Graphing on the Riemann Sphere

Djilali Benayat

We give a procedure to plot parametric curves on the sphere whose advantages over classical graphs in the Cartesian plane are obvious whenever the graph involves infinite domains or infinite branches.

■ Introduction

Graphing a curve in the Cartesian plane can be done only in a restricted "window" $[a, b] \times [c, d]$. If the function to be plotted has a large domain or range, it is practically impossible to get a global view of the curve. This makes it difficult to understand the asymptotic behavior of complicated curves with various kinds of infinities. Furthermore, for most functions (e.g., polynomials of degree greater than four), graphing in a large window loses important details, while graphing in a small window loses the global features.

The remedy is to compactify the plane and represent graphs on the Riemann sphere. The usual method is to map the plane graph to the sphere using the inverse stereographic projection. We prefer a slightly modified version: we smoothly wrap the plane $x = 1$ on the sphere $x^2 + y^2 + z^2 = 1$ using the inverse stereographic projection from the pole $(-1, 0, 0)$. The origin $(0, 0)$ maps to the blue point $(1, 0, 0)$ on the sphere, and the point at infinity maps to the red point $\omega = (-1, 0, 0)$.

■ Benefits of the Method

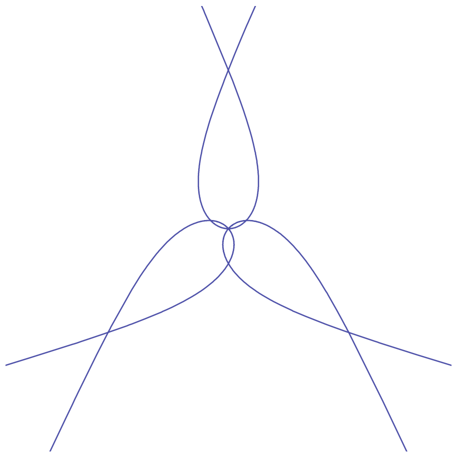
□ Asymptotic Behavior

Although the point at infinity cannot be reached, the mapping gives points so close to ω that it is as if we had reached it. As an illustration, here are the graphs of a polar curve first in the plane (with asymptotes) and then on the sphere.

```
In[1]:= polarcurve = Tan[Pi t / 4];
```

```
In[2]:= PolarPlot[polarcurve, {t, -2 π, 2 π},
         Axes → False, PlotRange → 4 {{-1, 1}, {-1, 1}}]
```

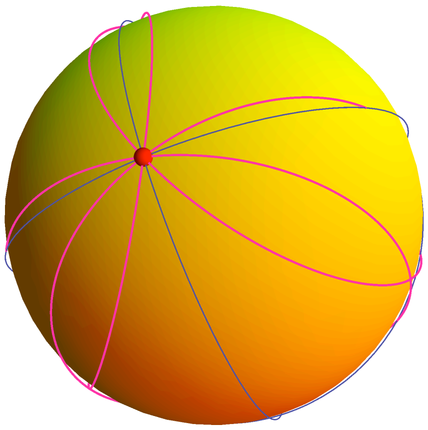
Out[2]=



The default view point shows ω, the image of the point at infinity.

```
In[3]:= defaultviewpoint = SpherePolarPlot[polarcurve, {t, -2 π, 2 π, .01 π}]
```

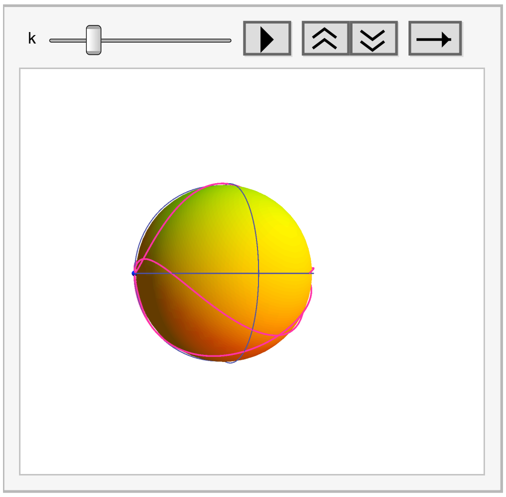
Out[3]=



Here is an animation where the view point goes once around the equator. The blue point zero is the image of the origin in the plane.

```
In[4]:= Animate[SpherePolarPlot[polarcurve,
           {t, -2 π, 2 π, .01 π}, ViewPoint → 2 {Cos[k], Sin[k], 0}],
          {k, 0, 2 π, .025 π}, SaveDefinitions → True]
```

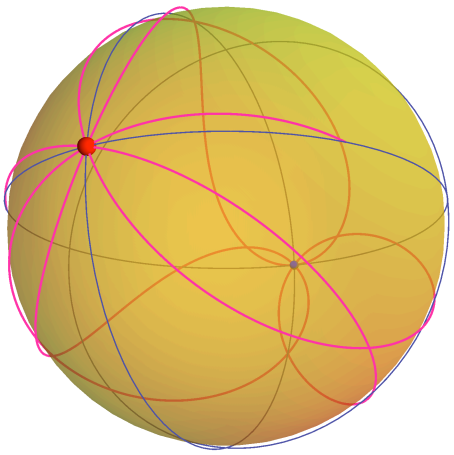
Out[4]=



To see through the sphere we use the `Opacity` option.

```
In[5]:= SpherePolarPlot[polarcurve, {t, -2 π, 2 π, .01 π}, Opacity → 0.5]
```

Out[5]=



□ Sensitivity and Faithfulness

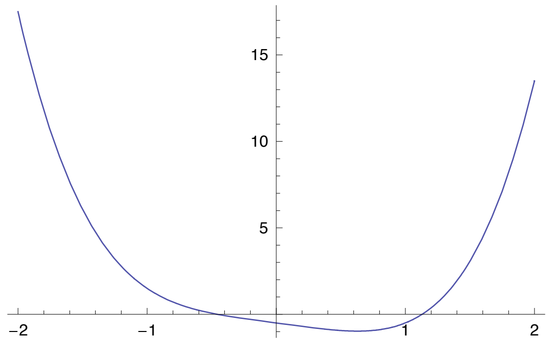
This modified inverse stereographic projection μ is very sensitive close to the origin: two points near the origin in the plane will appear far apart on the sphere. On the other hand, points near infinity will appear close together, thus showing the asymptotic behavior that is the qualitative property of the curve far away from the origin.

Faithfulness means that mapping the plane curve to the sphere does not alter the shape of the curve. In particular, at ω, the slopes of the different branches are exactly what they should be. We illustrate this by plotting the function $x^4 - x - 0.5$.

In[6]:=
```
quartic = x^4 - x - 0.5;
```

In[7]:=
```
Plot[quartic, {x, -2, 2}]
```

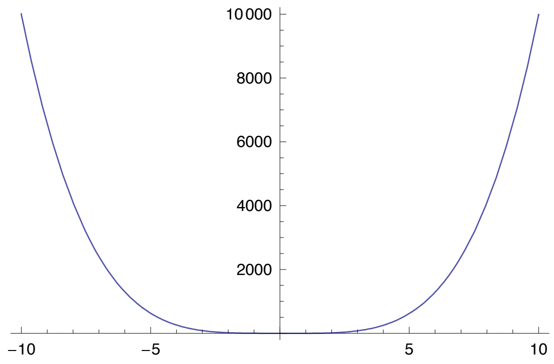
Out[7]=



Making the interval somewhat larger loses details close to the origin.

In[8]:=
```
Plot[quartic, {x, -10, 10}]
```

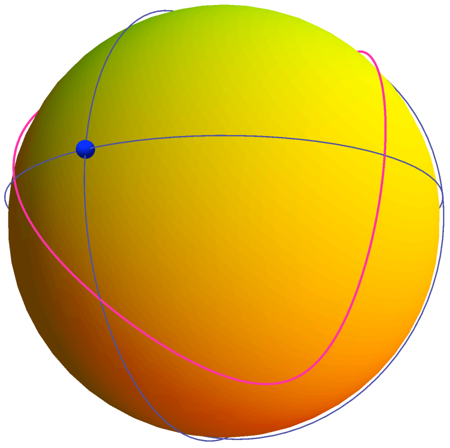
Out[8]=



Here is the same curve on the sphere, viewed to show zero and then ω.

```
In[9]:= SpherePlot[quartic, {x, -10, 10, 0.01}, ViewPoint → {1, 0.5, -0.3}]
```

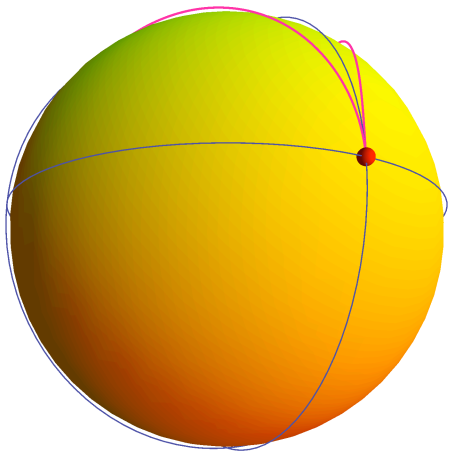
Out[9]=



The derivatives at $\pm\infty$ are ∞, and, indeed, we see that the curve is vertical near ω.

```
In[10]:= SpherePlot[quartic, {x, -10, 10, 0.01}, ViewPoint → {-1, 0.5, -0.3}]
```

Out[10]=



□ Unveiling the True Nature of a Curve

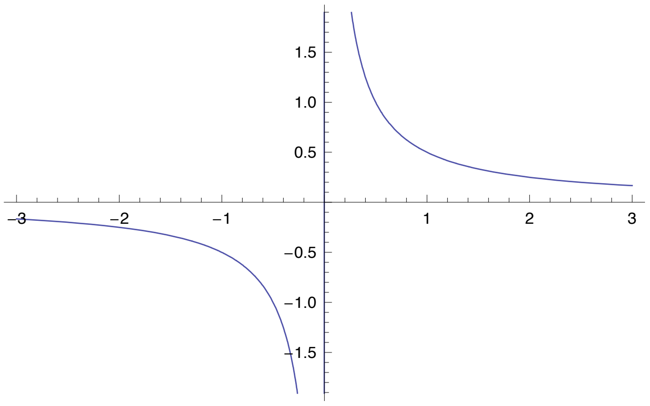
Geometers know that plane geometry is incomplete. We have to look at a curve in the projective plane (i.e., the Riemann sphere) to get complete results.

For example, students in high school are told that hyperbolas have two branches.

```
In[11]:= hyperbola = 1 / (2 x);

In[12]:= Plot[hyperbola, {x, -3, 3}]
```

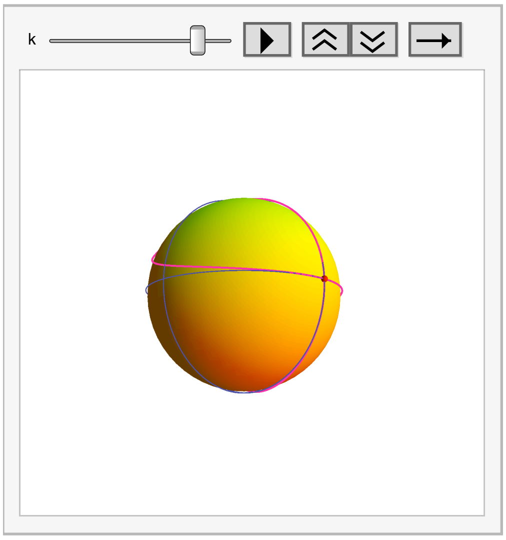
Out[12]=



In this animation, a hyperbola appears as a one-branched figure-eight curve on the sphere.

```
In[13]:= Animate[SpherePlot[hyperbola, {x, -100, 100, .1},
           ViewPoint → 2 {Sin[k + 3 / 2 Pi], Cos[k + 1 / 2 Pi], -.2}],
          {k, 0, 2 Pi, .05 Pi}, SaveDefinitions → True]
```

Out[13]=



The Mapping

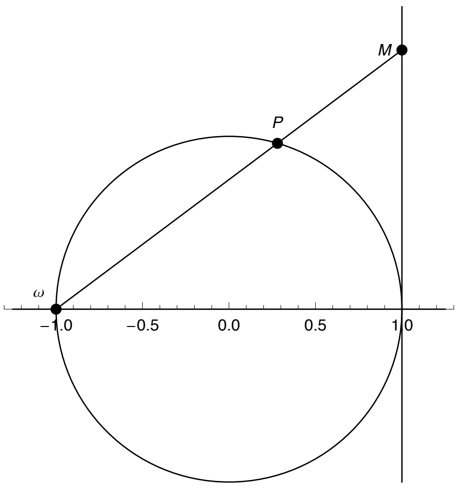
We define the mapping $\mu : \mathbb{R}^2 \to S^2$ using the following picture.

```
In[14]:= Solve[{y / (x + 1) == (3 / 2) / (1 + 1), x^2 + y^2 == 1}, {x, y}]
```

$$\text{Out[14]=} \left\{\left\{y \to \frac{24}{25}, x \to \frac{7}{25}\right\}\right\}$$

```
In[15]:= Show[Graphics[{Circle[{0, 0}, 1], Line[{{-1, 0}, {1, 3 / 2}}],
            Line[{{1, -1}, {1, 7 / 4}}], Line[{{-5 / 4, 0}, {5 / 4, 0}}],
            PointSize[.025], Point[{-1, 0}], Point[{1, 3 / 2}],
            Point[{7, 24} / 25], Text[TraditionalForm["ω"], {-1 - .1, .1}],
            Text[TraditionalForm[P], {7, 24 + 3} / 25],
            Text[TraditionalForm[M], {1 - .1, 3 / 2}]},
          Axes → {True, False}, AspectRatio → Automatic]]
```

Out[15]=



The projection pole is $\omega = (-1, 0, 0)$, $M = (1, u, v) \in \{1\} \times \mathbb{R}^2$ and $P = (x, y, z)$ is the point corresponding to M on S^2. Then $\overrightarrow{\omega P} = t\,\overrightarrow{\omega M}$ and to find the mapping $\mu : (u, v) \mapsto (x, y, z)$, we solve for t.

```
In[16]:= Solve[
           {x^2 + y^2 + z^2 == 1, x == 2 t - 1, y == u t, z == v t}, t, {x, y, z}]
```

$$\text{Out[16]=} \left\{\{t \to 0\}, \left\{t \to \frac{4}{4 + u^2 + v^2}\right\}\right\}$$

One solution gives the point ω and the other solution determines the mapping to the sphere.

The Program: Parametric Curves on the Sphere

The modified inverse stereographic function μ turns out to be fairly simple.

```
In[17]:= μ[{u_, v_}] := With[{s = 4 + u^2 + v^2}, {8 - s, 4 u, 4 v} / s]
```

We define the sphere and the equator, two meridians, a blue zero, and a red ω for orientation.

```
In[18]:= sphere[n_] :=
          Graphics3D[{Yellow, Opacity[n], Sphere[{0, 0, 0}, 0.985]}];

In[19]:= equator =
          Graphics3D[{Thickness[0.003`], Blue, First[ParametricPlot3D[
              {Cos[p], Sin[p], 0}, {p, 0, 2 π}, PlotPoints → 100]]}];

In[20]:= meridian1 =
          Graphics3D[{Thickness[0.003`], Green, First[ParametricPlot3D[
              {Cos[p], 0, Sin[p]}, {p, 0, 2 π}, PlotPoints → 100]]}];

In[21]:= meridian2 =
          Graphics3D[{Thickness[0.003`], Cyan, First[ParametricPlot3D[
              {0, Cos[p], Sin[p]}, {p, 0, 2 π}, PlotPoints → 100]]}];

In[22]:= zero = Graphics3D[{Blue, Sphere[{1, 0, 0}, 0.03]}];

In[23]:= ω = Graphics3D[{Red, Sphere[{-1, 0, 0}, 0.03]}];
```

To avoid infinities, `SphereParametricPlot` is based on `ScatterPlot3D` rather than `ParametricPlot3D`. In analogy with `Plot` and `PolarPlot`, we define the functions `SpherePlot` and `SpherePolarPlot`, both in terms of `SphereParamet`⋱ `ricPlot`.

The first argument to `SphereParametricPlot` is a pair of functions. The first argument to `SpherePlot` or `SpherePolarPlot` is a single function. The second argument to all three functions is a range specification. A `ViewPoint` option may be given.

```
In[24]:= SphereParametricPlot[{fx_, fy_},
           {u_, umin_, umax_, du_}, opts___] :=
          Show[{sphere[Opacity /. {opts} /. Opacity → 1], equator,
            meridian1, meridian2, zero, ω, Graphics3D[
             {Hue[0.9`], Thickness[0.005`], Line[Table[μ[{fx, fy}],
                {u, umin, umax, du / 10}]]}]}, Boxed → False,
           ViewPoint → (ViewPoint /. {opts} /. ViewPoint → -{1, 1/2, 1/3}),
           PlotRange → All]

In[25]:= SphereParametricPlot[{fx_, fy_}, {u_, umin_, umax_}, opts___] :=
          SphereParametricPlot[{fx, fy}, {u, umin, umax, (umax - umin)/150}, opts]

In[26]:= SpherePlot[r_, {t_, tmin_, tmax_, dt_}, opts___] :=
          SphereParametricPlot[{t, r}, {t, tmin, tmax, dt}, opts]
```

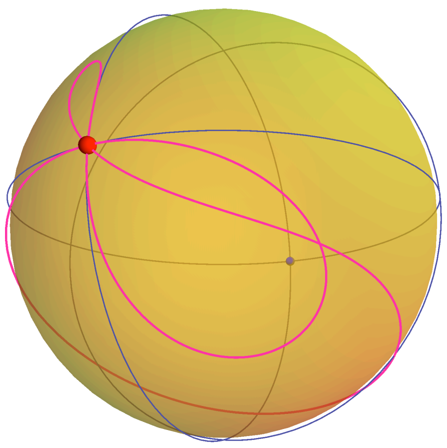
```
In[27]:= SpherePlot[r_, {t_, tmin_, tmax_, dt_}, opts___] :=
          SphereParametricPlot[{t, r}, {t, tmin, tmax, dt}, opts]

In[28]:= SpherePolarPlot[r_, {t_, tmin_, tmax_, dt_}, opts___] :=
          SphereParametricPlot[
           {r Cos[t], r Sin[t]}, {t, tmin, tmax, dt}, opts]
```

Here is one final example.

```
In[29]:= SphereParametricPlot[{(t^2 - 2) / (t + 1), (t^3 - 8) / (t^2 - 3 t + 2)},
          {t, -100, 100, .1}, Opacity → .5]
```

Out[29]=



Additional Material

ImplicitEquOnSphere.nb

Available at
www.mathematica-journal.com/data/uploads/2008/11/ImplicitEquOnSphere.nb.

References

D. Benayat, "Graphing on the Riemann Sphere," *The Mathematica Journal*, 2011. dx.doi.org/doi:10.3888/tmj.10.4–5.

About the Author

Djilali Benayat received a B.Sc. in mathematics (1968) and a Diploma of Advanced Studies (DEA) in functional analysis (1969) from Algiers University, a DEA in algebraic topology (1970) and a "Doctorat de 3e cycle" in *K*-theory (1972) from Strasbourg University, and a D.Sc. in algebraic topology (Theory of Adams Algebras in Stable Homotopy) (1987) from Metz and Algiers universities. Presently he is a professor of mathematics at the École Normale Supérieure of Algiers, Algeria. He has supervised a master's thesis and three D.Sc.'s in algebraic topology and number theory.

Djilali Benayat
École Normale Supérieure
Vieux Kouba
Algiers, Algeria
dbenayat@yahoo.fr