearby real glasses.

But now it gets tricky. SYNOPSYS uses a polynomial expression that yields the index at any wavelength in the visible region, given only the glass-map coordinates (Nd, Vd), the coefficients having been found via a least-squares fit to the entire Schott glass table. The figure below shows the Schott glass map, where the Graph option has been selected to show the partials P(F,e) vs. Ve. (Open the glass map with MGT or the PAD button BK7, select Schott, click Graph, and select that option.)



Plot Nd, Vd

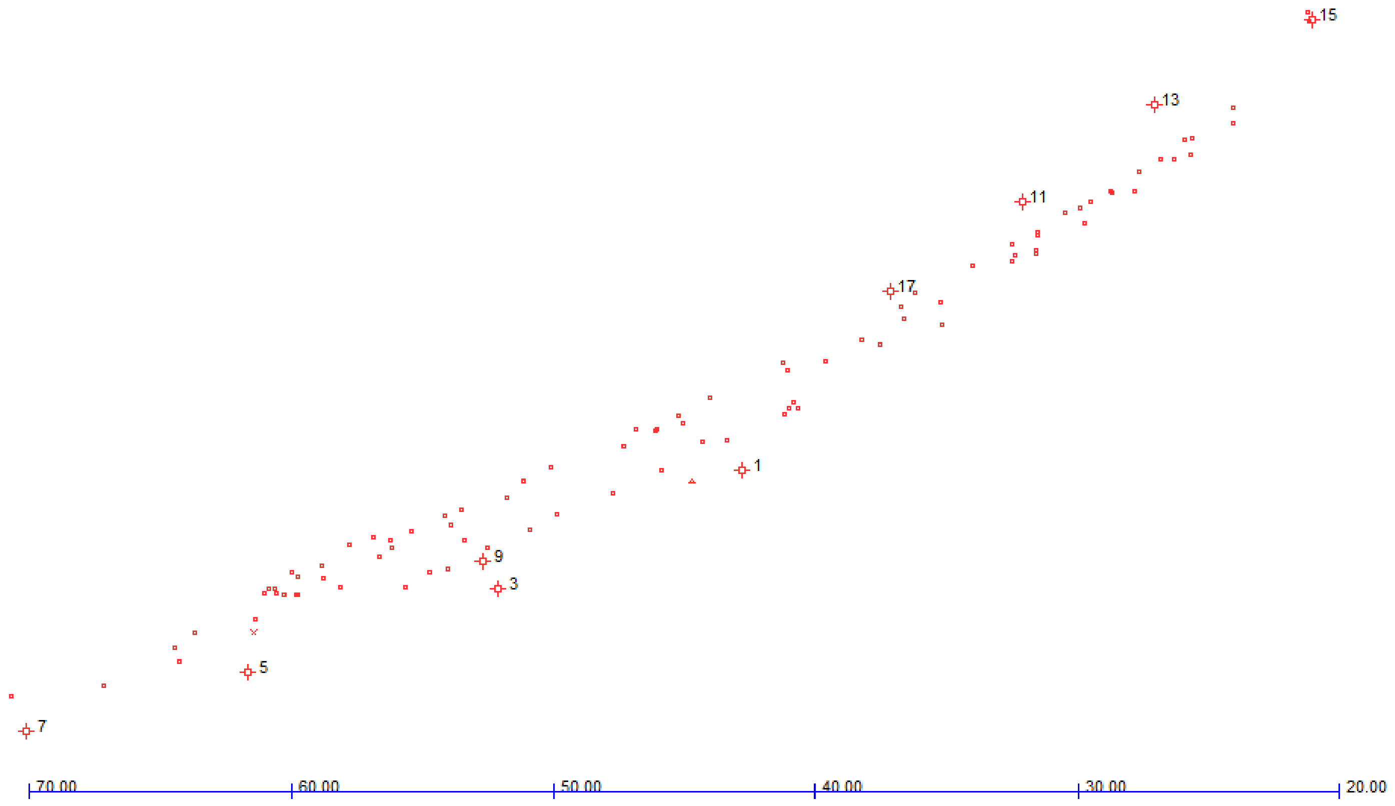
Plot N, V at current wavelengths (except graphs of deviation)

[✓ Plot P(F,e) vs. Ve]

Plot P(N3,N2) vs. V(N2)

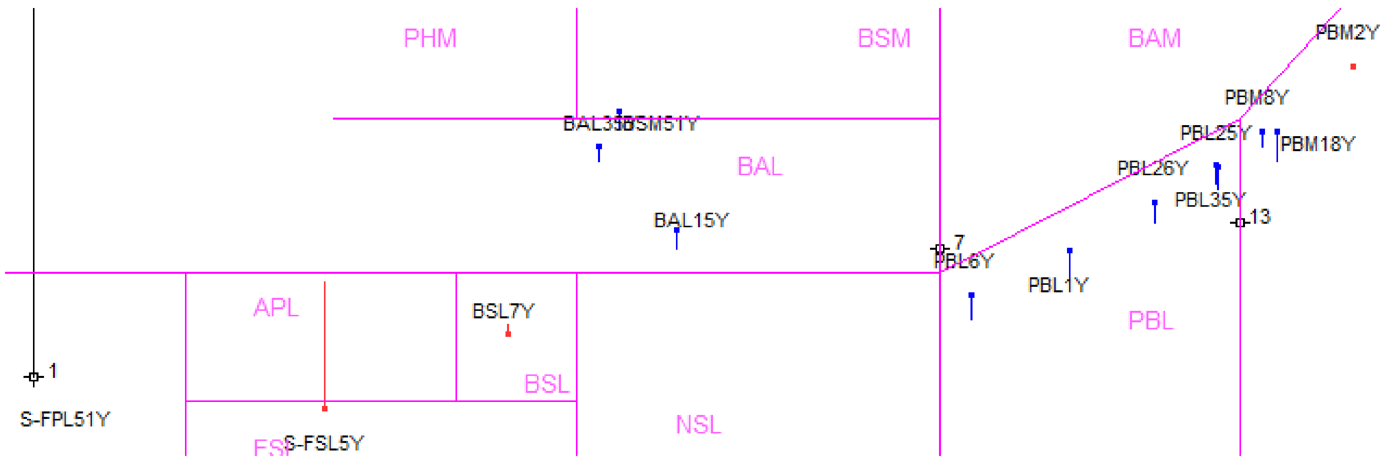
Plot thermal properties

Plot P** vs. P*



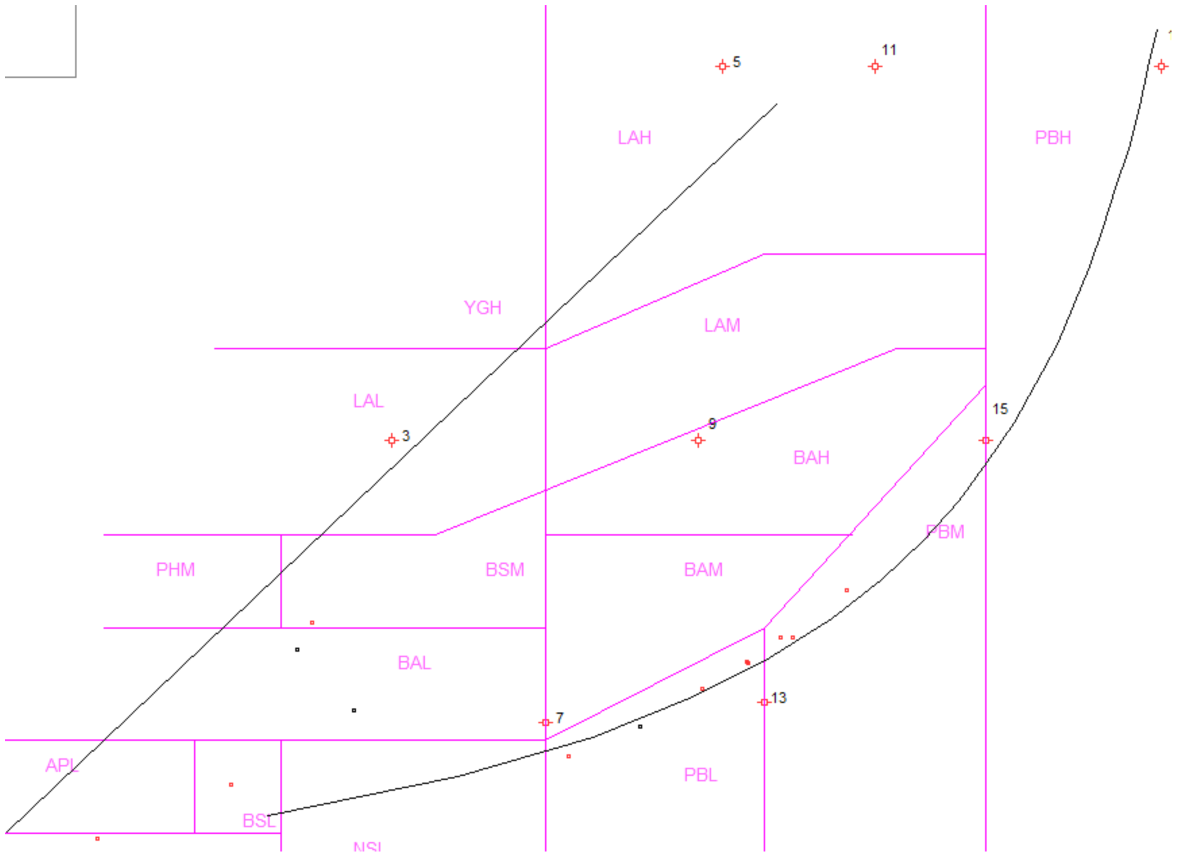
For this example we have prepared a lens of 8 elements with glass models assigned as shown by the red circles above. The goal is for the model to approximate the same distribution as the real glasses, which is does well enough to be useful.

Now we will show how to adapt the glass model for special conditions. A good example is a lens designed for the UV spectrum, where we were restricted to iLine glasses from the Ohara glass company. How can one vary the glass model while staying in the region where those glasses can be found? Simple. Here is the glass map for that lens. (To restrict the display to just the iLine glasses, we selected that radio button.)

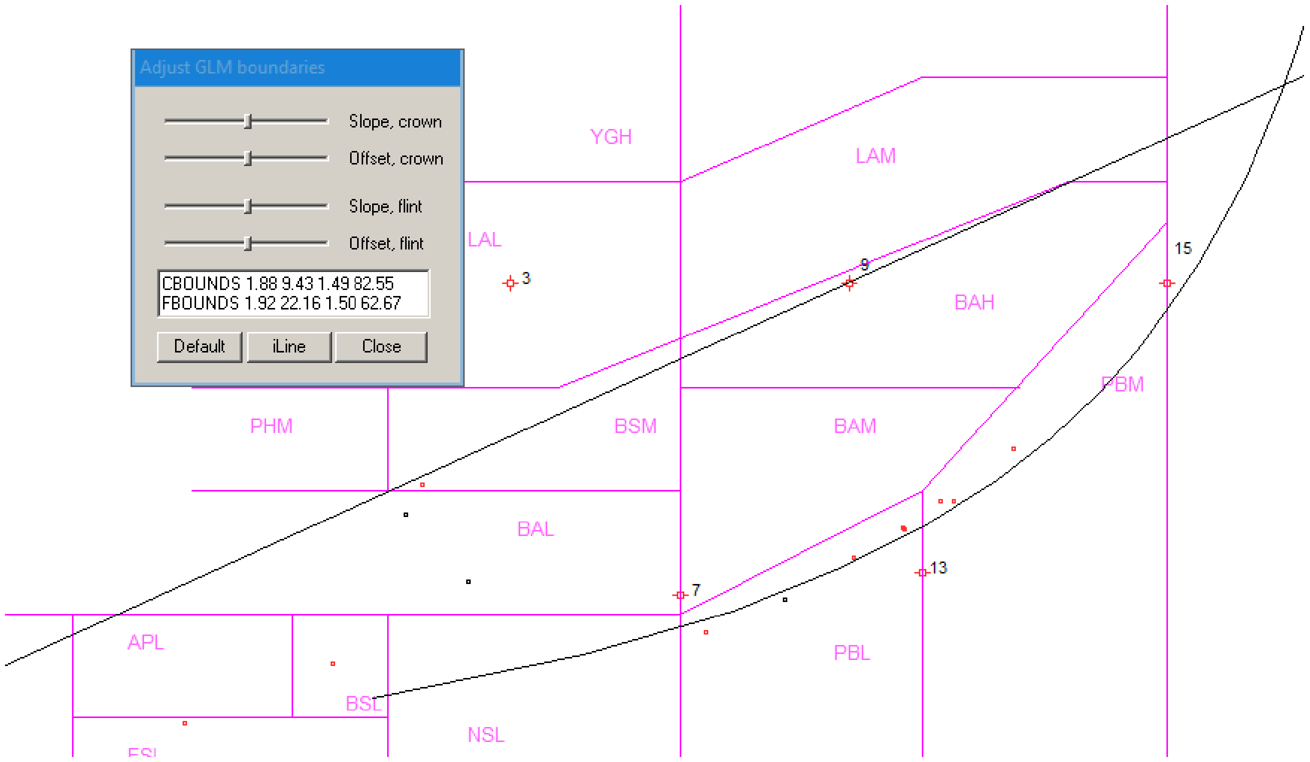


If we just varied the GLM variables as usual, we would probably wind up with very high-index materials, which are not very close to one of the iLine glasses. We can prevent that by changing the boundaries. Click the button

, and the program displays the current (in this case the default) boundaries.



Now, click the iLine button on the boundaries dialog. You see the region where iLine glasses are to be found. You can also adjust the boundary lines with the sliders in this dialog.



```
CBOUNDS 1.88 9.43 1.49 82.55
FBOUNDS 1.92 22.16 1.50 62.67
```

There are four parameters that can be specified in the PANT file for controlling the glass boundaries, and the edit box shown above gives data for the CBOUNDS and FBOUNDS directives. Select those lines and then copy-paste them into the PANT file, near the top. Then add another line, giving an upper limit to the GLM index variables of 1.6, with the CUL (crown, upper limit) line. The PANT file now starts with

```
PANT
CBOUNDS 1.88 8.43 1.49 82.55
FBOUNDS 1.92 22.16 1.50 62.67
CUL 1.6
…
```

Now, when the glasses are varied, they will remain in the area shown above, and we would expect no trouble finding an iLine glass to match the model.

One final note: When you give the program a glass model, you are specifying the *input* to the polynomial. The actual index that comes back at each wavelength is the *output* from the model, and the two are often slightly different. If the lens is assigned the CDF spectral lines, they will be very close – but if your spectrum is anything else, then you can expect the index listing from SPEC (which gives the model *input*) to be different from the output of PRT (which lists the *output* indices).

We have found this glass model to be invaluable for finding where on the glass map the lenses want to be. In some cases, the program has even managed to correct secondary color by a good choice of glass, all by itself.