Visualization of Complex Functions

Bernd Thaller, University of Graz, Austria

Source:

This article appeared first in
The Mathematica Journal 7(2), 163-180 (1998),
The package ComplexPlot.m described in this article is now part of the Visual-Quantum-Mechanics (VQM) packages. The VQM-packages are a collection of *Mathematica* packages, distributed with the book "Advanced Visual Quantum Mechanics", Springer-Verlag 2004. They can also be downloaded from

http://vqm.uni-graz.at/pages/dl_vqm_packages.html

The article has been updated to reflect the changes in the package since 1998.

Date: 2004-08-08

Introduction

Functions $f(x, y)$ of two variables with values in the field of complex numbers, such as analytic functions, are often considered abstract mathematical objects which are difficult to imagine. Indeed, the graph of such a function would have to be drawn in a four-dimensional space (with coordinates x, y, $\mathrm{Re}(f)$, $\mathrm{Im}(f)$), which cannot easily be done on a sheet of paper. Therefore, several aids for visualizing complex functions have been developed. For example, the standard package `ComplexMap.m` by Roman Maeder illustrates how the function transforms and distorts the complex plane. Another method uses colors for the visualization of complex values. The standard package `ArgColors.m` specifies colors to describe the argument of complex numbers. This color map is also used in Kevin McIsaac's package `ComplexPlot3D.m` [McIsaac].

Here we propose a refined color map of the complex plane. The color map is continuous and one-to-one from the complex plane onto the surface of the color manifold. In addition to describing the argument of a complex number by the hue of the color, the color map uses the lightness of the color to represent the absolute value. Thus the color map is suitable for colored density plots of complex functions. The package `ComplexPlot.m` provides commands to produce intuitive pictures of complex-valued functions, including density plots and surface plots. Since the human eye usually cannot recognize the precise color values, our method is not suited for a quantitative representation, but it can be used to give a quick impression of the most important qualitative properties of the complex function, such as the position and order of poles and zeros. The method is particularly useful for the visualization of quantum mechanical wave functions [Pauschenwein and Thaller 1996].

The Color Map

The complex numbers form a two-dimensional manifold which is usually described as a plane with coordinates $x = \mathrm{Re}(z)$ and $y = \mathrm{Im}(z)$. Sometimes polar coordinates $r = |z|$ and $\phi = \mathrm{Arg}(z)$ are also used to represent points in the complex plane. For our purpose, the representation of complex numbers as points (θ, ϕ) on a sphere is more suitable. This representation is given by the method of stereographic projection. Consider a sphere of radius r_0 centered at the origin of the three dimensional space. Let us interpret the x-y plane as the complex plane. From a point $(x, y, 0)$ corresponding to $z = x + i\,y = r\,e^{i\,\phi}$, we draw a straight line to the north pole $(0, 0, r_0)$ of the sphere. The intersection of the line with the surface of the sphere maps each point of the complex plane in a one-to-one way to a point on the surface. Using spherical coordinates (θ, ϕ) on the sphere, we find that the azimuthal angle is just the phase $\phi = \mathrm{Arg}(z)$, while the polar angle θ depends only on $r = |z|$ and on r_0:

```
θ[r_, r0_] := π - 2 ArcTan[r / r0]
```

The origin $z = 0$ of the complex plane is mapped to the south pole $\theta = \pi$. The north pole $\theta = 0$ is the image of a complex "number" which has an undefined phase (like the origin) and has an infinite absolute value. *Mathematica* uses the symbol **ComplexInfinity** (which is the same as **DirectedInfinity[]**) to represent this object. The complex numbers with $|z| = r_0$ are represented by the equator of the sphere. All complex numbers with absolute values bigger than r_0 are mapped to the northern hemisphere, and the points with $|z| \le r_0$ are mapped onto the southern hemisphere.

The manifold of colors that can be represented in a computer is three dimensional. In the most frequently used RGB system, the colors are described by coordinates (R, G, B) that give the intensities of the "primary" colors red, green, and blue. The graphics directive **RGBColor[***R*,*G*,*B***]** accepts arguments in the range $[0, 1]$. Thus, in *Mathematica* the color manifold is represented by the three-dimensional unit cube. The corners of the cube represent the additive primary colors red $(1, 0, 0)$, green $(0, 1, 0)$, blue $(0, 0, 1)$, and the subtractive primary colors yellow $(1, 1, 0)$, cyan $(1, 0, 1)$, magenta $(0, 1, 1)$. The remaining corners are black $(0, 0, 0)$ and white $(1, 1, 1)$. All shades of gray are on the main diagonal from black to white.

A color code for complex numbers can be obtained by defining a one-to-one mapping from the surface of the sphere into the manifold of colors. To do this in a natural way, it is useful to have a short look at some less familiar coordinate systems for colors. (For more information on color systems, see [Smith and Blachman 1995].)

The distance of a color $C = (R, G, B)$ from the origin measured in the maximum metric is called the *brightness*, $b(C) = \max\{R, G, B\}$. The distance from a gray point with the same brightness is called *saturation*, $s(C) = \max\{R, G, B\} - \min\{R, G, B\}$. Note that the possible values of the brightness b are between 0 and 1, and for each value of b the saturation varies between 0 (gray) and $s_{\max}(b) = b$. The set of all colors with constant saturation s and brightness b is hence a closed polygonal curve $\Gamma_{s,b}$ with length $6s$ (which is formed by six edges of a cube, each of length s).

The *hue* $h(C)$ of a color C is $\lambda / 6\,s$, where λ is the length of the part of $\Gamma_{s,b}$ between C and the red corner of $\Gamma_{s,b}$ (the corner with maximal red component) in the positive direction (counter-clockwise, if viewed from the white corner). That means $h = 0$ and $h = 1$ both correspond to the red corner and it is most natural to define the hue as a cyclic variable modulo 1. Hence the colors at the corners of $\Gamma_{s,b}$ (which are red, yellow, green, cyan, blue, and magenta with saturation s and brightness b) have the hue values $0, 1/6, 1/3, 1/2, 2/3$, and $5/6$ (mod 1). The built-in color directive **Hue[***h*,s_b,*b***]** defines a color in terms of its hue h, the (scaled) saturation $s_b = s / b$, and brightness b (the HSB color system). The saturation s_b is scaled so that its values at a given brightness b range between 0 and 1.

It is sometimes considered a drawback of the HSB system that the "pure" colors red, yellow, green, and so on (the corners of the RGB cube) by definition have the same brightness as white ($b = 1$). The definition of the *lightness* is designed to give a quantity that describes the human perception more accurately.

For any color $C = (R, G, B)$, the lightness $l(C)$ is defined as the average of the maximal and the minimal component, $l(C) = (\max\{R, G, B\} + \min\{R, G, B\})/2 = b(C) - s(C)/2$. We have $0 \le l \le 1$. At a given lightness l, the brightness b ranges in $l \le b \le \min\{1, 2l\}$. Hence lightness $l = 0$ denotes black, and $l = 1$ (that is, $b = 1$, $s = 0$) is white. The maximal saturation $s_{\max}$ at a given lightness l is l for $l \le 1/2$ and $2 - 2l$ for $l \ge 1/2$. The set of color points that have maximal saturation with respect to their lightness is just the surface of the RGB color cube. On the surface, the brightness is also maximal, so that $b = 2l$ for $l \le 1/2$ and $b = 1$ for $l \ge 1/2$. Thus the pure colors have lightness $1/2$ and maximal saturation. The standard package `Color.m` defines the color directive **`HLSColor[`***h*,*l*,s_l**`]`** which describes a color in terms of hue *h*, lightness *l*, and saturation $s_l = s/s_{\max}(l)$, where this time the saturation is scaled so that its values at a given *lightness* range between 0 and 1. Hence the colors at the surface of the color manifold are described by **`HLSColor[`***h*,*l*,1**`]`** or equivalently by **`Hue[`***h*,1,2*l***`]`** for $l \le 1/2$ and **`Hue[`***h*,2-2*l*,1**`]`** for $l \ge 1/2$.

The colors on the surface of the color manifold have maximal saturation (in the HLS system) and can therefore be distinguished most easily. It is therefore most natural to color the sphere (and hence the complex plane) by defining a continuous mapping from the points (θ, ϕ) of the sphere onto the surface of the color manifold. This can be done most easily in the HLS color system.

Our color map associates the point (θ, ϕ) on the sphere to the color given by **`HLSColor[`**$\phi/(2\pi)$, $1 - \theta/\pi$, 1**`]`**. We have chosen a linear dependence of the lightness on the polar angle θ such that the south pole ($z = 0$, $\theta = \pi$) has lightness 0 (black), and the north pole (**`ComplexInfinity`**, $\theta = 0$) appears with lightness 1 (white). The colors on the sphere have maximal saturation 1 in the HLS system and thus have maximal contrast. The cyclic variable hue is most naturally associated to the azimuthal angle ϕ, which corresponds to the argument of the complex number z. The pure colors red, yellow, green, and so on, are on the equator of the sphere. They correspond to complex numbers with $|z| = r_0$. The points more to the north appear lighter, the points on the southern hemisphere are darker. Figure 1 shows a visualization of the color map of the sphere. The color map of the complex plane is shown in Figure 2.

```
<< Graphics`Colors`

coloredCircle[θ_, steps_] := Module[{pts, lns, lst, s = Sin[θ]},
  pts = Table[{s Cos[phi], s Sin[phi], Cos[θ]}, {phi, 0., 2. π, 2. π/steps}];
  lns = Line /@ Partition[pts, 2, 1];
  lst = Table[HLSColor[h, 1 - θ/π, 1], {h, 0., 1., 1/(steps - 1)}];
        Transpose[{lst, lns}]]

coloredSphere[steps_, circles_] := Table[coloredCircle[θ, steps], {θ, 0., π, N[π]/circles}]

graph = Graphics3D[{Thickness[0.01], coloredSphere[30, 30]}];
```

```
Show[graph, Background → GrayLevel[0.7], Axes → True,
  Ticks → None, AxesLabel → {x, y, z}, ViewPoint → {1.784, -2.628, 1.178}];
```

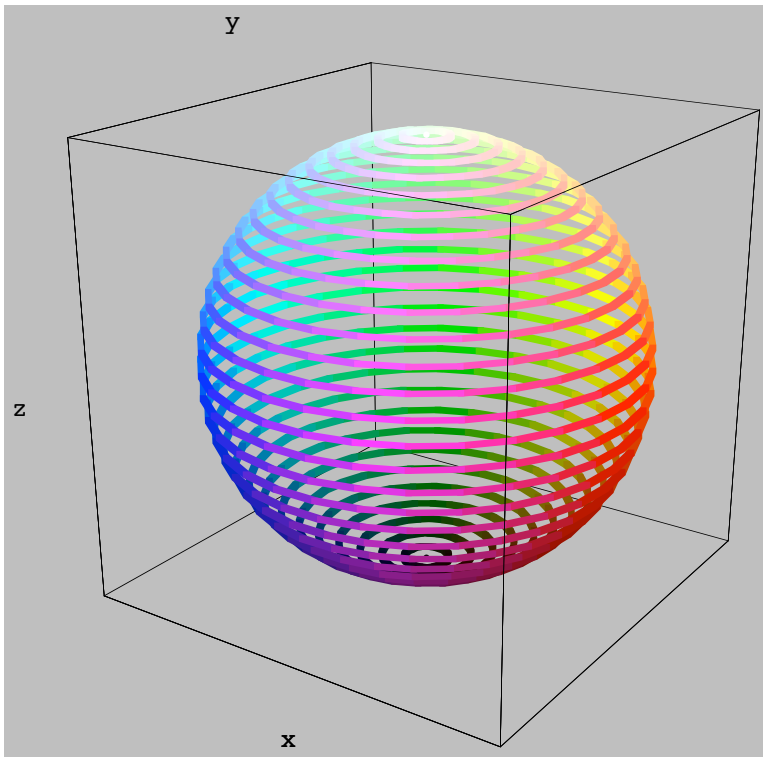


Figure 1. The color map of the complex sphere.

Finally, our color map for complex numbers (given in polar coordinates $r = |z|$ and $\phi = \mathrm{Arg}(z)$) can be defined as

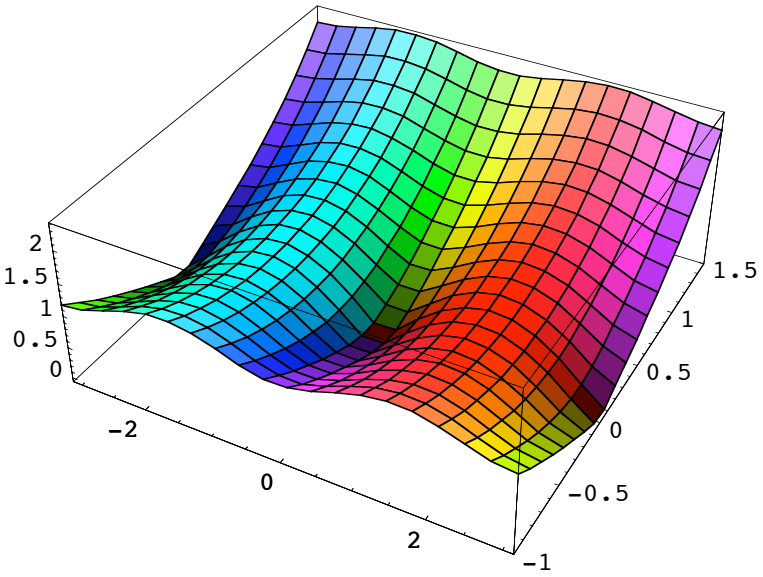
```
light[θ_] := N[1 - θ / π]

ComplexToColorMap[r_, ϕ_, r0_: 1.] :=
  Module[{li = light[θ[r, r0]]},
    Hue[Mod[ϕ / 2 / π, 1],
              If[li <= 0.5, 1, 2 - 2 li],
              If[li <= 0.5, 2 li, 1]] ]
```

Surface Plots and Density Plots

The color map defined in the previous section makes it easy to produce surface plots of complex-valued functions.

```
ComplexPlot3D[func_, {x_Symbol, x1_, x2_}, {y_Symbol, y1_, y2_}, optns___Rule] := Plot3D[
  {Abs[func], ComplexToColorMap[Abs[func], Arg[func]]}, {x, x1, x2}, {y, y1, y2}, optns]
```

```
ComplexPlot3D[Sin[x + I y], {x, -π, π}, {y, -1, 1.5}, PlotPoints → 20];
```



Here the height of the surface shows the absolute value $|z|$ of a complex number z, while the color is defined according to **ComplexToColorMap**.

Finer details or rapid oscillations of complex functions would be better represented in a sort of density plot. Unfortunately, the command **DensityPlot** can only use color functions that depend on a single real variable (the value of the real function to be plotted). Therefore, **ComplexToColorMap** cannot be used with the option **ColorFunction**. However, we can use the graphics primitive **RasterArray** to define a two-dimensional array of rectangles that are colored according to our convention. The following command plots a "density graphic" of an array of complex numbers:

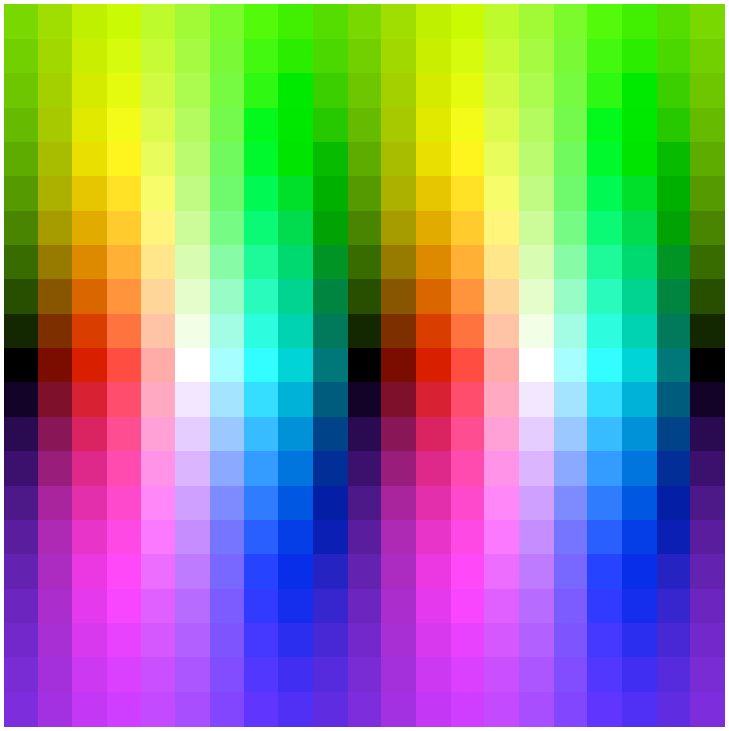
```
ListComplexDensityPlot[valuearray_, optns___] :=
 Module[{colors},
  colors =
   MapThread[ComplexToColorMap[#1, #2] &,
       {Abs[valuearray], Arg[valuearray]}, 2];
  Show[Graphics[RasterArray[colors], optns]]
]
```

For this array of complex values,

```
A = Table[Tan[x + I y], {y, -1, 1, .1}, {x, -N[ Pi], Pi, N[Pi / 10]}];
```

the plot clearly shows the zeros (black) and poles (white) of the complex function tan(z).

```
ListComplexDensityPlot[A, AspectRatio -> 1];
```



The package `ComplexPlot.m` is based on the ideas and definitions described above.

The Package ComplexPlot.m

The main goal in designing the package has been to define a command **`ComplexDensityPlot`** that behaves like the built-in **`DensityPlot`**. Moreover, the package contains commands for colored surface plots and contour plots of complex functions. If the package and its documentation are placed in a subdirectory `VQM` of the `AddOns/Applications` directory, it can be loaded with the command

```
<< VQM`ComplexPlot`
```

(We don't have to restart the kernel, because the package prevents shadowing [Wagner 1996]. As a replacement for the previously defined commands **`ComplexPlot3D`** and **`ListComplexDensityPlot`** the package defines the commands **`QComplexPlot3D`** and **`QListComplexDensityPlot`**. Note that all symbols defined by the VQM packages start with the letter Q.)

To make the documentation accessible through the Help Browser, create the directory `AddOns/Applications/Graphics/Documentation/English` and put the files `BrowserCategories.m`, `BrowserIndex.nb`, `ComplexPlot.nb`, and `CPcmds.nb` in this directory. After executing the command Rebuild Help Index in the *Mathematica* front end, the new help topics should be available.

The command **QComplexDensityPlot[***f***[***x***,***y***],{***x***,***xmin***,***xmax***},{***y***,***ymin***,***ymax***}]** implements an analogue of **DensityPlot** for complex-valued functions. It generates a two-dimensional raster of colors which are determined from the complex values of *f* according to the color map discussed in the previous section. For example, the following command gives a colored density plot of the identity function, which illustrates our color map of the complex plane.

```
QComplexDensityPlot[x + I y, {x, -3, 3}, {y, -3, 3}];
```

```
Arg::indet : Indeterminate expression Arg[0. + 0. i] encountered. More…
```

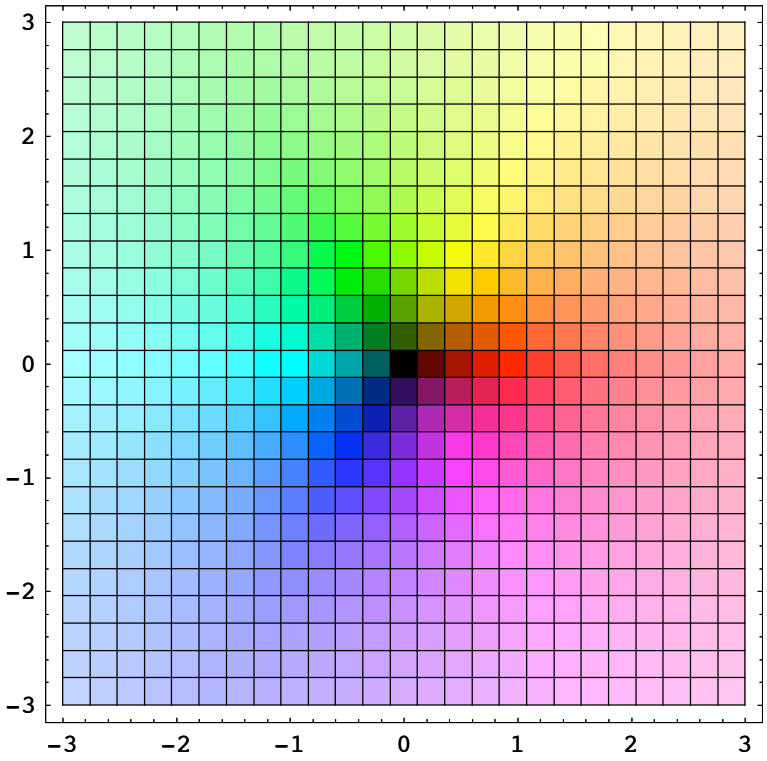


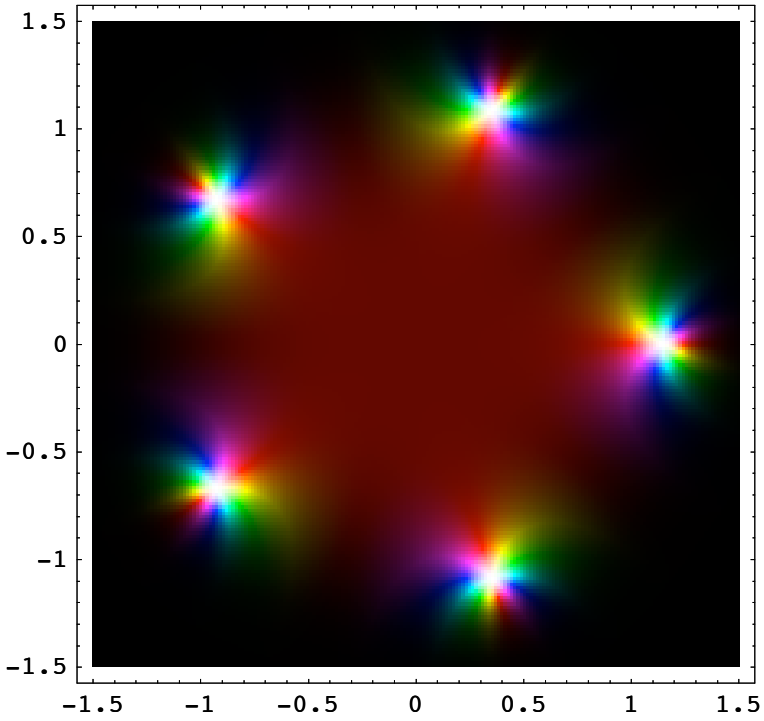
Figure 2. The color map of the complex plane.

Note that red is positive real. The complex numbers on the unit circle have maximal brightness and saturation (lightness $1/2$). For z on the unit circle, $-z$ has the complementary color. A complex zero appears black. The warning is printed because the argument (and hence the hue) of zero is not defined. Other exceptional values producing error messages are **ComplexInfinity** and **Indeterminate**. You can turn these messages off using the command

```
Off[Arg::indet]
```

Density plots are well suited for displaying fine details of functions at high resolutions. We can increase the resolution using the option **PlotPoints**. For plots at high resolution, we should turn off the mesh lines. The following command takes a long time on slow computers.

```
QComplexDensityPlot[1/((x + I y)^5 - 2)^2,
 {x, -1.5, 1.5}, {y, -1.5, 1.5}, PlotPoints → 200, Mesh → False]
```

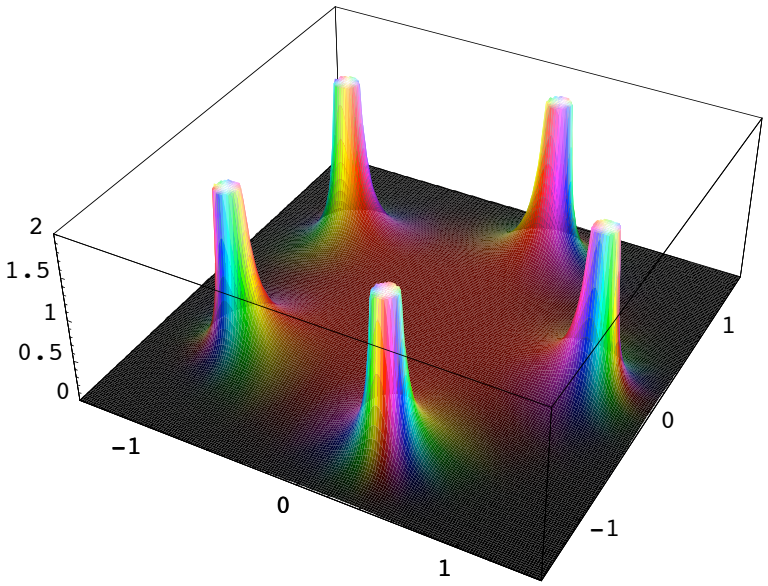



```
-QColorDensityGraphics-
```

The bright points indicate the five poles of the function $f(z) = (z^5 - 2)^{-2}$. It can be seen that the poles are of second order, because all colors appear twice on a small circle around the pole.

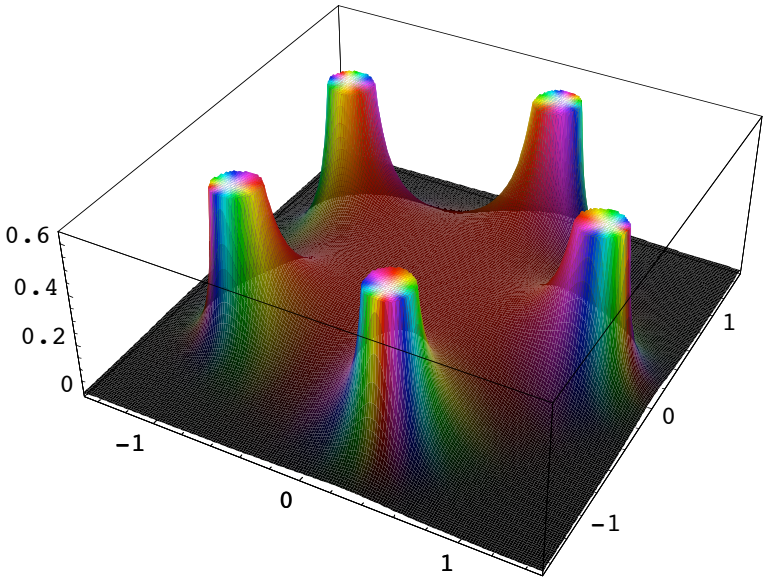
The command **QComplexDensityPlot** returns a graphics object called **QColorDensityGraphics**, which can be transformed into other graphics objects, such as **ContourGraphics**, **DensityGraphics**, or **SurfaceGraphics**.

```
Show[SurfaceGraphics[%], Axes -> True, AspectRatio -> Automatic, PlotRange -> {0, 2}];
```



The package provides the command **QComplexPlot3D** for three-dimensional surface plots of complex functions. The graphics above can also be generated using the command

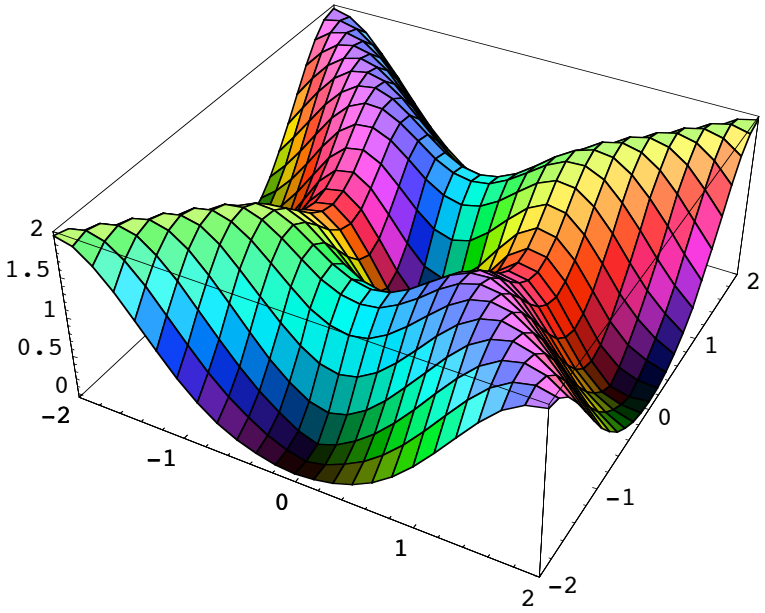
```
QComplexPlot3D[1/((x + I y)^5 - 2)^2, {x, -1.5, 1.5},
 {y, -1.5, 1.5}, PlotPoints → 200, Mesh → False]
```




```
- SurfaceGraphics -
```

Here is another example:

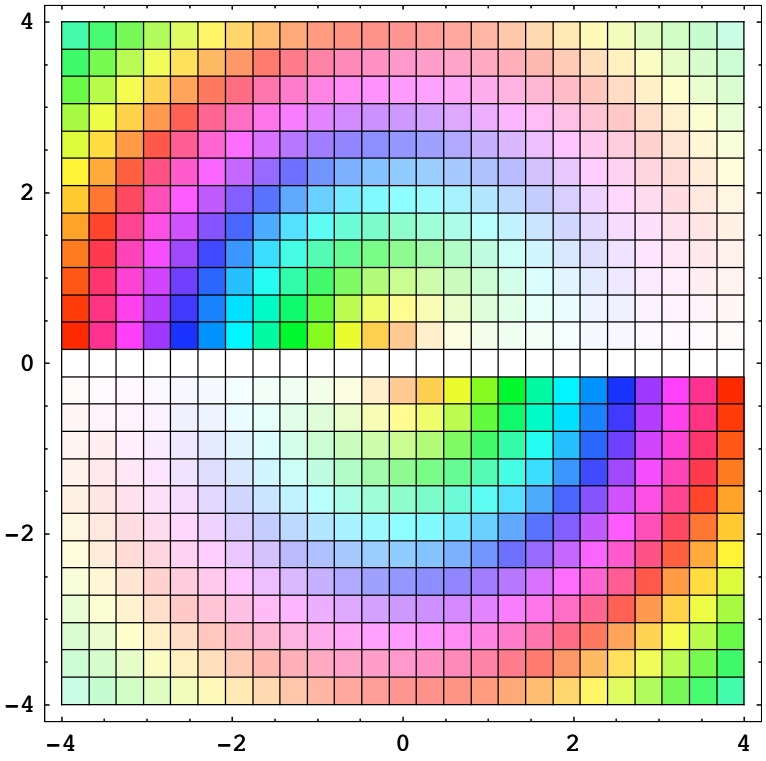
```
QComplexPlot3D[4 Exp[-x^2 - y^2] Sin[(x + I y)^2], {x, -2, 2}, {y, -2, 2}]
```




```
- SurfaceGraphics -
```

You can specify your own color map by setting the option **QComplexToColorMap**.

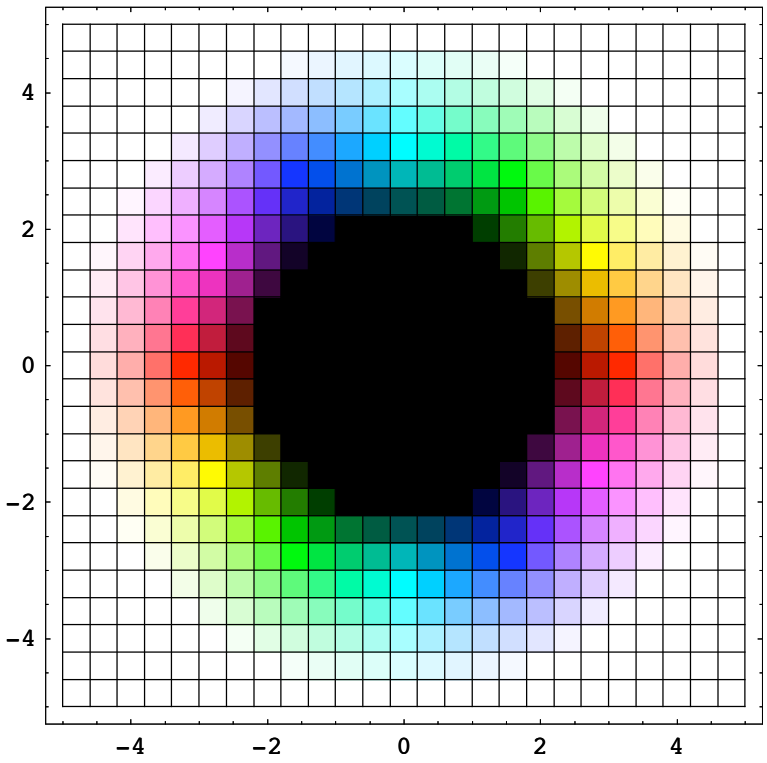
```
newcolormap = Function[{r, ϕ}, Hue[Mod[r / 4, 1], Mod[ϕ / π, 1], 1]];

QComplexDensityPlot[x + I y, {x, -4, 4}, {y, -4, 4}, QComplexToColorMap → newcolormap];
```



The default color map is given by the symbol **$QComplexToColorMap**. There are various options for modifying this built-in color map. The option **QSphereRadius** sets the radius of the sphere used for the stereographic projection. The default value is 1. Setting **QSphereRadius** to R causes all complex numbers with $|z| = R$ to be colored with lightness $1/2$ (maximal brightness and saturation). This setting is useful if we want to investigate complex functions in regions where the absolute value is very small or very large compared to 1.

If we are interested only in a certain range of values, we can set the option **QValueRange** to a range $\{v_{\min}, v_{\max}\}$, which displays complex numbers with absolute values larger than $v_{\max}$ at maximal lightness, and complex numbers with $|z| < v_{\min}$ at minimal lightness. The values for maximal and minimal lightness are controlled by the **QLightnessRange** option and are set by default to **{0,1}**.

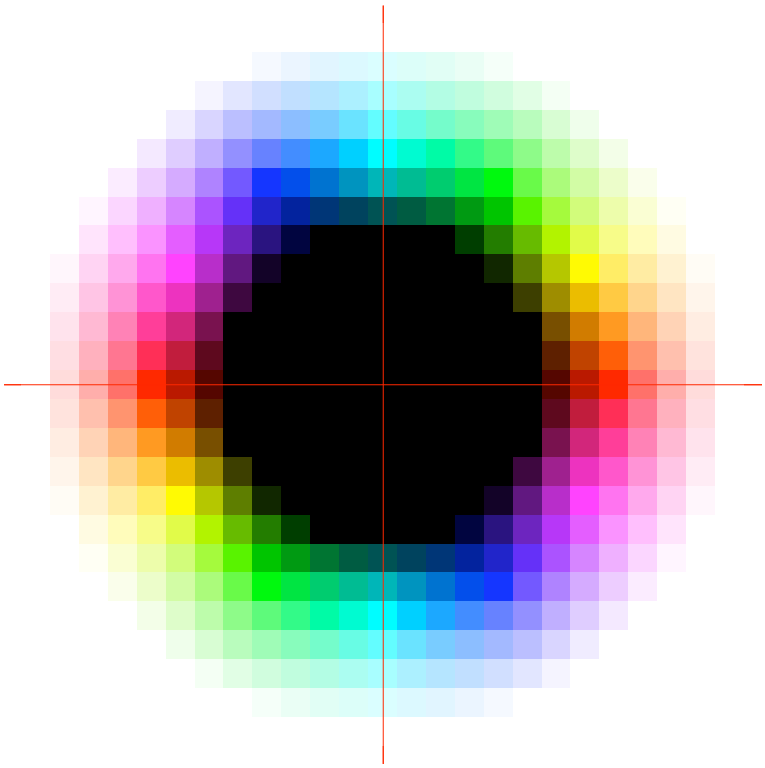
```
gr = QComplexDensityPlot[(x + I y)^2, {x, -5, 5},
    {y, -5, 5}, QValueRange → {5, 25}, QSphereRadius → 10];
```



It is also possible to set $v_{\max} < v_{\min}$. A plot of $f(x + i\,y)$ with **QValueRange** set to **{∞,0}** is the same as a plot of $1/f(x - i\,y)$ with the default setting of **{0,∞}**.

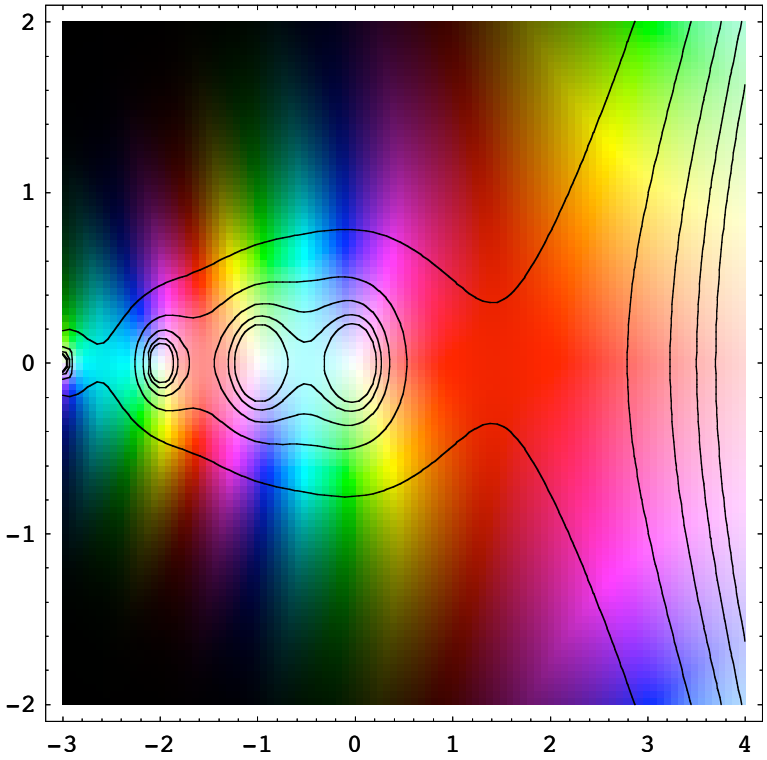
The **QColorDensityGraphics** object stores the table of colors obtained by applying the color map to the complex values. Thus, we can quickly redraw the plot with modified **Graphics** options, such as **AspectRatio**, **MeshStyle**, or **FrameLabels**. (We can't modify the options that control the color map in this way.)

```
Show[gr, Mesh → False, Frame → None, Axes → True, AxesStyle → RGBColor[1, 0, 0]];
```



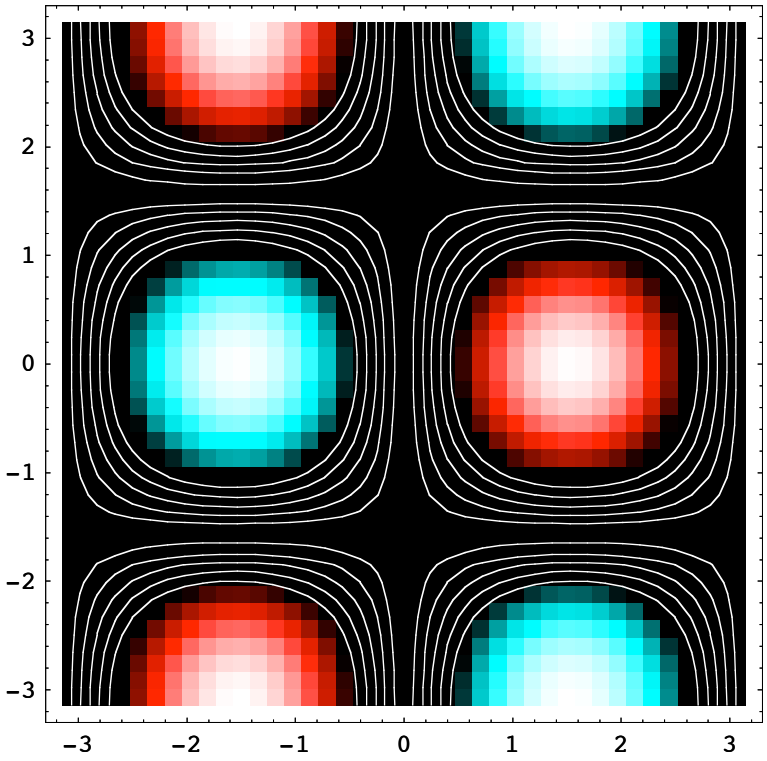
Sometimes, a contour plot displays the absolute value of a function more clearly than a density plot. The package provides a command for combining **ContourPlot** with **ComplexDensityPlot**:

```
QComplexContourPlot[Gamma[x + I y], {x, -3, 4},
  {y, -2, 2}, PlotPoints → 100, PlotRange → {0, 5}, Contours → 5];
```



Note that the option **PlotRange** only affects the range of the contours. You have to use the **QValueRange** option to influence the colors. In the following plot, contours are used to represent values where $0 \le |f(z)| \le 0.5$ and colors are used for the range $0.5 \le |f(z)| \le 1$.

```
QComplexContourPlot[Sin[x] Cos[y], {x, -Pi, Pi}, {y, -Pi, Pi},
    Contours -> 5, PlotRange -> {0, 0.5},
  QValueRange -> {0.5, 1}, ContourStyle -> GrayLevel[1], PlotPoints -> 40];
```



Finally, there is a command **QColorArrayPlot** for plotting a two-dimensional raster of color directives. It can be used to display color images in *Mathematica*. The following auxiliary function reads a picture file in non-interleaved "raw format," that is, a stream of bytes describing the color information, without a header. Many image editing programs (such as Adobe Photoshop) can save pictures in raw format.

```
GetPicture[pathname_, width_, height_] :=
Module[{pixels = ReadList[pathname, Byte], rd, gr, bl},
    rd = Table[pixels[[i + width j]],
        {j, height - 1, 0, -1}, {i, 1, width}] / 255.;
    gr = Table[pixels[[i + width (j + height)]],
        {j, height - 1, 0, -1}, {i, 1, width}] / 255.;
    bl = Table[pixels[[i + width (j + 2 height)]],
        {j, height - 1, 0, -1}, {i, 1, width}] / 255.;
    MapThread[RGBColor, {rd, gr, bl}, 2]
]
```

Here is an example:

```
colorarray = GetPicture["insert_pathname_to_file_here/shuttle.raw", 72, 50];
```

```
QColorArrayPlot[colorarray, Frame -> None,
    AspectRatio -> Automatic, Mesh -> False];
```

The Zeta Function

We will illustrate the usefulness of the package `ComplexPlot.m` by investigating graphically the famous Riemann zeta function $\zeta(z)$, which has a rich and interesting structure [Karatsuba and Voronin 1992]. This function is defined as the analytic continuation to the whole complex plane of the function

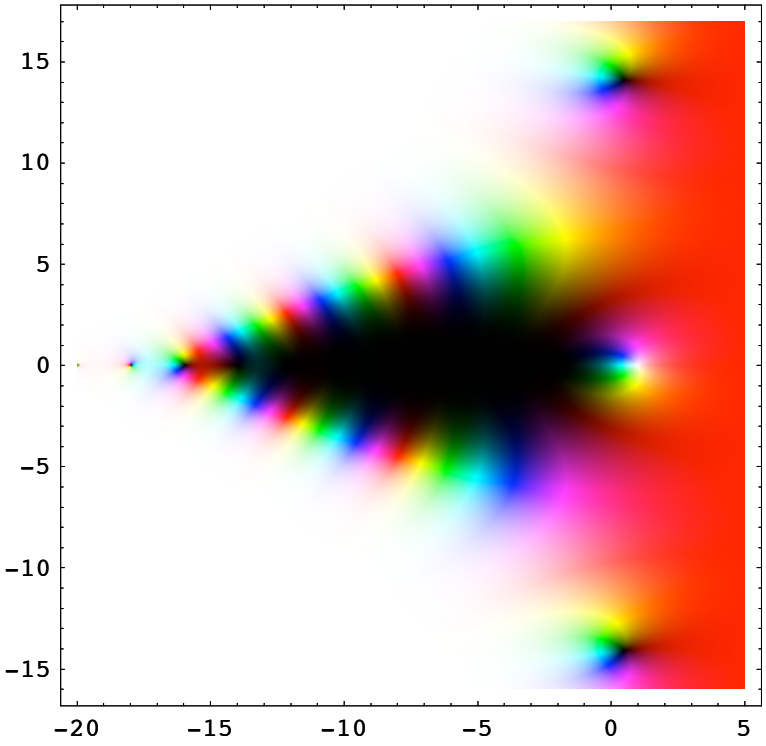
$$\sum_{n=1}^{\infty} \frac{1}{n^x}$$

```
Zeta[x]
```

which is initially given only for $x > 1$. The zeta function is meromorphic everywhere with a single pole at $z = 1$. The Riemann hypothesis is the conjecture that all zeros with $\mathrm{Re}(z) > 0$ lie on the "critical line" $\mathrm{Re}(z) = 1/2$ [Aizenberg et al. 1997]. The only zeros with $\mathrm{Re}(z) \leq 0$ are simple zeros at the negative even integers. The Riemann zeta function plays a prominent role in number theory, because its properties are related to the distribution of prime numbers.

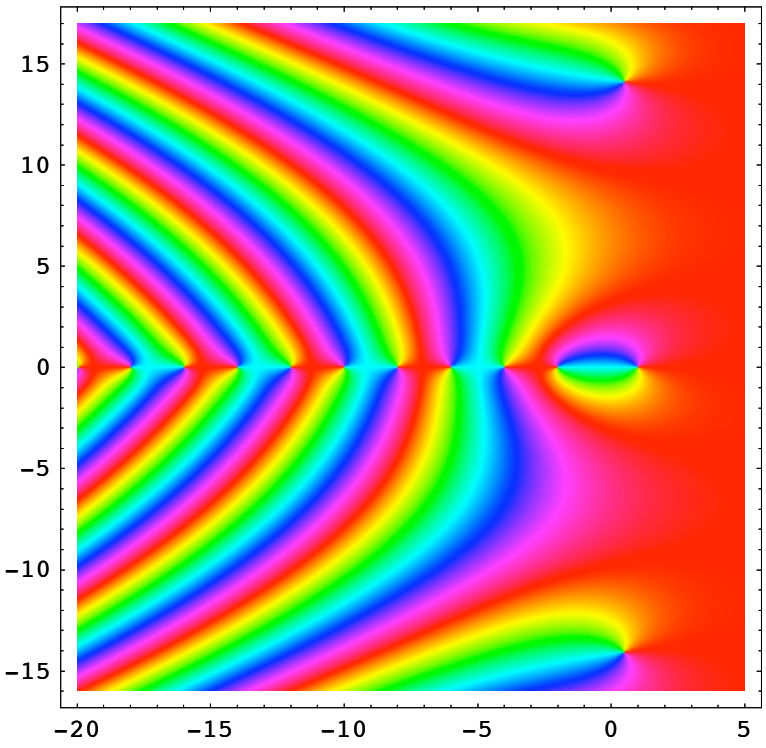
The function $\zeta(z)$ has absolute values ranging over many orders of magnitude within small regions. Usually, surface plots of such functions are not very useful, so we use a **`QColorDensityPlot`** for the visualization. (The following high-resolution plot takes several minutes on a Power Mac 8500. You should decrease the number of plot points on slower computers.)

```
QComplexDensityPlot[Zeta[x + I y], {x, -20, 5}, {y, -16, 17}, Mesh → False, PlotPoints → 300];
```



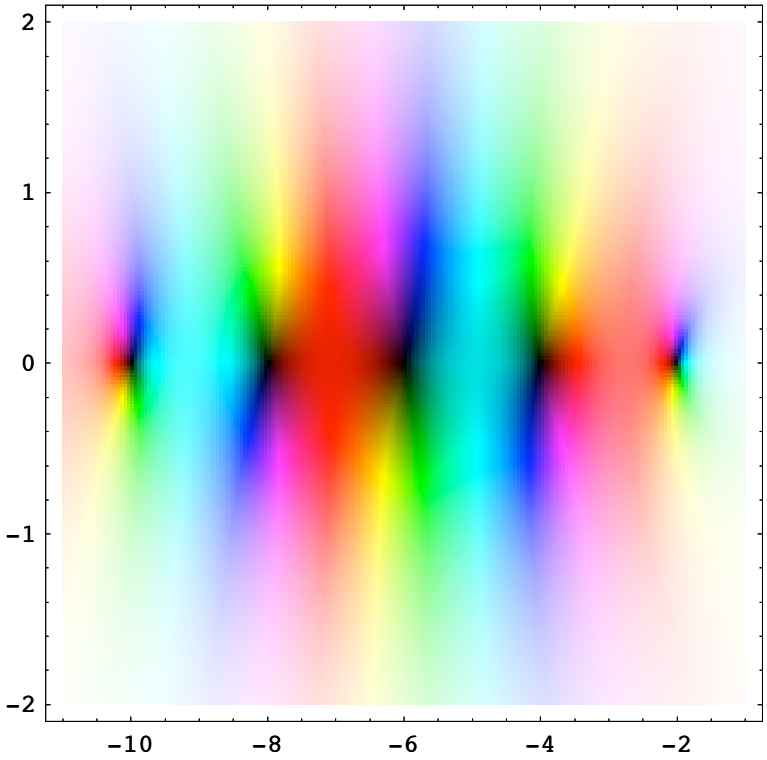
We can see the pole at $z = 1$ and the first of the zeros along the critical line $\mathrm{Re}(z) = 1/2$. The phase information is hidden in the regions that are black or white, where the absolute value of $\zeta(z)$ is very small or large compared to 1. We can make this information visible by restricting the maximal and minimal lightness of the colors in the graphic. If we set **`QLightnessRange`** to **`{0.5,0.5}`**, the information on the absolute value is totally supressed. The following plot gives the phase information for the same region as in the plot above:

```
QComplexDensityPlot[Zeta[x + I y], {x, -20, 5}, {y, -16, 17},
  Mesh → False, PlotPoints → 300, QLightnessRange → {0.5, 0.5}];
```



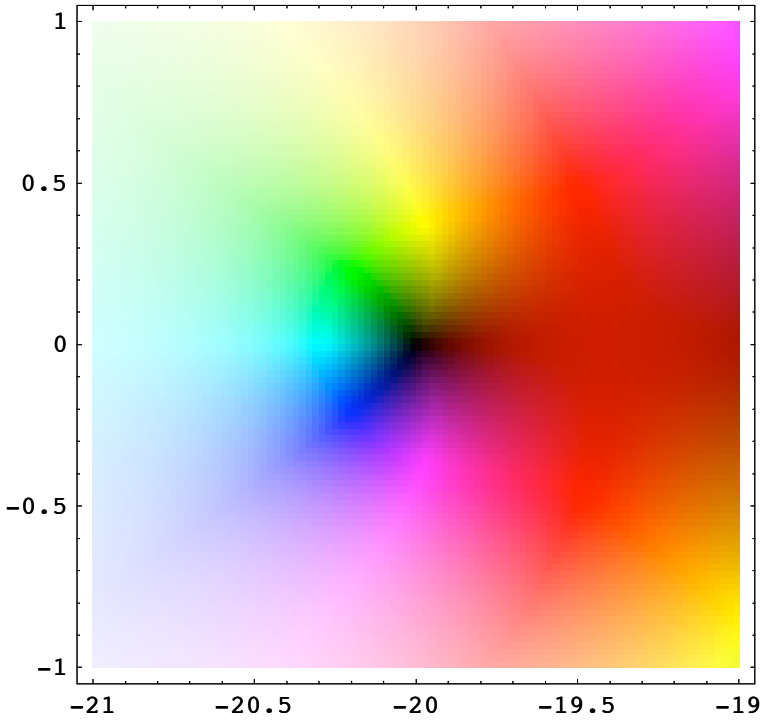
We can have a closer look at the region of z for which $|\zeta(z)|$ is very small by setting **QSphereRadius** to some small value. The following plot enlarges the black region of the first graphics and displays the values with $|\zeta(z)| = 0.005$ with maximal brightness and saturation:

```
QComplexDensityPlot[Zeta[x + I y], {x, -11, -1},
  {y, -2, 2}, Mesh → False, PlotPoints → 200, QSphereRadius → 0.005];
```



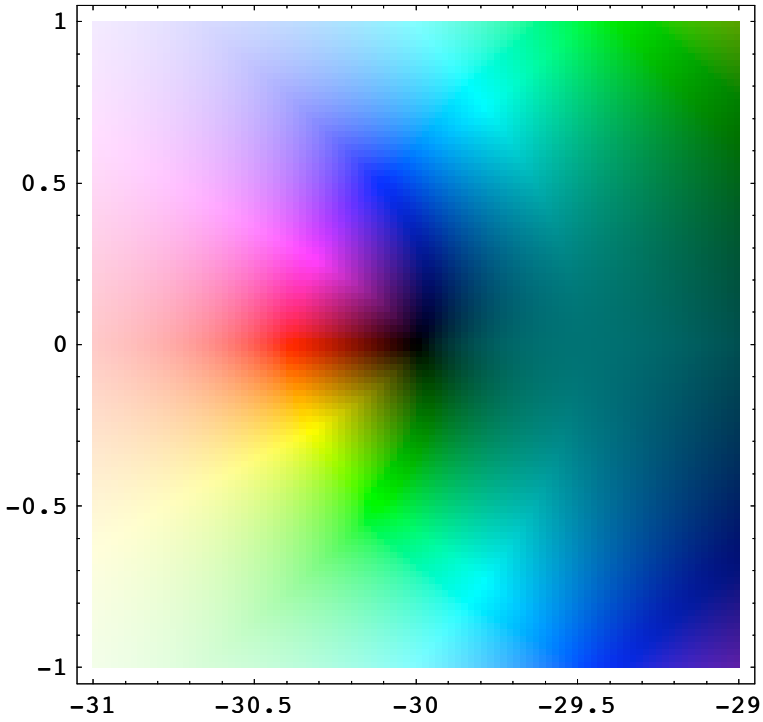
This plot shows clearly the zeros (all of first order) of the zeta function in the region under consideration. The zeros are located at the negative even numbers. The zeros at -16, -18, -20 are already visible in the first plot. To look at the behavior of the zeta function near the zero at $z = -20$, we have to increase **QSphereRadius**. Otherwise, the following plot would show only a tiny black point in a white surrounding:

```
QComplexDensityPlot[Zeta[x + I y], {x, -21, -19},
  {y, -1, 1}, Mesh → False, PlotPoints → 100, QSphereRadius → 50];
```



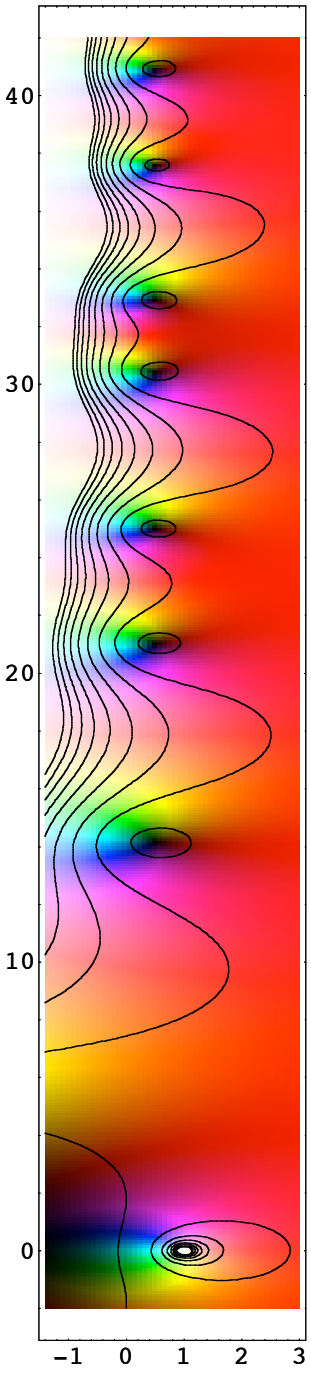
This is an even more extreme example:

```
QComplexDensityPlot[Zeta[x + I y], {x, -31, -29}, {y, -1, 1},
Mesh -> False, PlotPoints -> 100, QSphereRadius -> 10^8];
```



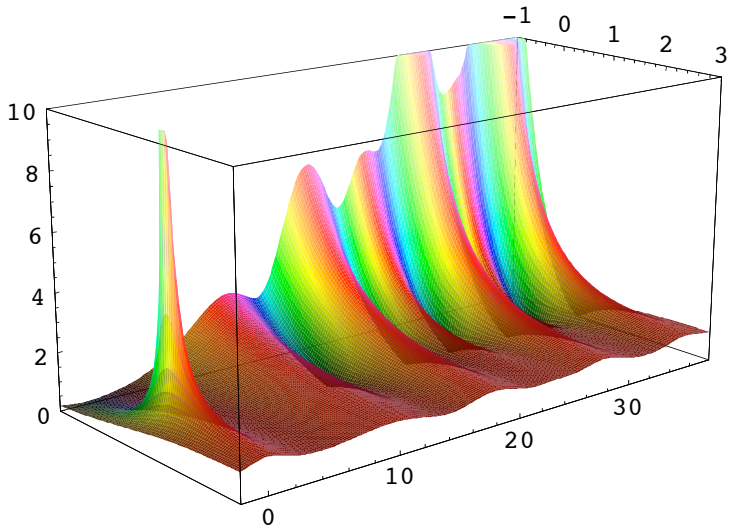
To display the behavior near the critical line and the precise position of the zeros, a contour plot seems most appropriate.

```
QComplexContourPlot[Zeta[x + I y], {x, -1.4, 3}, {y, -2, 42},
  PlotPoints → {50, 500}, PlotRange → {-1.3, 8}, AspectRatio → 5];
```



The colors of the graphic show that the poles and zeros of the Riemann zeta function are of first order. Finally, we show a high-resolution surface plot of the zeta function in the same region.

```
QComplexPlot3D[Zeta[x + I y], {x, -1, 3}, {y, -2, 38},
  Mesh → False, PlotPoints → {100, 250}, PlotRange -> {0, 10},
    QSphereRadius -> 3, ViewPoint -> {2.504, -2.085, 0.910}, BoxRatios -> {1, 2, 1}];
```



Many other examples can be seen on my web page dedicated to the visualization of complex functions: **http://vqm.uni-graz.at/pages/complex/**

References

Aizenberg, L., V. Adamchik, and V.E. Levit. 1997. Approaching the Riemann hypothesis with *Mathematica*. *The Mathematica Journal* 7(1): 54-57.

Karatsuba, A.A., and S.M. Voronin. 1992. *The Riemann Zeta-Function*. De Gruyter Expositions in Mathematics 5. Walter de Gruyter. Berlin. New York.

McIsaac, Kevin. ComplexPlot3D.m. *Mathematica* package. Available from *MathSource* as item 0200-484, **http://www.wolfram.com/cgi-bin/msitem?0200-484**. See also Roman Maeder, *The Mathematica Programmer*, p.187. Academic Press 1994.
Presently (2004), this package seems to have disappeared from *MathSource* and from the internet.

Pauschenwein, J., and B. Thaller, 1996. Visualizing quantum-mechanical wavefunctions in three dimensions with AVS. *Computers in Physics* 10(6): 558-566.

Smith, C., and N. Blachman. 1995. *The Mathematica Graphics Guidebook*. Addison-Wesley.

Wagner, D.B. 1996. Contexts and shadowing. *The Mathematica Journal* 6(2): 41-51.

Bernd Thaller
Institute for Mathematics and Scientific Computing
University of Graz
Heinrichstr. 36
A-8010 Graz
Austria

bernd.thaller@uni-graz.at
http://vqm.uni-graz.at/