Newton's Method

Math 118−−Dynamical Systems

Matthew Leingang, Course Assistant

■ Initialization

```
In[1]:= << Graphics`Colors`

In[2]:= << "~/math/118/Bounce.m"

In[3]:= ?Arrow

        Arrow[start, finish, (opts)] is a graphics primitive representing an arrow starting at
          start and ending at finish.(Added by Bounce.m:) Arrow[{pt1, pt2,... }, (opts)]
          creates the graphics primitives drawing arrows from pt1 to pt2, from pt2 to pt3, etc.
```

■ Preliminaries

Let f be a differentiable function on the real line. We would like to find a point x_* where $f(x_*) = 0$.

Newton considered the following algorithm: Look at the graph of f. Given any x_0, draw the tangent to the graph at x_0 and follow it back down to the x−axis. Let this intersection point be $N_f(x_0)$.

What does this mean analytically? The tangent line is given by (good ol' point−slope formula):

```
In[4]:= Solve[y - f[x0] == f'[x0] (x - x0), y]

Out[4]= {{y → f[x0] + x f'[x0] - x0 f'[x0]}}
```

And to find the point where $y = 0$, we just solve again.

```
In[5]:= Solve[(y /. Flatten[%]) == 0, x]
```

$$\text{Out[5]= } \left\{\left\{x \to -\frac{f[x0] - x0\, f'[x0]}{f'[x0]}\right\}\right\}$$

```
In[6]:= Simplify[%]
```

$$\text{Out[6]= } \left\{\left\{x \to x0 - \frac{f[x0]}{f'[x0]}\right\}\right\}$$

So this is how we can code our Newton's map.

```
In[7]:= NM[f_, x_] :=
          x - f[x] / f'[x]
```

```
In[8]:= NM_f_[x_] := NM[f, x]
```

And here's some code for drawing a "Newton's bouncing" diagram. Don't worry so much about the pattern matching before the := symbol. Things that might be useful: **Epilog** takes the graphics directives and puts them on top of the plot. And don't forget what **Arrow** on a list of points does.

```
In[9]:= NewtonDiagram[f : (_Symbol | _Function), {x_Symbol, xmin_?NumericQ, xmax_?NumericQ},
        p_?NumericQ, n_Integer?NonNegative,
        opts___] :=
         Plot[f[x], {x, N[xmin], N[xmax]},
          Epilog ->
           {Red, Arrow[
             NestList[(If[#[[2]] == 0,
               {#[[1]], f[#[[1]]]},
               {NM[f, #[[1]]], 0}])&,
              {N[p], 0}, n],
             HeadScaling -> Relative]}
          opts]
```

■ Convergence Issues

Just to be sure, what are the fixed points of N_f?

```
In[10]:= Solve[NM_f[x] == x, {f[x]}]
```

```
Out[10]= {{f[x] → 0}}
```

As predicted, they are precisely the fixed points of *f*. Moreover,

```
In[11]:= NM_f'[x]
         % /. %%
```

$$\text{Out[11]=}\ \frac{f[x]\ f''[x]}{f'[x]^2}$$

```
Out[12]= {0}
```

So as long as *f'*(*x*) is not zero at the root of *f*, we have a *super*attracting fixed point. This is very good news.

■ A quadratic

Let

```
In[13]:= f[x_] := x^2 - 1
```

■ Exercise 1

```
In[14]:= NM_f[x]
```

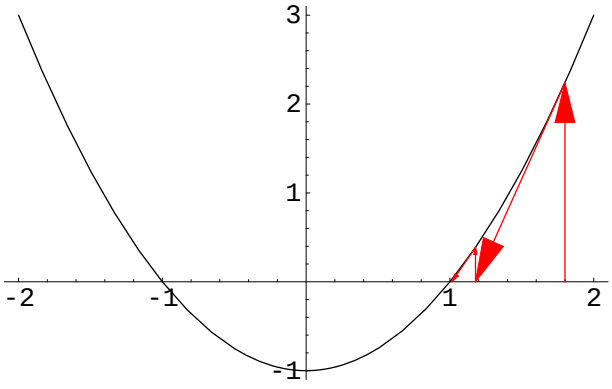
$$\text{Out[14]=}\ x - \frac{-1 + x^2}{2\,x}$$

```
In[15]:= Simplify[%]
```

$$\text{Out[15]}= \frac{1 + x^2}{2\,x}$$

Here's a plot that graphically shows how the Newton's map works:

```
In[16]:= NewtonDiagram[f, {x, -2, 2},
          1.8, 4]
```




```
Out[16]= - Graphics -
```

Let's find the critical behavior of the Newton function itself.

```
In[17]:= NMf'[x]
```

$$\text{Out[17]}= \frac{-1 + x^2}{2\,x^2}$$

```
In[18]:= Solve[% == 0, x]
         f[x] /. %
```

```
Out[18]= {{x → -1}, {x → 1}}
```

```
Out[19]= {0, 0}
```

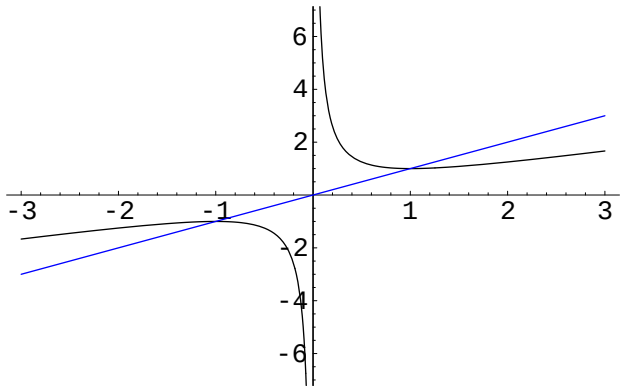
So the critical points of N_f are the roots of f. To find the poles of f, we can use a special *Mathematica* function.

```
In[20]:= Series[NMf[x], {x, 0, 2}]
```

$$\text{Out[20]}= \frac{1}{2\,x} + \frac{x}{2} + O[x]^3$$

So N_f has a pole of order 1 at zero.

```
In[21]:= Plot[{NMf[x], x}, {x, -3, 3},
          PlotStyle -> {{}, Blue}]
```



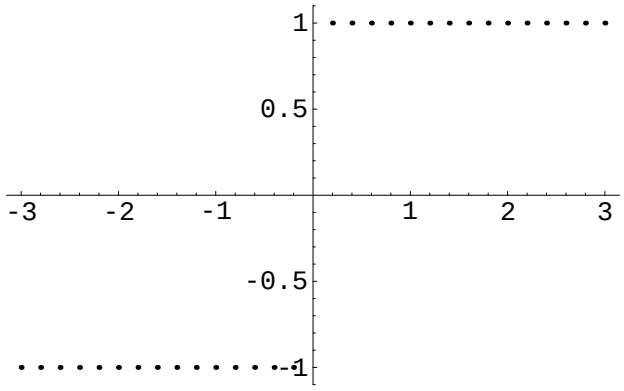

```
Out[21]= - Graphics -
```

■ Exercise 2

```
In[22]:= Table[{x, Nest[NMf, x, 20]}, {x, -3., 3., 0.2}];
```

If you've been writing enough *Mathematica* code, you know what this is supposed to do. It generates pairs $(x, N_f{}^{20}(x))$, where x ranges between −3 and 3 with steps of 0.2.

```
In[23]:= ListPlot[%,
          PlotRange -> {-1.1, 1.1}]
```




```
Out[23]= - Graphics -
```

■ Exercise 3

We observe the following from the above plot: It looks like if $x < 0$, then $N_f{}^n(x) \to -1$, and all positive numbers tend to 1 under iterations of N_f.

Proof: We may assume that $x > 0$; the negative case is handled symmetrically. In fact, we may also assume that $x > 1$, because if $x \neq 1$ but is positive, then $N_f(x) > 1$. On the other hand, notice that in the range $(1,\infty)$, we have that $N_f(x) < x$. So iterates of x will decrease with limit at 1.

■ A cubic

Let

```
In[24]:= g[x_] := x^3 - x
```

■ Exercise 4

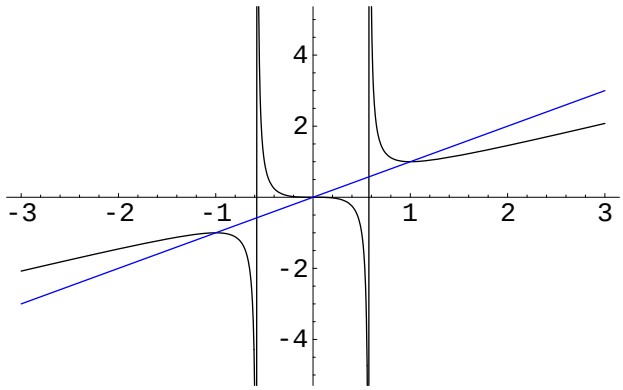
```
In[25]:= NM_g[x]
```

$$\text{Out[25]= } x - \frac{-x + x^3}{-1 + 3x^2}$$

```
In[26]:= Simplify[%]
```

$$\text{Out[26]= } \frac{2x^3}{-1 + 3x^2}$$

```
In[27]:= Plot[{NM_g[x], x}, {x, -3, 3},
          PlotStyle -> {{}, Blue}]
```




```
Out[27]= - Graphics -
```

Note that the vertical "lines" aren't really part of the graph−−they are caused by the discontinuity which *Mathematica* foolishly connects. Anyway, there are poles given by

```
In[28]:= Solve[g'[x] == 0, x]
```

$$\text{Out[28]= } \left\{\left\{x \to -\frac{1}{\sqrt{3}}\right\}, \left\{x \to \frac{1}{\sqrt{3}}\right\}\right\}$$

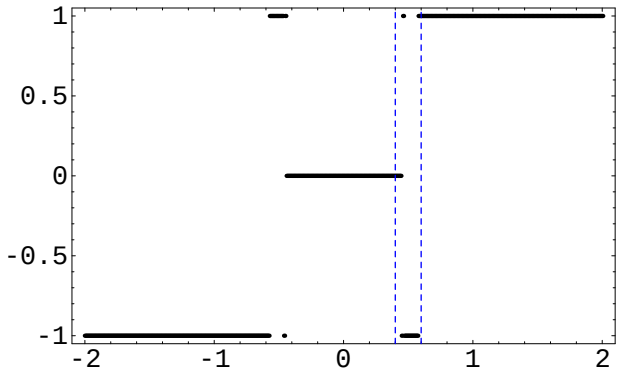
```
In[29]:= Series[Ng[x], {x, 1 / Sqrt[3], 2}]
```

$$\text{Out[29]= } \text{Ng}\left[\frac{1}{\sqrt{3}}\right] + \text{Ng}'\left[\frac{1}{\sqrt{3}}\right]\left(x - \frac{1}{\sqrt{3}}\right) + \frac{1}{2}\text{Ng}''\left[\frac{1}{\sqrt{3}}\right]\left(x - \frac{1}{\sqrt{3}}\right)^2 + O\left[x - \frac{1}{\sqrt{3}}\right]^3$$

Again, the pole is of order 1.

■ Exercise 5

```
In[30]:= pts = Table[{x, Nest[NMg, x, 20]}, {x, -2., 2., 0.01}];

In[31]:= ListPlot[pts,
          Axes -> False,
          Frame -> True,
         Epilog -> {Blue, Dashing[{0.01, 0.01}],
          Map[Line[{{#, -1.1}, {#, 1.1}}]&, {0.4, 0.6}]}]
```



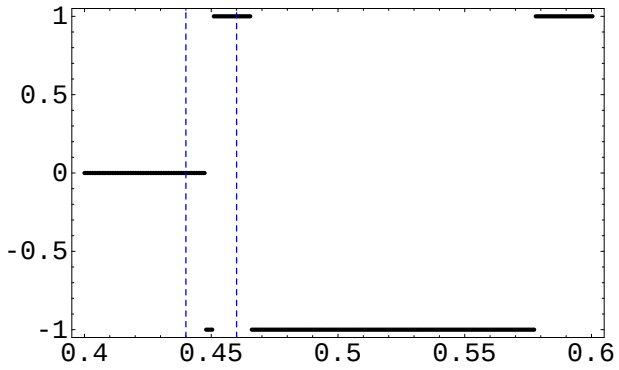

```
Out[31]= - Graphics -
```

This looks much different than our situation for the quadratic. Sure, we have a neighborhood around each root where the iterates converge to it, but what about between them? Let's blow up a region between two basins of attraction, the one indicated in the dashed rectangle above.

■ Exercise 6

```
In[32]:= N[1 / Sqrt[3]]

Out[32]= 0.57735

In[33]:= pts = Table[{x, Nest[NMg, x, 20]}, {x, 0.4, 0.6, 0.001}];
```

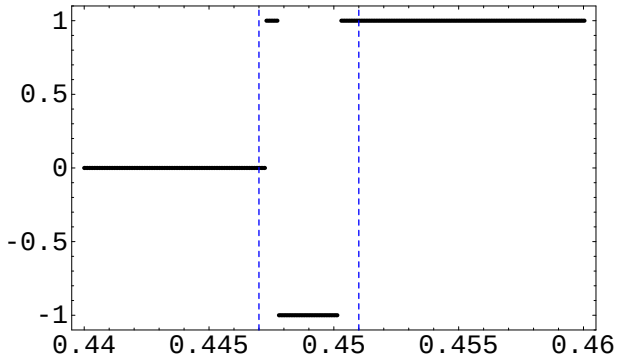
In[34]:=
```
ListPlot[pts,
 Axes -> False,
 Frame -> True,
 Epilog -> {Blue, Dashing[{0.01, 0.01}],
 Map[Line[{{#, -1.1}, {#, 1.1}}]&, {0.44, 0.46}]}]
```



Out[34]= - Graphics -

In[35]:=
```
pts = Table[{x, Nest[NM_g, x, 20]}, {x, 0.44, 0.46, 0.0001}];
```

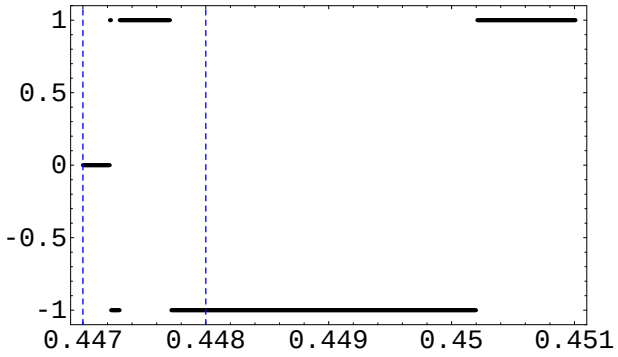
In[36]:=
```
ListPlot[pts,
 Axes -> False,
 Frame -> True,
 PlotRange -> {-1.1, 1.1},
 Epilog -> {Blue, Dashing[{0.01, 0.01}],
 Map[Line[{{#, -1.1}, {#, 1.1}}]&, {0.447, 0.451}]}]
```



Out[36]= - Graphics -

In[37]:=
```
pts = Table[{x, Nest[NM_g, x, 20]}, {x, 0.447, 0.451, 0.00001}];
```

```
In[38]:= ListPlot[pts,
          Axes -> False,
          Frame -> True,
          PlotRange -> {-1.1, 1.1},
         Epilog -> {Blue, Dashing[{0.01, 0.01}],
          Map[Line[{{#, -1.1}, {#, 1.1}}]&, {0.447, 0.448}]}]
```




```
Out[38]= - Graphics -
```

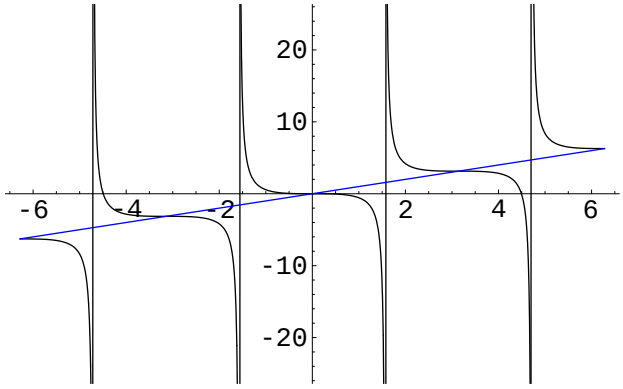
We will see more dramatic results later.

■ A transcendental function

Now we consider the sine function.

■ Exercise 7

```
In[39]:= Plot[{NMSin[x], x}, {x, -2 Pi, 2 Pi},
         PlotStyle -> {{}, {Blue}}]
```




```
Out[39]= - Graphics -
```

As before, its critical points are the roots of the sine function:
$\{n\pi \mid n \text{ any integer}\}$ and its poles are at the roots of the cosine function
$\{\frac{n\pi}{2} \mid n \text{ any integer}\}$.

```
In[40]:= NM_Sin[x]

Out[40]= x - Tan[x]

In[41]:= Series[NM_Sin[x], {x, π / 2, 1}]

Out[41]= 1 / (x - π/2) + π/2 + 2/3 (x - π/2) + O[x - π/2]^2
```

Again, each pole is of order 1 .

■ Exercise 8, with a digression on Compiled Functions

Sine is a very transcendental function. So to compute it exactly takes a lot of time and space. In many cases, when we want to do lots of number−crunching (especially in the next Computer Project!), we need to compile our functions. We lose a lot of the symbolic capabilities of *Mathematica* here, but we gain speed. Here is an example and an illustration.

```
NSin = Compile[{x, n},
Module[{z},
   z = x; Do[z = z - Tan[z], {n}]; z]]

General::spell1 :
 Possible spelling error: new symbol name "NSin" is similar to existing symbol "Sin".

CompiledFunction[{x, n}, Module[{z}, z = x; Do[z = z - Tan[z], {n}]; z], -CompiledCode-]
```

Compare this compiled function with its symbolic analogue. For high iterates, the speed difference becomes remarkable:

```
Timing[Nest[NM_Sin, 1.6, 200]]

{0.11 Second, 31.4159}

Timing[NSin[1.6, 200]]

{0. Second, 31.4159}
```

Warning: unless you write the code correctly, the compiled function will not necessarily save time! It's best to check using the **Timing** function as above.

Anyway, here's how we use it to get the plots for $N_{\sin}$.

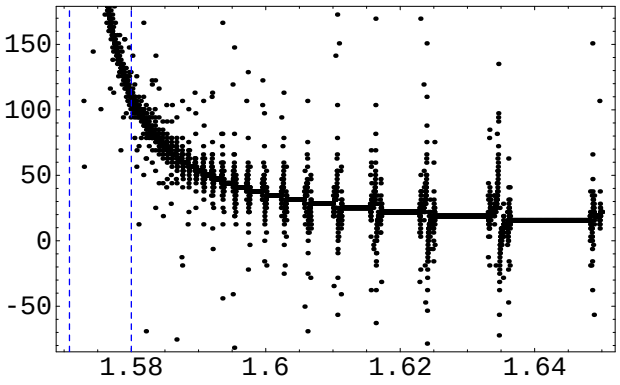
```
pts = Table[{x, NSin[x, 20]}, {x, 0., Pi, 0.001}];

ListPlot[pts,
 Axes -> False,
 Frame -> True,
FrameTicks -> {Automatic, Table[n Pi, {n, -2, 2}]},
Epilog -> {Blue, Dashing[{0.01, 0.01}],
 Map[Line[{{#, -10}, {#, 10}}]&, {Pi / 2, 2}]}]


- Graphics -

pts = Table[{x, NSin[x, 20]}, {x, N[Pi / 2], 2., 0.0001}];
```

```
ListPlot[pts,
 Axes -> False,
 Frame -> True,
Epilog -> {Blue, Dashing[{0.01, 0.01}],
 Map[Line[{{#, -100}, {#, 100}}]&, {Pi / 2, 1.65}]}]
```

```
- Graphics -
```

```
pts = Table[{x, NSin[x, 20]}, {x, N[Pi / 2], 1.65, 0.00001}];
```

```
ListPlot[pts,
 Axes -> False,
 Frame -> True,
Epilog -> {Blue, Dashing[{0.01, 0.01}],
 Map[Line[{{#, -200}, {#, 200}}]&, {Pi / 2, 1.58}]}]
```

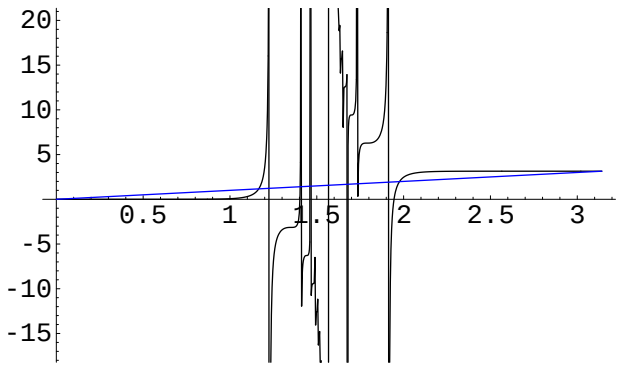



```
- Graphics -
```

■ Exercise 9

Consider the function $N_{\sin}{}^{2}$. Plotting it is a good example of the limitations of *Mathematica*.

```
In[45]:= Plot[{NM_Sin[NM_Sin[x]], x}, {x, 0, Pi},
          PlotStyle -> {{}, {Blue}}]
```




```
Out[45]= - Graphics -
```

Those "peaks" aren't local maxima−−they are asymptotes! This leads us to the conjecture that given *any* root of the sine function (that is, any $n\pi$), we can find a point x_0 near $\frac{\pi}{2}$ that approaches that root under iteration.

Proof: Here's the idea. Let *r* be our given root. Then there is a neighborhood *U* about *r* such that all points in *U* converge to *r* under iterations of $N_{\sin}$. Pick any y in U; then there exists arbitarily small slopes of lines through (*y*,0) that are tangent to the graph of the sine function. That means there are points arbitrarily far away that get mapped into + under $N_{\sin}$, hence converge to r. Any one of these arbitrarily far away points gets hit by a point near $\frac{\pi}{2}$, and accordingly we can pick one of them as close to $\frac{\pi}{2}$ as we like.

■ Complex Numbers and Color!

■ Exercises 10 and 11

Let

```
In[46]:= h[z_] := z^3 - 1
```

so that

```
In[47]:= NMh[z]
```

$$\text{Out[47]= } z - \frac{-1 + z^3}{3 z^2}$$

```
In[48]:= Simplify[%]
```

$$\text{Out[48]= } \frac{1}{3 z^2} + \frac{2 z}{3}$$

We can compile complex functions, too.

```
In[49]:= Nh = Compile[{{z, _Complex}, {n, _Integer}},
          Module[{t},
             t = z; Do[t = 1 / (3 t^2) + 2 t / 3, {n}]; t]]
```

$$\text{Out[49]= } \text{CompiledFunction}\left[\{z, n\}, \text{Module}\left[\{t\}, t = z; \text{Do}\left[t = \frac{1}{3 t^2} + \frac{2 t}{3}, \{n\}\right]; t\right], -\text{CompiledCode}-\right]$$

```
In[50]:= data = Table[Nh[x + I y, 10],
           {y, 1.2, -1.2, -0.1}, {x, -1.2, 1.2, 0.1}];
```

This function should have generated a table of the "limit" (approximated by the 10th iterate) of the function with initial conditions $x + iy$, for *x* and *y* in the given range.

In[51]:= **TableForm**[**data**]

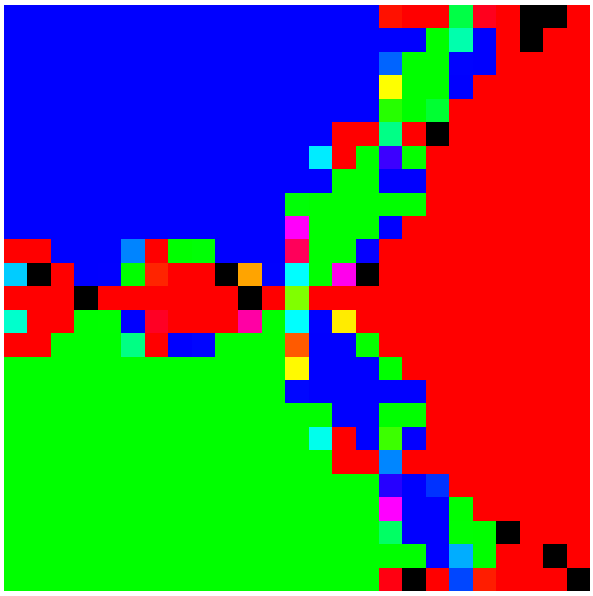
Out[51]//TableForm=

-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.866025 I	-0.5 + 0.8
1.00035 + 0.000787998 I	0.998578 + 0.000662602 I	-0.5 + 0.866025 I	-0.5 + 0.8
-4996.8 + 1067.5 I	1. + 1.20371 $\times 10^{-35}$ I	1. - 6.68949 $\times 10^{-8}$ I	-0.499641
1. + 0. I	1. + 0. I	1. + 0. I	1.00004 -
-4996.8 - 1067.5 I	1. - 1.20371 $\times 10^{-35}$ I	1. + 6.68949 $\times 10^{-8}$ I	-0.499641
1.00035 - 0.000787998 I	0.998578 - 0.000662602 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8
-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.866025 I	-0.5 - 0.8

This function will assign a Hue (which is a color directive) to each root. Then we can plot the values as an array of colored squares.

In[52]:= **RootColor**[**z_**] **:=** **Hue**[**Arg**[**z**] **/** (**2 Pi**)]

```
In[53]:= Show[Graphics[RasterArray[Map[RootColor, data, {2}]]],
          AspectRatio -> 1]
```

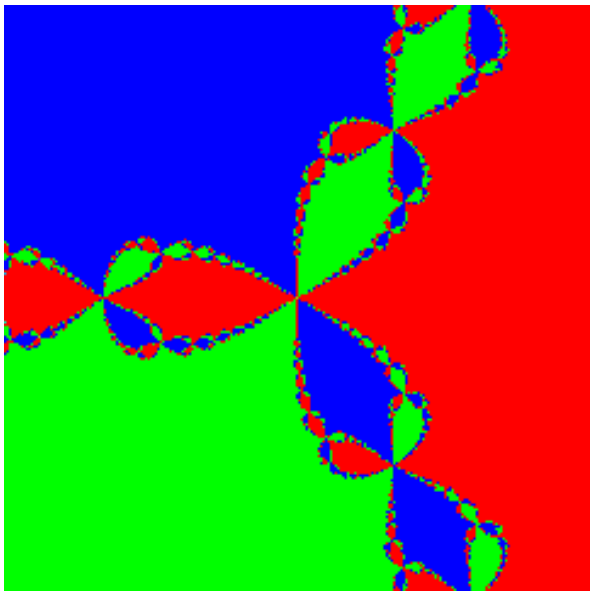


```
Out[53]= - Graphics -
```

Neat, huh? Let's get it a bit more detailed.

```
In[54]:= Timing[data = Table[Nh[x + I y, 30],
           {y, 1.2, -1.2, -0.01}, {x, -1.2, 1.2, 0.01}];]
Out[54]= {80.43 Second, Null}
```

```
In[55]:= Show[Graphics[RasterArray[Map[RootColor, data, {2}]]],
         AspectRatio -> 1]
```



```
Out[55]= - Graphics -
```

Now no one denies the intricacy and beauty of this picture. What does it mean? You can’t "see" the roots on the picture, but they lie in the middle of those three big colored regions. Each root has a neighborhood on which there is quick convergence to that root, but in between roots, there are areas where the convergence is actually to the *furthest* rather than the closest root!