Conformal Maps

The purpose of this category is to help visualize the mapping properties of a complex analytic function. The basic method for using this category is to first, as usual, choose a canned object from the Conformal Map menu, or else choose User Defined... from the menu and define your own mapping. As soon as a mapping has been selected, the program draws a coordinate grid in the graphics window (the "preimage grid"). This is a "Cartesian" grid in the rectangle of the complex z-plane defined by umin $\leq \Re(z) \leq$ umax and vmin $\leq \Im(z) \leq$ vmax, so the total resolution of this grid is uResolution $\times$ vResolution. (As usual, you can modify the values of umin, umax vmin, vmax, uResolution and vResolution using the Settings Menu.)

When the preimage grid has been drawn, it remains visible for a few seconds and then is replaced by its image under the chosen mapping (the "image grid"). But the preimage grid has not been erased! You can see it at any time by holding down the option and command keys and pressing the mouse at any point in graphics window. The preimage grid will appear

and stay in view until you release the mouse button, at which point the image grid will reappear. You can keep switching back and forth between the two images by pressing and releasing the mouse button while holding down the option and command keys.

The idea, of course, is to develop a feeling about what gets mapped where by the chosen mapping. If the map is at all complicated, it can be difficult to match up a line in the pre-image with its corresponding curve in the image so, to help with this matching, each gridline is drawn in a different color—with the same color used of course for both the preimage and image. This is called "color coding the gridlines'. Since color coding is seen better against a black background, black is the default background in this category, but it is possible to choose a white background using the View menu.)

You can switch the preimage domain from a rectangle in cartesian coordinates to a wedge in polar coordinates by choosing Polar Grid from the Conformal Map menu. Now the preimage curves will be parts of rays thru the origin and parts of circles centered at the origin.

You can add your own lines and circles to the preimage view (and their images to the image view) by selecting either Choose Line by Mouse or Choose Circle by Mouse from the Conformal Map menu. After selecting Choose Line by Mouse, the preimage view will appear, and you should press the mouse to select one point of your line, and then drag to a different point to select the second point. Immediately after you release the mouse at the second point, the line will be drawn in red. After a second the image view will return, and the image curve of the line you chose will be drawn , also in red. For a circle the procedure is the same, with the first mouse point establishing the center and the second some point on the circumference. Circles and their images are drawn in green.

If you click and drag then a conformal map image will follow the mouse around. If you now depress the Shift key and move the cursor up or down then the conformal map image gets smaller or larger. Morever, if you hold down Command and then drag out a rectangle in the usual Mac way, then when you release the mouse (with Command still down) your selection rectangle will zoom to the entire window.

The ATOs in this category are highly cross-referenced and taken together provide an abbreviated intoduction to conformal mapping. Start by selecting $z \mapsto z^2$ from the Conformal Map menu, then select About This Object from the Documentation menu (or click the ATO button in the small Special ATO! window), and then start experimenting with the various suggestions you will find there and in th ATOs of other conformal maps that are nentioned there.

Many of the pre-programmed examples in this category have carefully tailored default morphs. The polynomial $z \mapsto z^2$ is deformed to the identity by varying the exponent from 2 to 1. In a number of other cases, morphing also defaults to a deformation between the identity map and the selected map. For example The fractional linear map $z \mapsto (z-1)/(z+1)$ is deformed through fractional linear maps to the identity and thereby shows that polar coordinates look the same at 0 and ∞. The exponential map is actually implemented as $\exp(aaaz)$, where aaa is a complex parameter, say $aaa = a + ib$. Note that this amounts to precomposing $z \mapsto \exp(z)$ with the map that stretches z by a factor $r = \sqrt{a^2 + b^2}$ and rotates it by an angle $\theta = \arctan(b/a)$. In the default

morph, a is 1 and b varies from 0 to 0.4, so the standard parameter lines—circles and straight lines—are gradually deformed into spirals.

The User Defined... dialog is a little different for this category than for the categories dealing with real-valued objects. At the top there is a box, pointed to on the left by "$z - - - >$". In this box you should fill in a formula for the function of the complex variable z that you wish to study. This formula—which defaults to that for a general polynomial of degree five, $aa + bb * z + cc * z^2 + dd * z^3 + ee * z^4 + ff * z^5$—can involve not only z, but also nine complex parameters, $aa, bb, cc, \ldots, ii$, which for convenience can be set in this same dialog. (They may also be set using the Set Parameters... item of the Settings menu.) What you are really setting in this dialog are expressions for these parameters, not the parameters themselves, which are evaluated from the expressions when you dismiss the dialog. This means that you can for example use the expression $\pi + e * i$ in the box for aa to give it the value $3.14159+2.71828i$. (You get the symbol π by typing Option-p—but you can also write pi if you prefer.) (Note that this explains why we have chosen to use $aa, bb, cc, \ldots$ instead of simply $a, b, c, \ldots$

for the names of the nine parameters—it was to avoid conflicting with $e = 2.71828\ldots$ and $i = \sqrt{-1}$.) The expressions for the parameters are evaluated in alphabetic order, which means you will get what you expect if you set aa to 1, bb to 2 and cc to $aa+bb$ (but setting aa to $bb + cc$ and bb to 1 and cc to 2 won't work.)

Built-in complex functions available for creating the expression for the conformal mapping function are: sin, cos, tan, cot, csc, sec, sinh, cosh, tanh, coth arcsin, arccos, arctan, arccot, ln, exp, sqrt,cubert (the cube root), sn, cn, dn, Re, Im, Arg, abs, conj, round, and trunc. Some of these need a little explaining. Many of them are multiple-valued, and we have tried to make the "standard" branch-cuts. (In particular, the arg and ln function have branch cuts along the negative real axis). The seemingly real-valued functions Re, Im, Arg, abs actually are complex with zero imaginary part. The function abs is the modulus function and conj is complex conjugation. Round and trunc apply themselves to both the real and imaginary parts of their arguments. The functions sn, cn, and dn are the usual Jacobi elliptic functions. They depend on a parameter, the modulus, usually denoted

by m. The value of m can be set in the same dialog as the parameters $aa, bb, cc, \ldots, ii$.

Note that Karcher's more symmetrically normalized elliptic functions are included in the pre-programmed examples. For these, the modulus is not restricted to be real, which is to say that they are defined on tori, of all conformal types, not just the rectangular tori. The standard morphs of these elliptic functions emphasize how the sphere is covered by (spherically) congruent images of pieces that are 1/8th of the torus.

The complex expression parser and evaluator, like the real one, was written by David Eck. In fact, at my request , David modified his real expression unit to handle complex expressions, and I cannot thank him enough for doing me this favor at a time when he was also very busy with his own work.