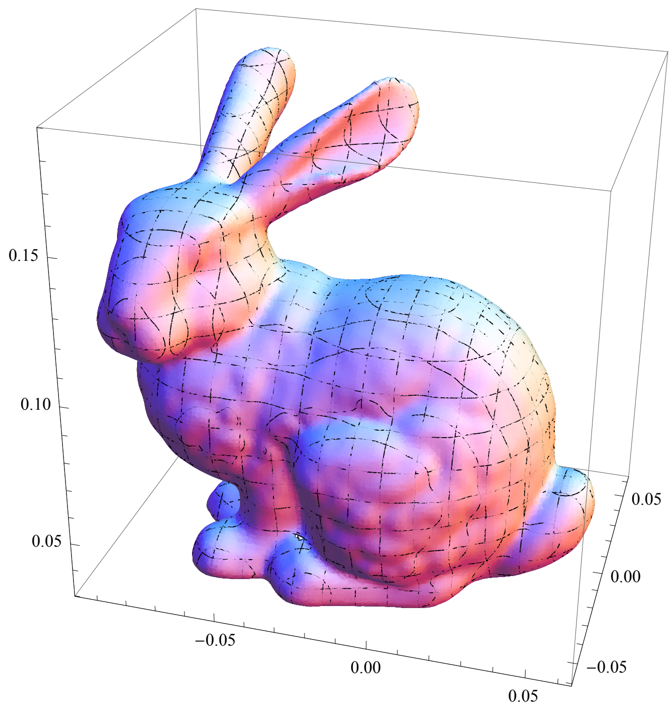
Geometry: A Primer

```
WordData["geometry", "Definitions"]
```

```
{{geometry, Noun} →
  the pure mathematics of points and lines and curves and surfaces}
```

In[27]:=
```
data := ExampleData[{"Geometry3D", "StanfordBunny"}, "VertexData"]
ListSurfacePlot3D[data, MaxPlotPoints → 50]
```




```
// FullForm
```

A very large output was generated. Here is a sample of it:

```
Graphics3D[GraphicsComplex[
  List[List[-0.093117181023564`, -0.027147756306162885`, 0.12184614394650314`],
   List[-0.09312986017687692`, -0.027107721476843864`, 0.12184614394650314`],
   List[-0.093117181023564`, -0.027107721476843864`, 0.12179324718031949`],
   List[-0.09274591522543137`, -0.027107721476843864`, 0.1202872186297118`],
   List[-0.09240862221806928`, -0.02832664368730603`, 0.12028721862971181`],
   List[-0.09275694964018771`, -0.02832664368730603`, 0.12184614394650314`],
   List[-0.09357975005565491`, -0.0258887992663817`, 0.12184614394650314`],
   List[-0.09322849895259303`, -0.0258887992663817`, 0.1202872186297118`],
   List[-0.093117181023564`, -0.02616789300756168`, 0.1202872186297118`],
   List[-0.093117181023564`, -0.02769924543496`, 0.1234050692632945`],
   List[-0.0932922859398417`, -0.027107721476843864`, 0.12340506926329449`],
   List[-0.09258606227604958`, -0.029545565897768196`, 0.12184614394650314`],
   List[-0.09233272619809216`, -0.029545565897768196`, 0.12028721862971178`],
```

```
    List[-0.09215714486918956`, -0.027107721476843864`, 0.11872829331292047`],
    List[-0.09191260754741531`, -0.02832664368730603`, 0.11872829331292048`],
    List[-0.09293490818520901`, -0.02832664368730603`, 0.12340506926329449`],
    List[-0.09398795567448988`, -0.024669877055919533`, 0.12184614394650314`],
    List[-0.09366297063373058`, -0.024669877055919533`, 0.120287218629711 8`],
    List[-0.093117181023564`, -0.0258887992663817`, 0.1200021322190643`],
    List[-0.0936973806557824`, -0.0258887992663817`, 0.12340506926329449`],
    List[-0.093117181023564`, -0.027748447549580516`, 0.12496399458008582`],
    \[LeftSkeleton]53053\[RightSkeleton],
    List[0.023771875798810897`, 0.008865528676568788`, 0.04260955650172277`],
    List[0.05413638747137687 6`, 0.001355005920382164`, 0.06185386772637497`],
    List[0.0214362164689735 9`, 0.016376051432755414`, 0.04260955650172277`],
    List[0.04778641955751008`, -0.013666039591991086`, 0.04260955650172277`],
    List[0.05795413450806899`, -0.013666039591991086`, 0.05223171211404887`],
    List[0.05969294244707377`, -0.006155516835804461`, 0.06185386772637497`],
    List[0.05230897307367704 5`, 0.001355005920382164`, 0.05223171211404887`],
    List[0.04064847522305373`, 0.001355005920382164`, 0.04260955650172277`],
    List[0.05688786396858605`, -0.006155516835804461`, 0.05223171211404887`],
    List[0.0467732807894438`, -0.006155516835804461`, 0.04260955650172277`],
    List[-0.08485448485482829`, -0.028687085104364338`, 0.09072033456335327`],
    List[-0.07534689117441401`, -0.028687085104364338`, 0.07147602333870107`],
    List[-0.04647580894240326`, -0.04370813061673759`, 0.09072033456335327`],
    List[-0.07501419440104849`, 0.016376051432755414`, 0.14845326823730987`],
    List[-0.046589409988136835`, -0.04370813061673759`, 0.04260955650172277`],
    List[-0.0655962922691146`, 0.03139709694512866`, 0.16769757946196207`],
    List[-0.025519421509406025`, 0.02388657418894204`, 0.10034249017567937`],
    List[-0.026375867648671326`, 0.008865528676568788`, 0.17731973507428817`],
    List[0.02217882644935429`, 0.02388657418894204`, 0.09072033456335327`],
    List[0.03262836341512129`, -0.028687085104364338`, 0.04260955650172277`]],
   List[List[\[LeftSkeleton]1\[RightSkeleton]],
    List[\[LeftSkeleton]1\[RightSkeleton]]],
   Rule[VertexNormals, List[\[LeftSkeleton]1\[RightSkeleton]]]],
  \[LeftSkeleton]8\[RightSkeleton]]
```

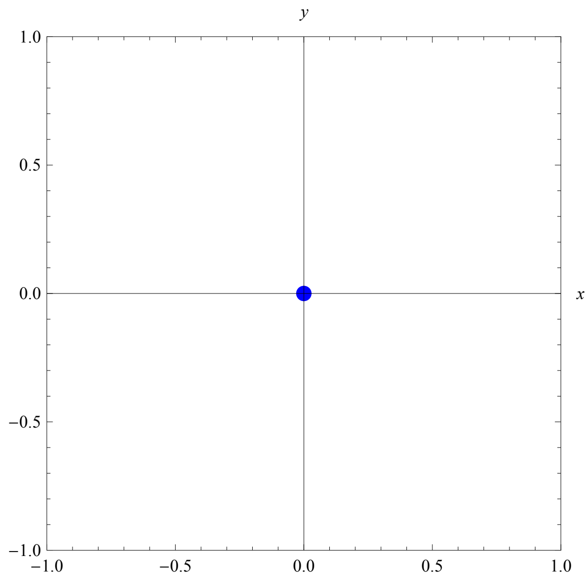
Show Less | Show More | Show Full Output | Set Size Limit...

Point of Origin: Exit the Void

In[15]:=
```
point[coords_, size_: .03, plotRange_: 1, color_: Blue] :=
 Graphics[{PointSize[size], color, Point[coords]}, Frame → True,
  Axes → True, PlotRange → plotRange, AxesLabel → {x, y}, ImageSize → 300]
```

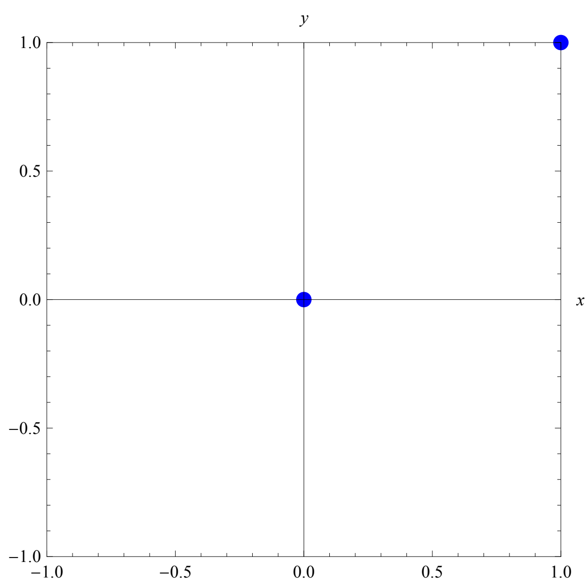
In[16]:=
```
point[{0, 0}]
```

Out[16]=



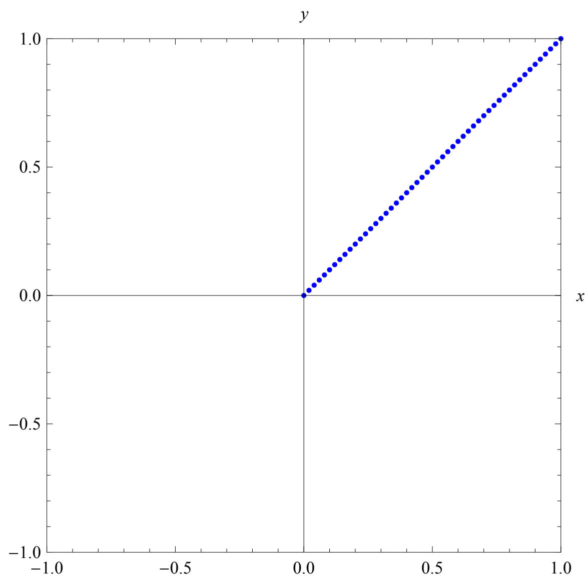
In[17]:=
```
point[{{0, 0}, {1, 1}}]
```

Out[17]=



In[18]:=
```
point[Table[{t, t}, {t, 0, 1, 1 / 50}], .01]
```

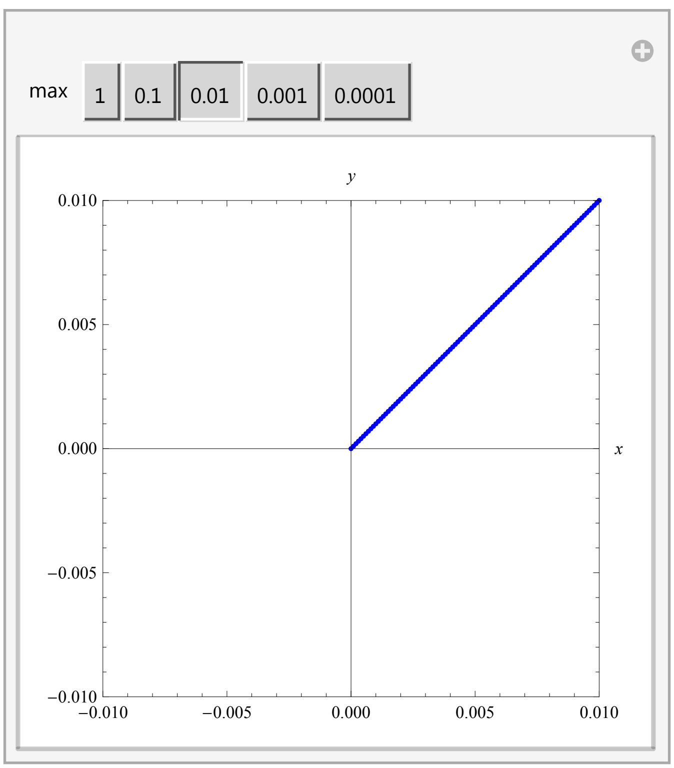
Out[18]=



Is this a line?

In[19]:=
```
tenthousandPoints[max_] := point[Table[{t, t}, {t, 0, max, 1 / 10 000}], .01, max]
Manipulate[tenthousandPoints[max], {max, {1, .1, .01, .001, .0001}}]
```

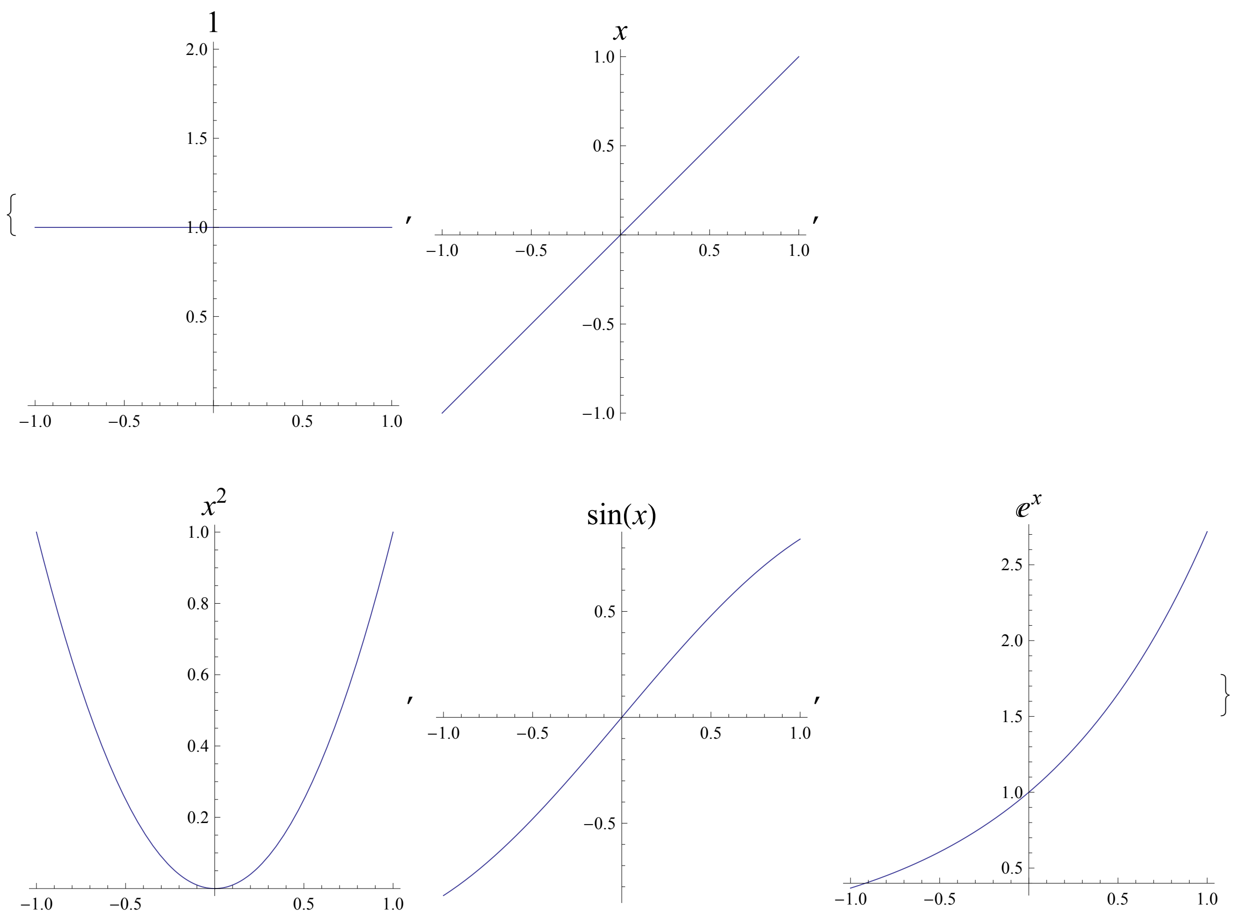
Out[20]=



The Shortest Distance Between Two Points

In[21]:=
```
Plot[#, {x, -1, 1}, AspectRatio → 1, ImageSize → 200, PlotLabel → Style[#, 16]] & /@
  {1, x, x^2, Sin[x], E^x}
```

Out[21]=

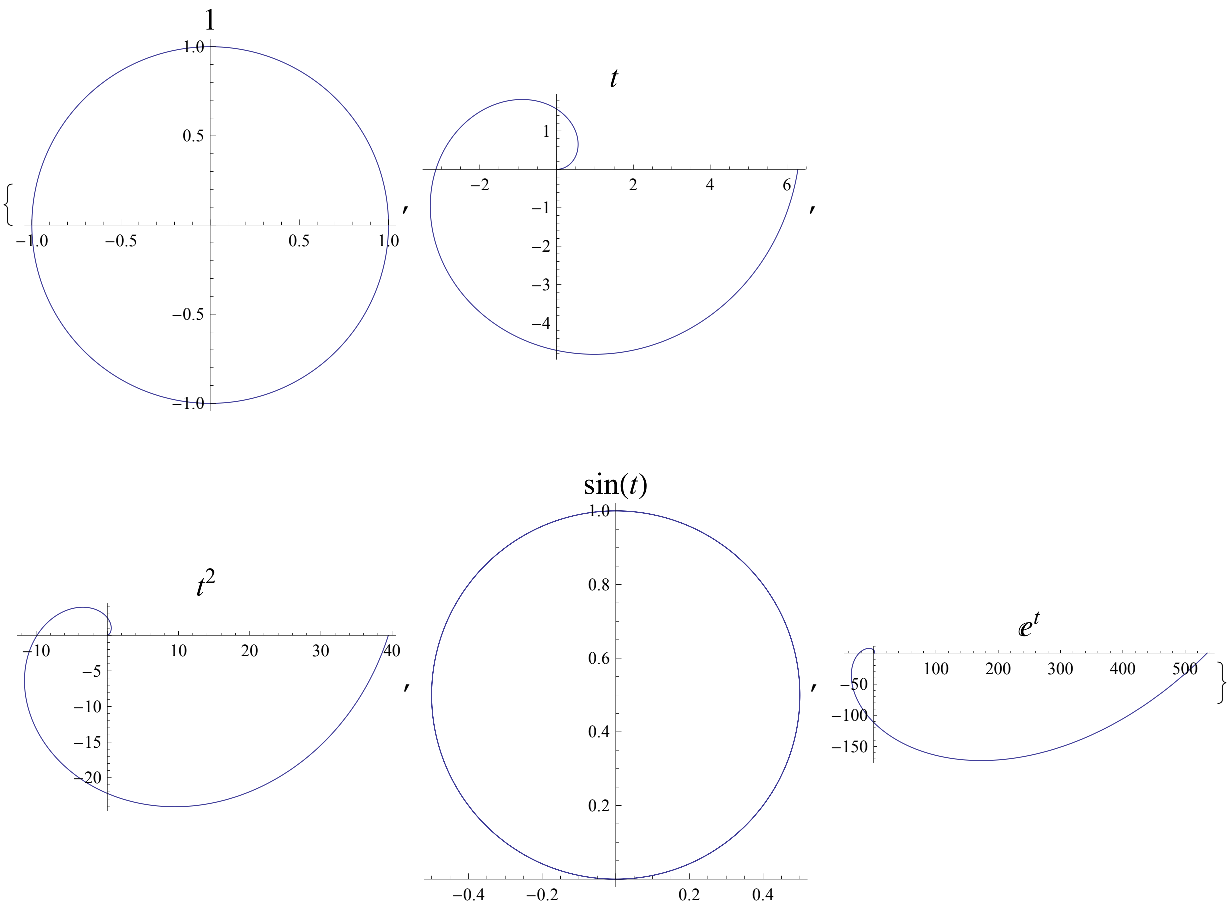


How ‘bout outside linear coordinates?

Polar

In[22]:=
```
PolarPlot[#, {t, 0, 2 π}, AspectRatio → Automatic,
   ImageSize → 200, PlotLabel → Style[#, 16]] & /@ {1, t, t^2, Sin[t], E^t}
```

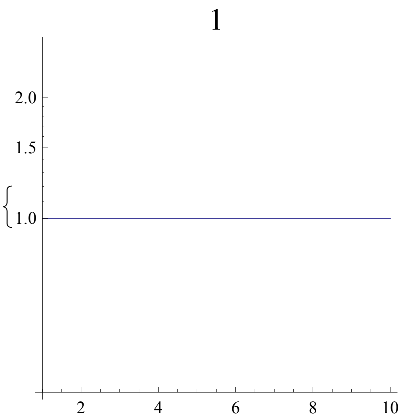
Out[22]=



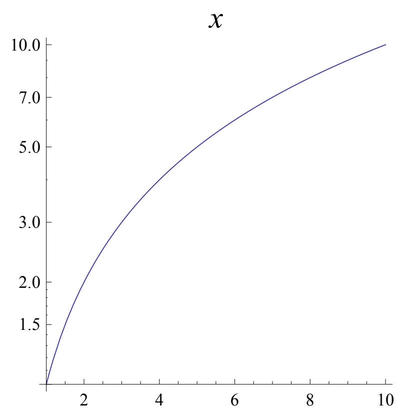
Linear X; Logarithmic Y

In[23]:=
```
LogPlot[#, {x, 1, 10}, AspectRatio → 1, ImageSize → 200, PlotLabel → Style[#, 16]] & /@
  {1, x, x^2, Sin[x], E^x}
```

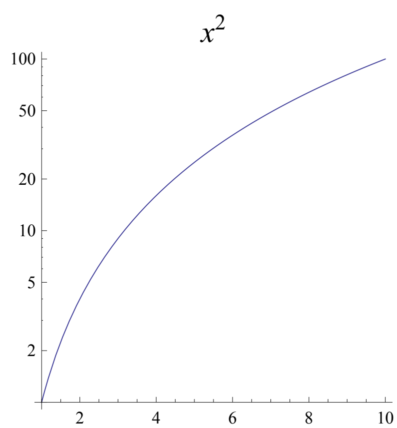
Out[23]= {



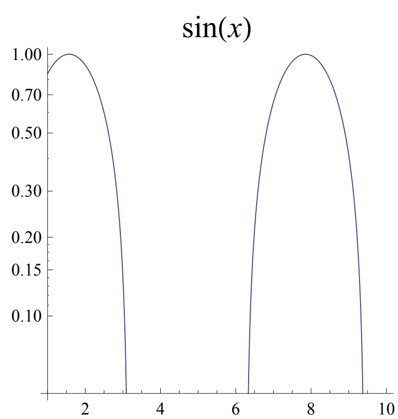
,



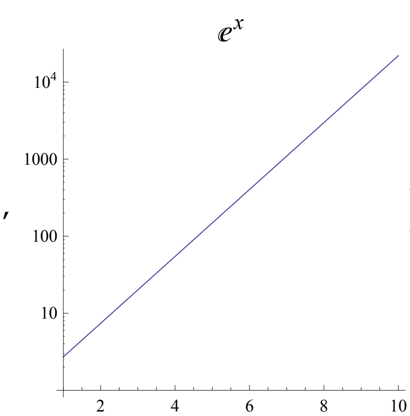
,



,



,

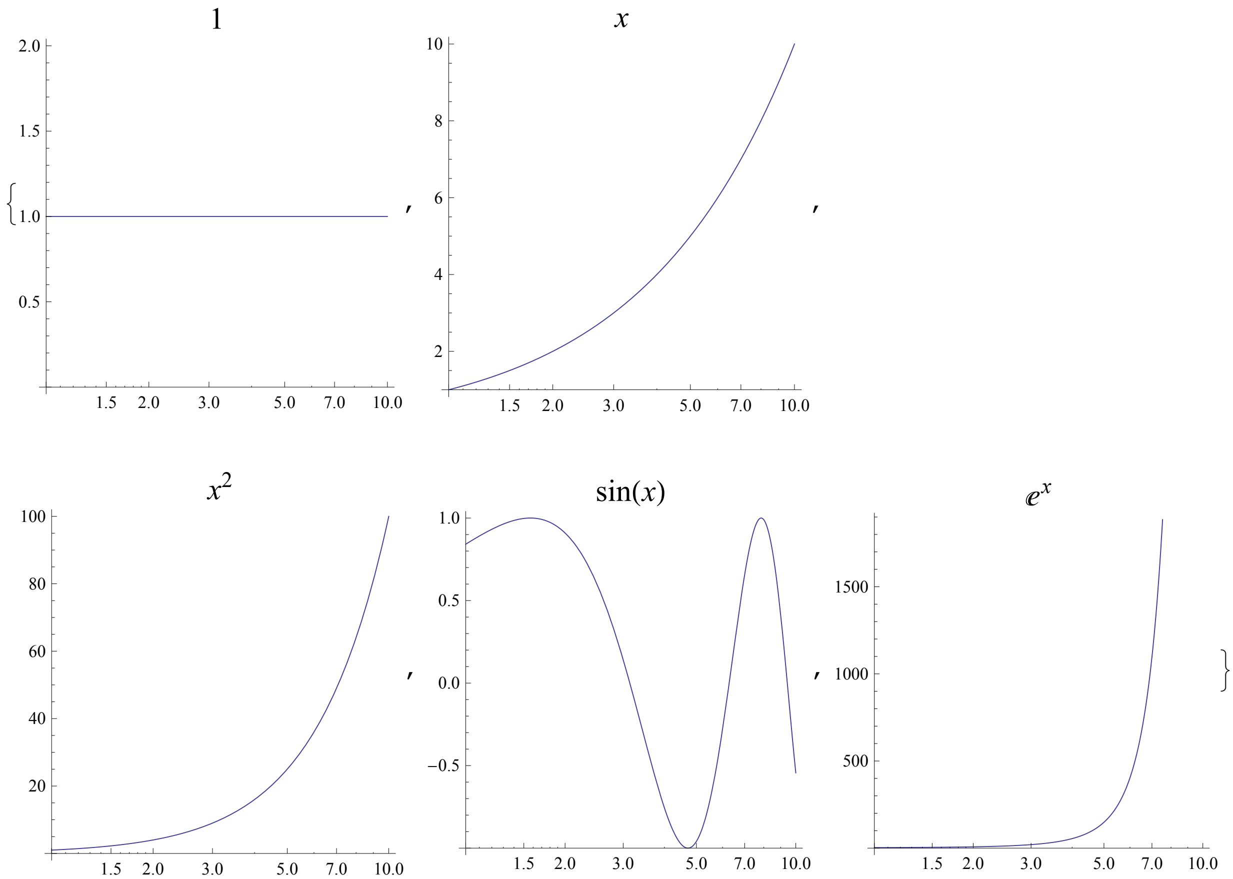


}

Logarithmic X; Linear Y

In[24]:=
```
LogLinearPlot[#, {x, 1, 10}, AspectRatio → 1,
  ImageSize → 200, PlotLabel → Style[#, 16]] & /@ {1, x, x^2, Sin[x], E^x}
```

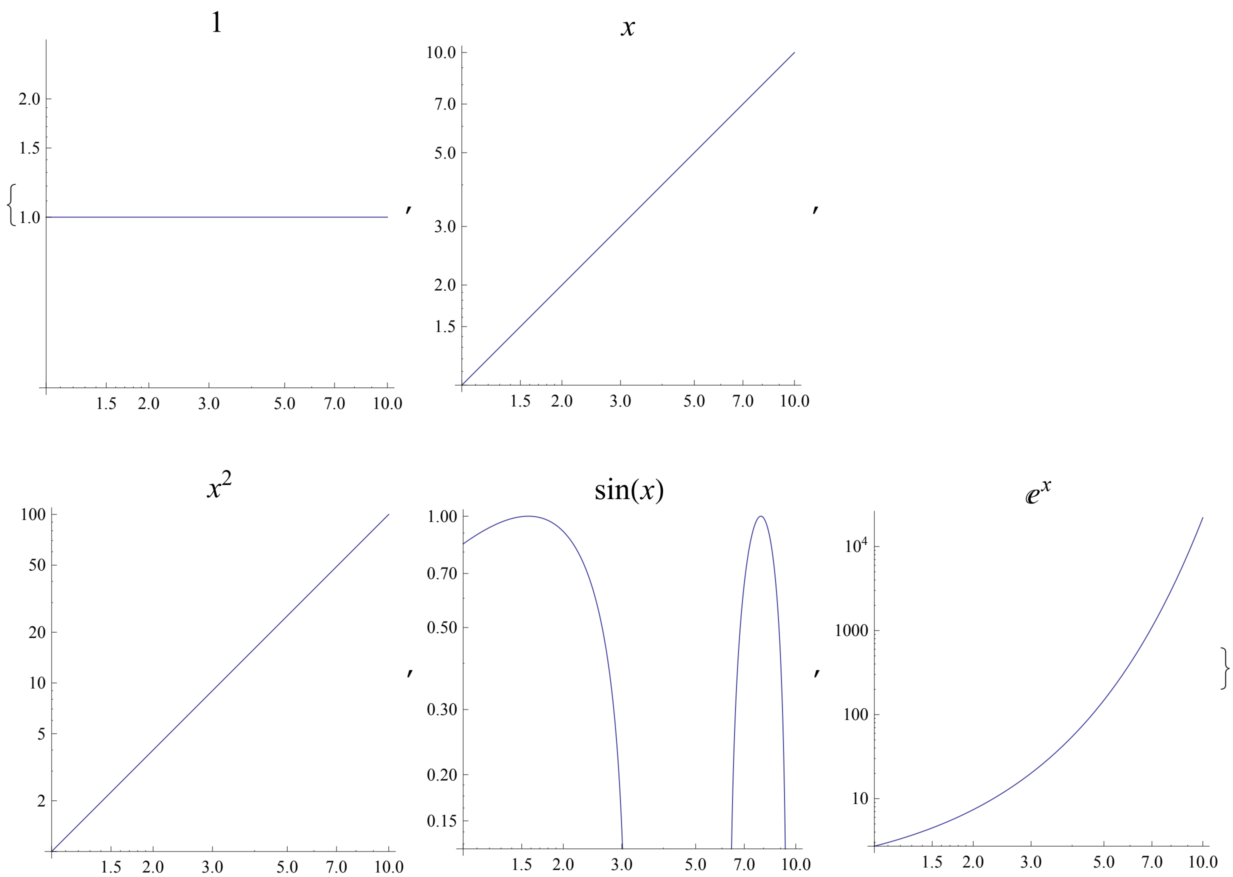
Out[24]=



Logarithmic X; Logarithmic Y

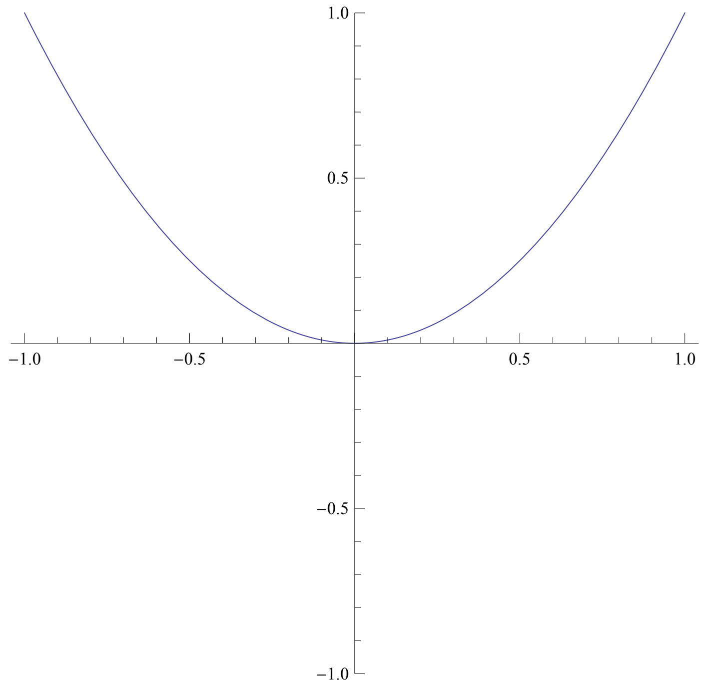
In[25]:=
```
LogLogPlot[#, {x, 1, 10}, AspectRatio → 1,
   ImageSize → 200, PlotLabel → Style[#, 16]] & /@ {1, x, x^2, Sin[x], E^x}
```

Out[25]=

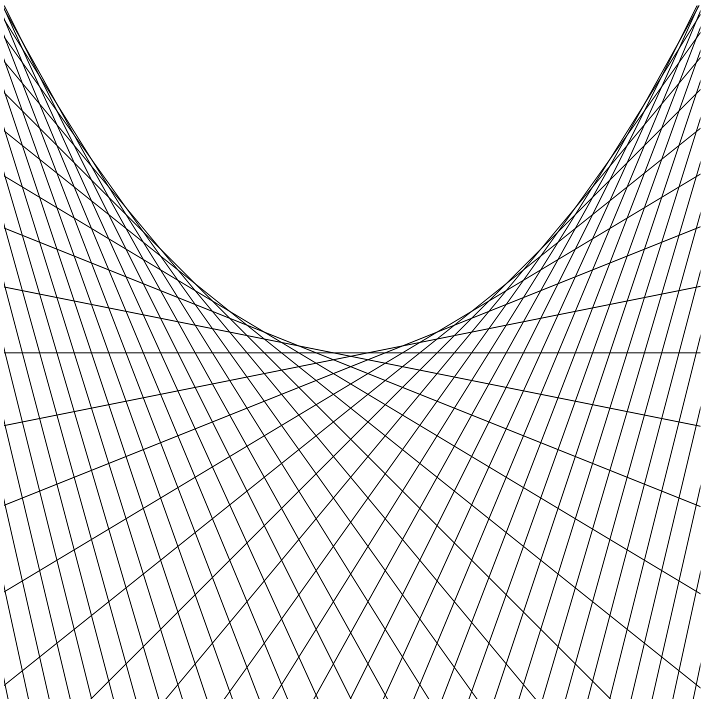


Quadratic Equation

```
Plot[x^2, {x, -1, 1}, PlotRange → 1, AspectRatio → Automatic]
```



```
f[x_] := x^2
p = Table[{{a - 2, f'[a] ((a - 2) - a) + f[a]},
    {a + 2, f'[a] ((a + 2) - a) + f[a]}}, {a, -10, 10, .1}];
Graphics[Line[p], PlotRange → {{-1, 1}, {-1, 1}}]
```

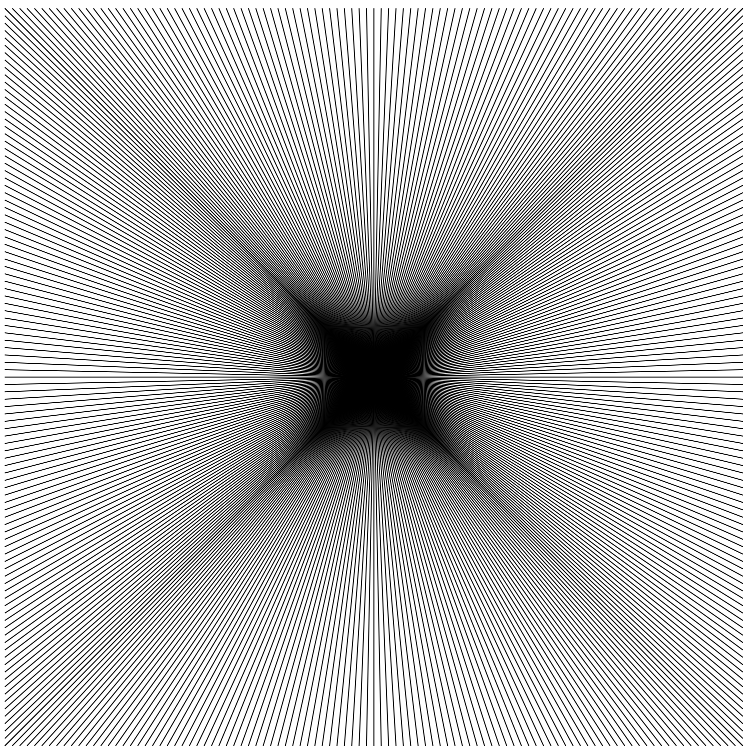


Moiré pattern — Would you believe these are all cartesian lines?

In[14]:=
```
Manipulate[
 Graphics[Table[Line[{{{x, 0}, {1 - x, 1}}, {{0, x}, {1, 1 - x}}}], {x, 0, 1, density}],
  ImageSize → 400],
 {{density, .01}, {1, .5, .25, .1, .05, .025, .01, .005, .0025, .001}}]
```

Out[14]=

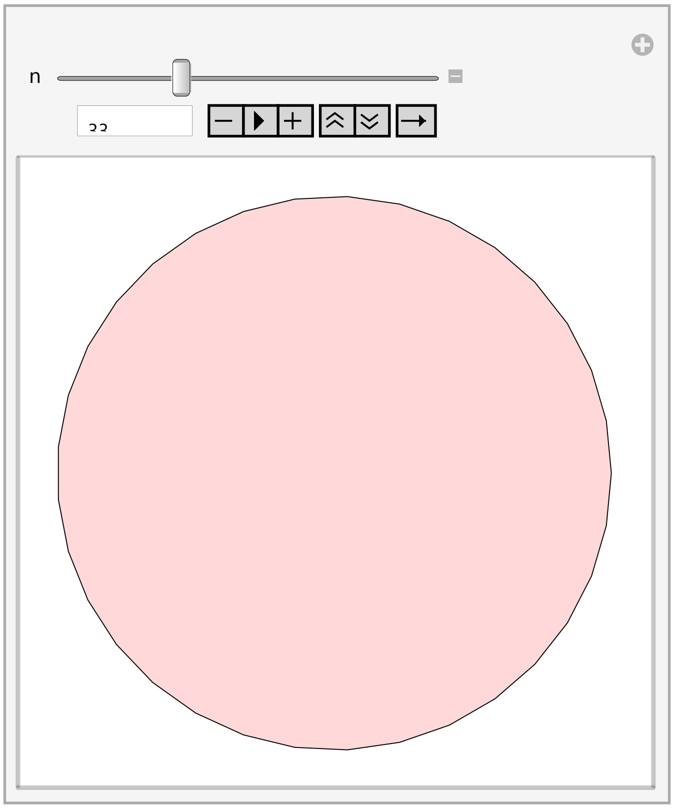
density 0.01



Curves

A circle, or just a high fidelity n-gon? Are they different!?

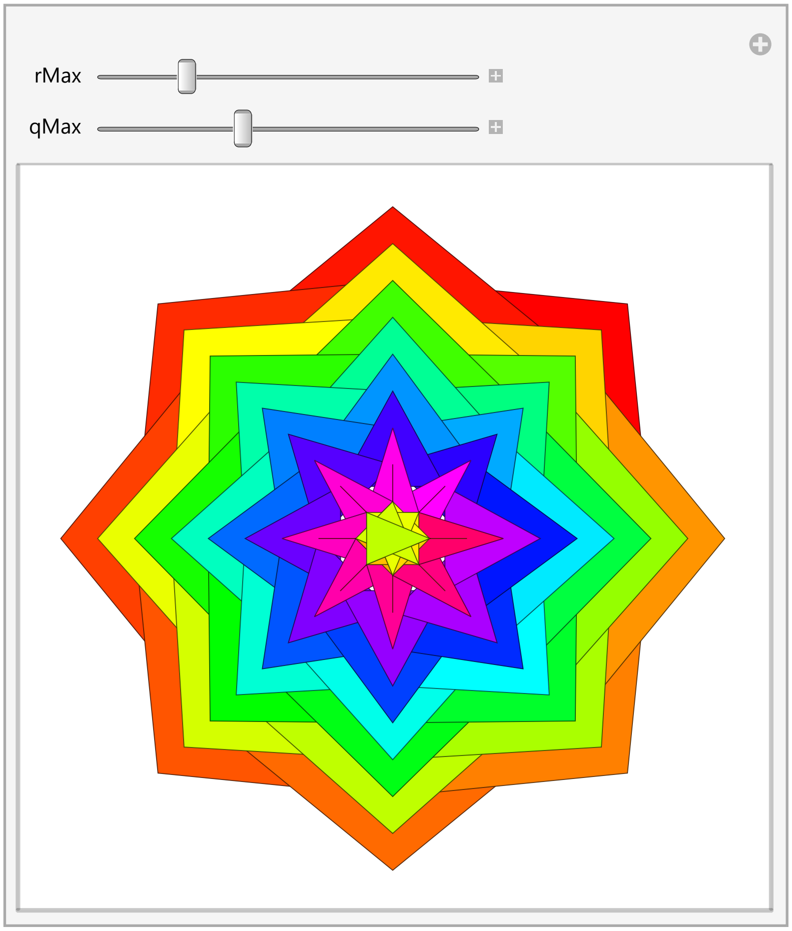
```
Manipulate[Graphics[{EdgeForm[Black], LightRed,
    Polygon[Table[{Cos[2 Pi k / n], Sin[2 Pi k / n]}, {k, n}]]},
  ImageSize → 300], {{n, 3}, 3, 100, 1}]
```



Do you polygon?

This looks scary, but focus on the English! We’re manipulating a graphic consisting of a table of colored polygons which are comprised of a bunch of sin+cos values. With rMax starting at 6 and qMax starting at 8, can you figure out what these values do?

```
With[{d = 2 Pi / qMax},
 Manipulate[Graphics[Table[{EdgeForm[Opacity[.6]], Hue[(-11 + q + 10 r) / (rMax qMax)],
      Polygon[{(8 - r) {Cos[d (q - 1)], Sin[d (q - 1)]},
        (8 - r) {Cos[d (q + 1)], Sin[d (q + 1)]}, (10 - r) {Cos[d q], Sin[d q]}}]},
    {r, rMax}, {q, qMax}]], {{rMax, 6}, 1, 40, 1}, {{qMax, 8}, 1, 20, 1}]]
```

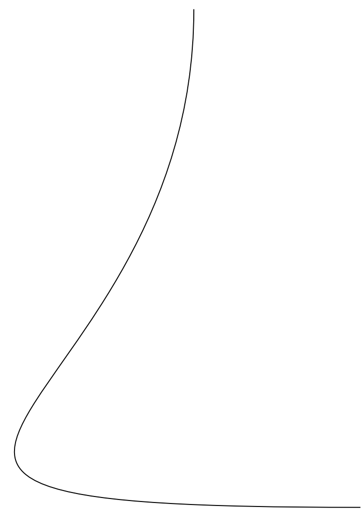


Sexy+Lazy Application: The Bézier Curve

Demonstration by example; here’s an arbitrary curve

In[8]:=
```
start := {{2, 2}, {2, -3}, {-4, -4}, {4, -4}}
Graphics[{BezierCurve[start]}]
```

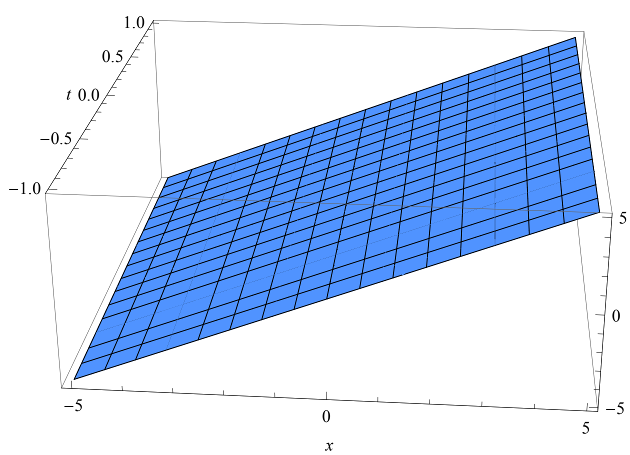
Out[9]=



Now let’s write a tweening function—this just transitions linearly over time. In this case, t is arbitrary

In[7]:=
```
tween[p1_, p2_, t_] := (1 - t) p1 + t p2

Plot3D[(1 - t) x + t x, {x, -5, 5}, {t, -1, 1}, AxesLabel → Automatic, ImageSize → 600]
```



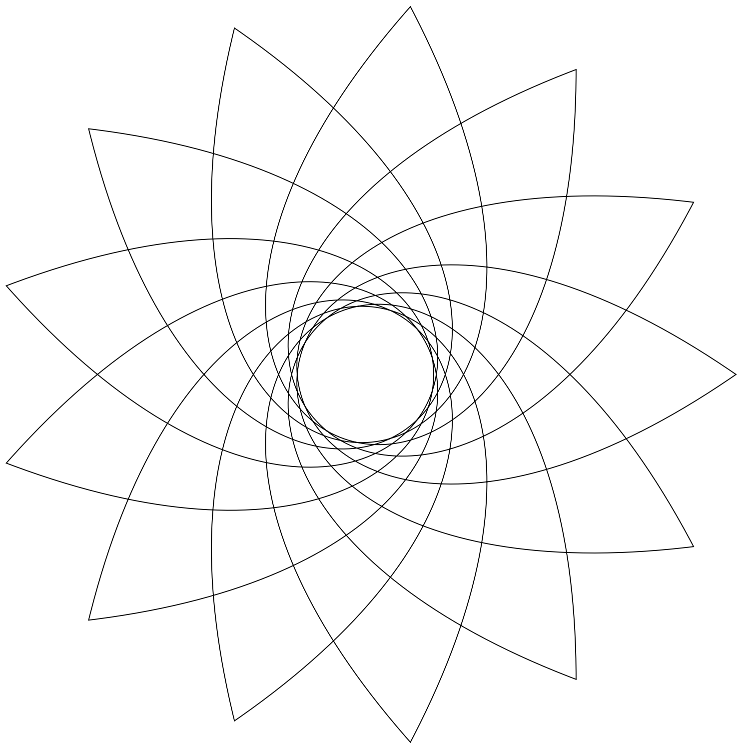
Using our tweening function and the starting curve

In[6]:=
```
bezier[c1_, c2_, c3_, c4_, t_] :=
 Graphics[{Thick, BezierCurve[{c1, c2, c3, c4}], Green,
   Line[{c1, c2, c3, c4}], Red, Point[{c1, c2, c3, c4}], Thick,
   BezierCurve[start], Green, Line[start], Red, Point[start],
   Thickness[.01], Purple, BezierCurve[tween[start, {c1, c2, c3, c4}, t]]}]
```

```
Manipulate[
 Animate[bezier[c1, c2, c3, c4, t], {t, 0, 1}, AnimationDirection → ForwardBackward],
 Row[{Control[{c1, {-4, -4}, {4, 4}}], Control[{c2, {-4, -4}, {4, 4}}], Control[
    {c3, {-4, -4}, {4, 4}}], Control[{c4, {-4, -4}, {4, 4}}]}], AspectRatio → 1]
```

Bézier Flower

```
Graphics[BezierCurve[Table[{Cos[2 k Pi / 13], Sin[2 k Pi / 13]}, {k, 0, 156, 4}]],
 ImageSize → 600]
```

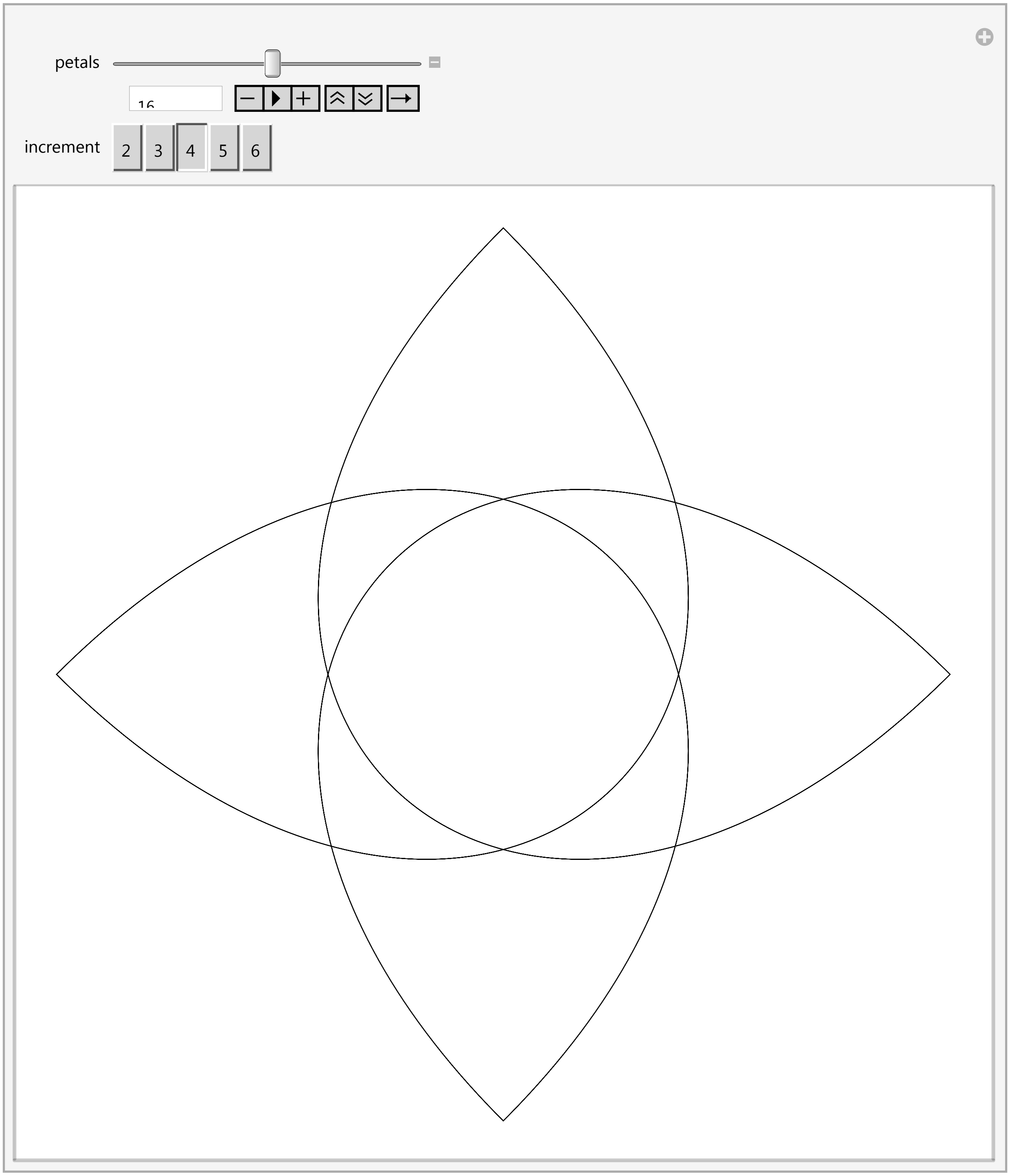


Hmm...the flower has 13 petals, and the divisor inside Cos and Sin is 13; additionally, the max limit of k is 156, incrementing by 4.

156/13 = 12 = 4*3

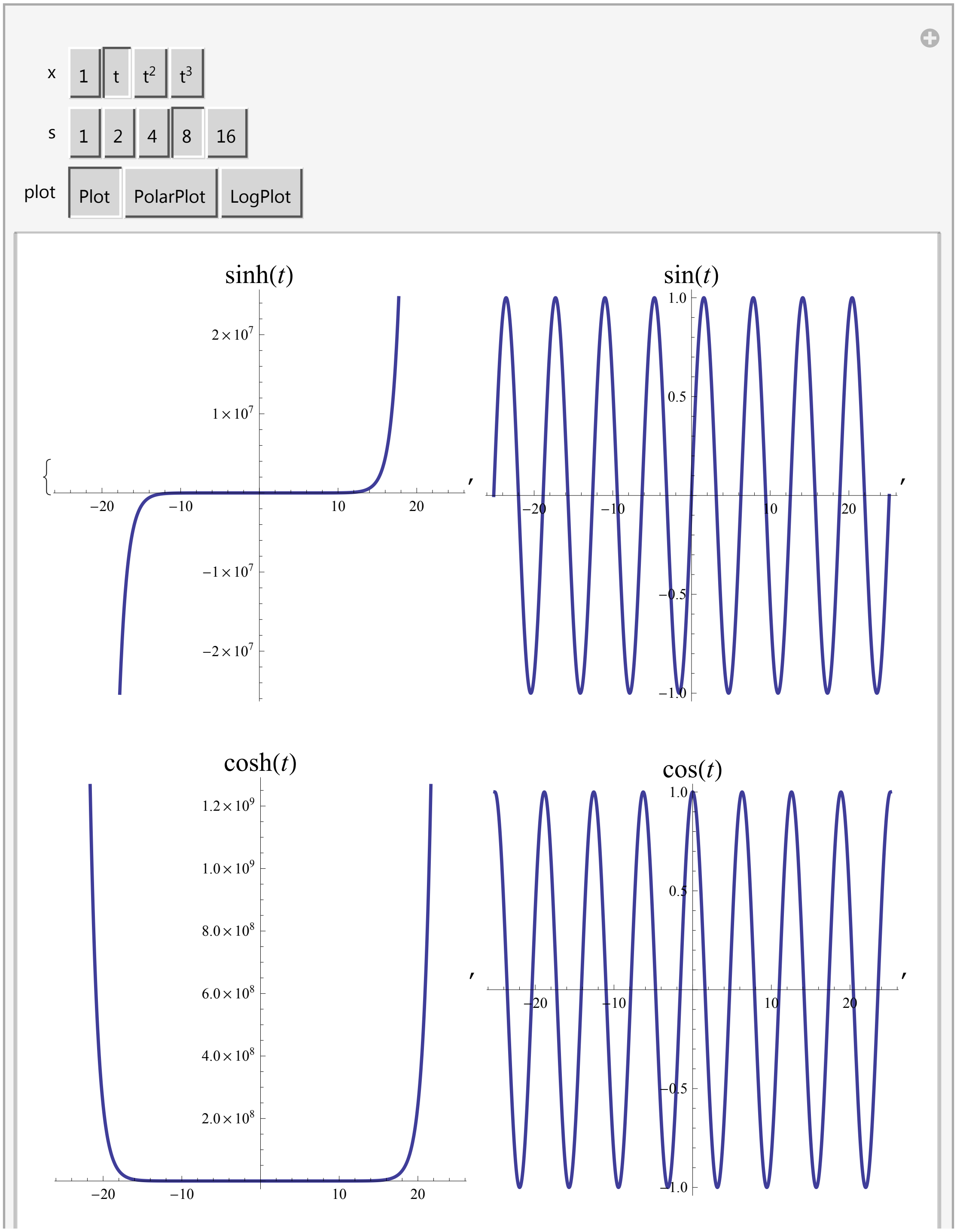
Any intuitions here?

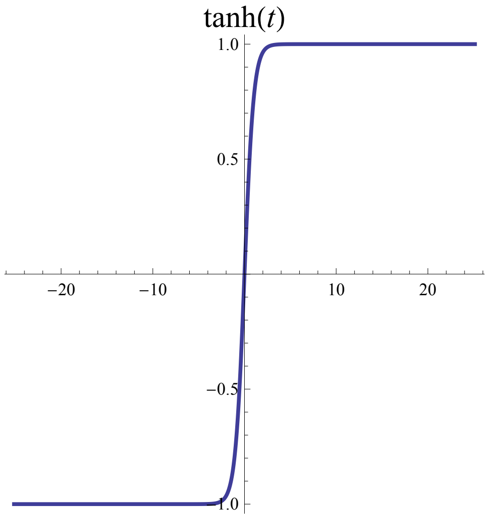
```
Manipulate[Graphics[BezierCurve[Table[{Cos[2 k Pi / petals], Sin[2 k Pi / petals]},
    {k, 0, 3 * increment * petals, increment}]], ImageSize → 600],
 {{petals, 13}, 3, 28, 1}, {{increment, 4}, {2, 3, 4, 5, 6}}]
```


petals
16
increment
2
3
4
5
6

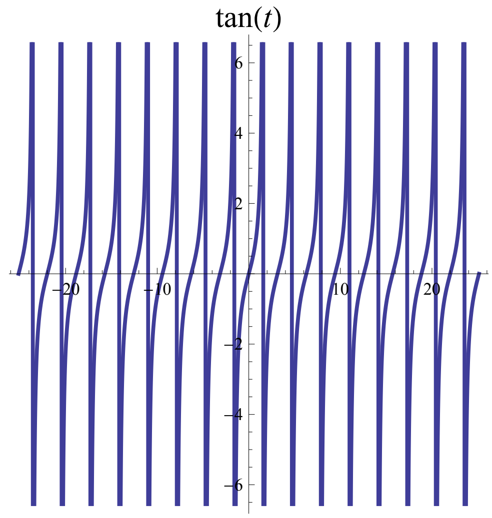
Hyperbolic Spaces | Contour Surfaces

```
Manipulate[
 plot[#, {t, -π s, π s}, PlotLabel → Style[#, 16], PlotStyle → Thick, AspectRatio → 1,
    ImageSize → 250, PlotRange → Automatic] & /@ {Sinh[x], Sin[x], Cosh[x],
   Cos[x], Tanh[x], Tan[x], Csch[x], Csc[x], Sech[x], Sec[x], Coth[x], Cot[x]},
 {{x, t}, {1, t, t^2, t^3}}, {s, {1, 2, 4, 8, 16}}, {plot, {Plot, PolarPlot, LogPlot}}]
```

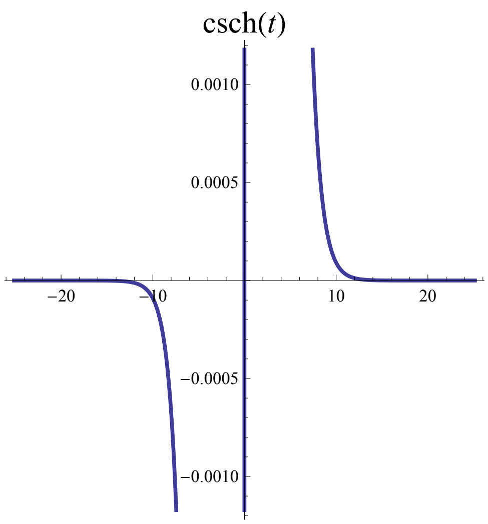



tanh(t)
1.0
0.5
−0.5
−1.0
−20
−10
10
20

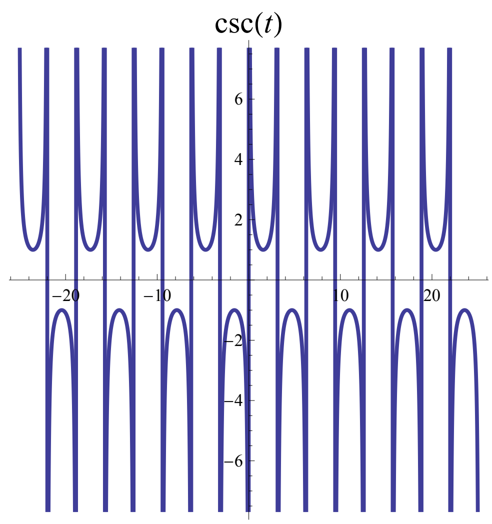
,


tan(t)
6
4
2
−2
−4
−6
−20
−10
10
20

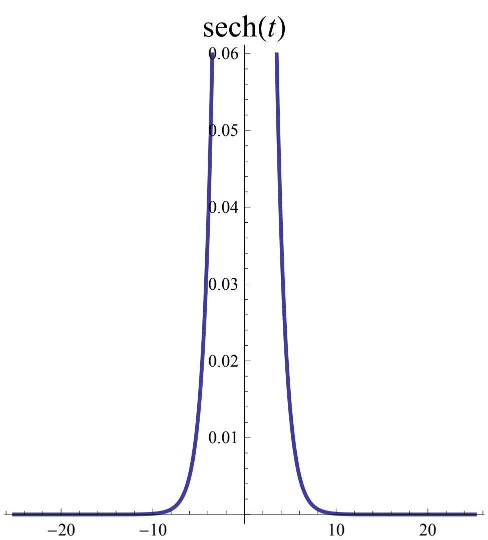
,


csch(t)
0.0010
0.0005
−0.0005
−0.0010
−20
−10
10
20

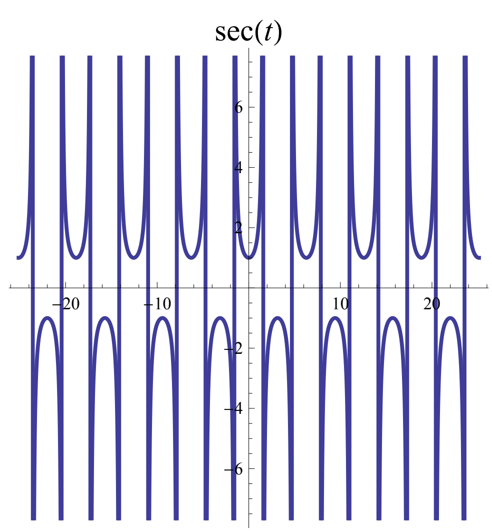
,


csc(t)
6
4
2
−2
−4
−6
−20
−10
10
20

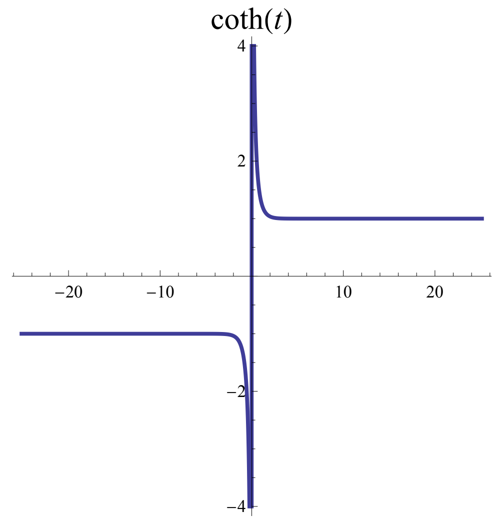
,


sech(t)
0.06
0.05
0.04
0.03
0.02
0.01
−20
−10
10
20

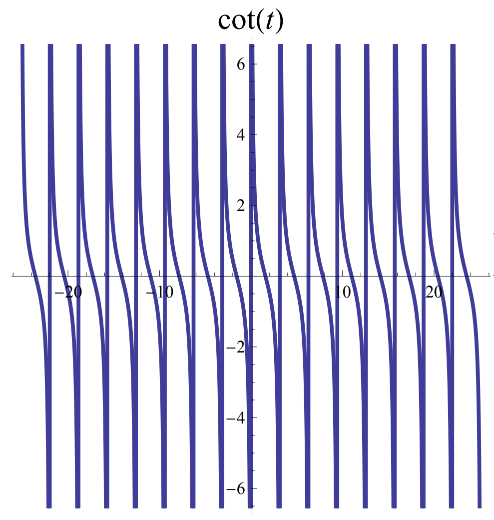
,


sec(t)
6
4
2
−2
−4
−6
−20
−10
10
20

,



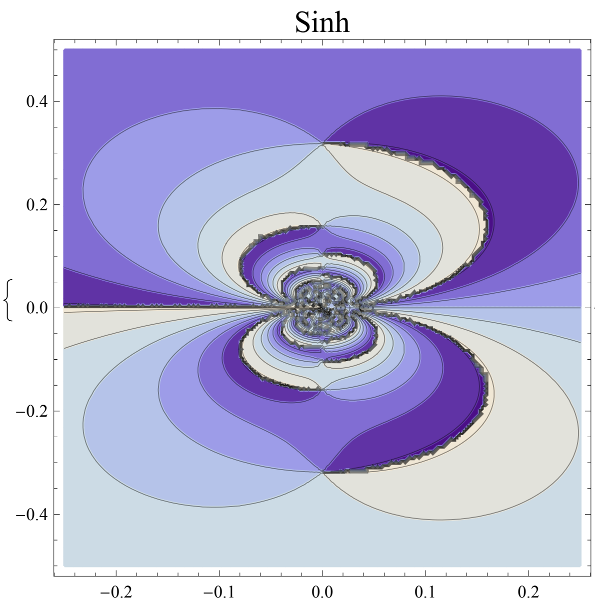
,



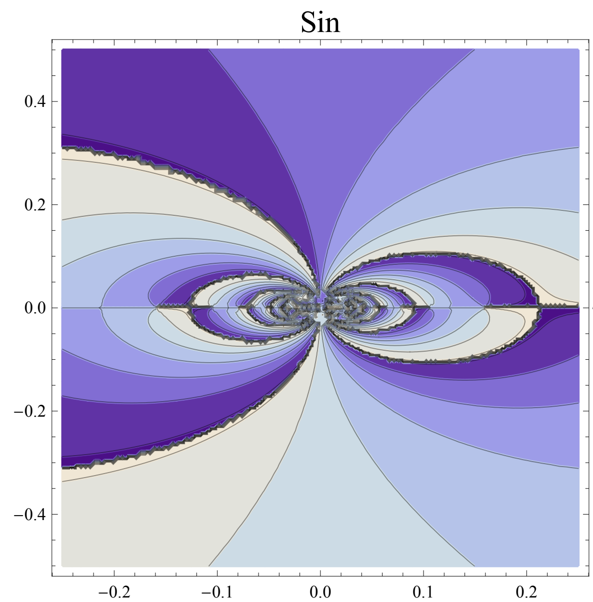
}

In[12]:=
```
ContourPlot[Arg[#[1 / (x + I y)]], {x, -1 / 4, 1 / 4}, {y, -1 / 2, 1 / 2}, Exclusions -> {},
    PlotRange -> All, ImageSize → 300, PlotLabel → Style[#, 16]] & /@
  {Sinh, Sin, Cosh, Cos, Tanh, Tan, Csch, Csc, Sech, Sec, Coth, Cot}
```

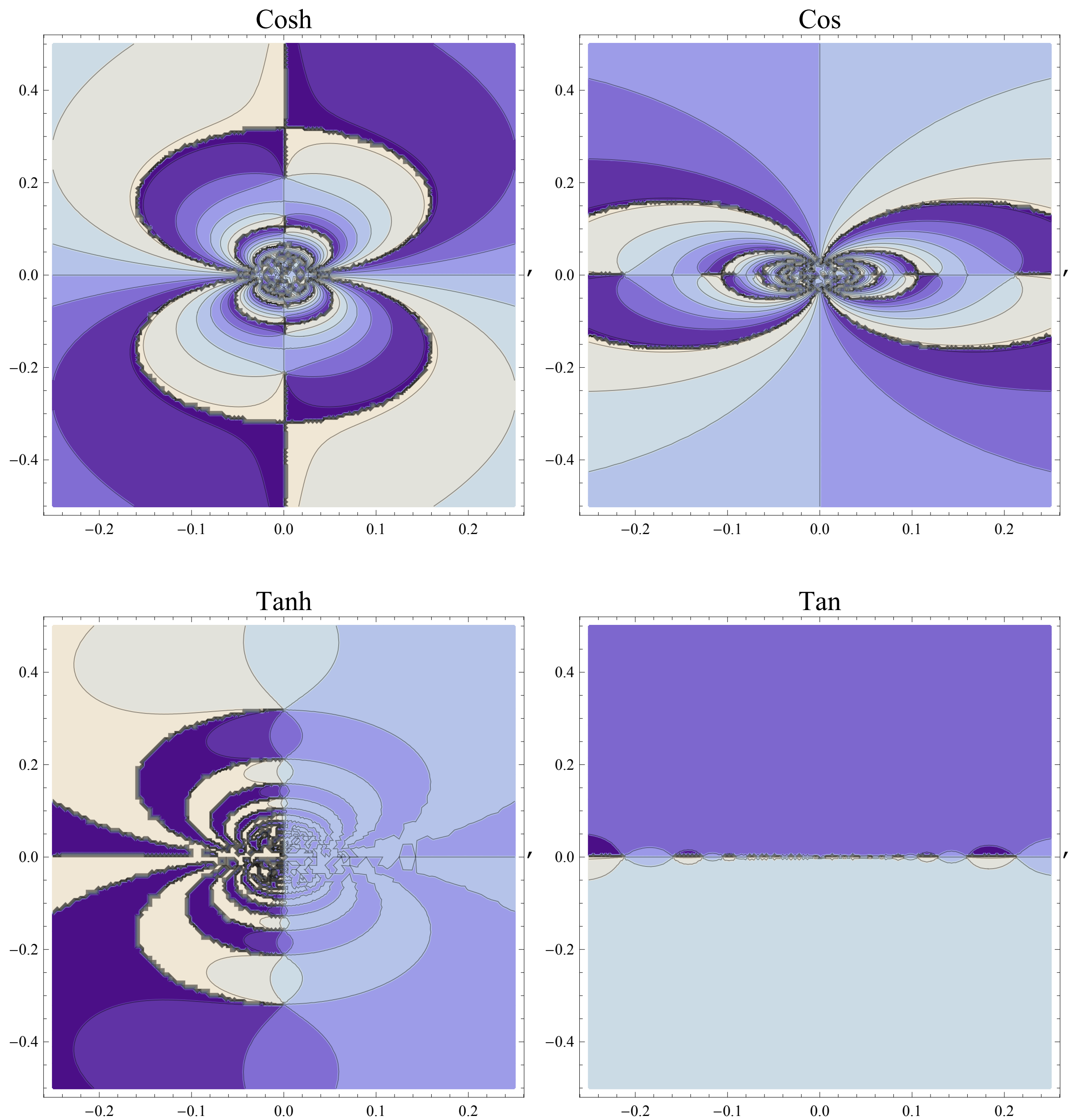
Out[12]= {

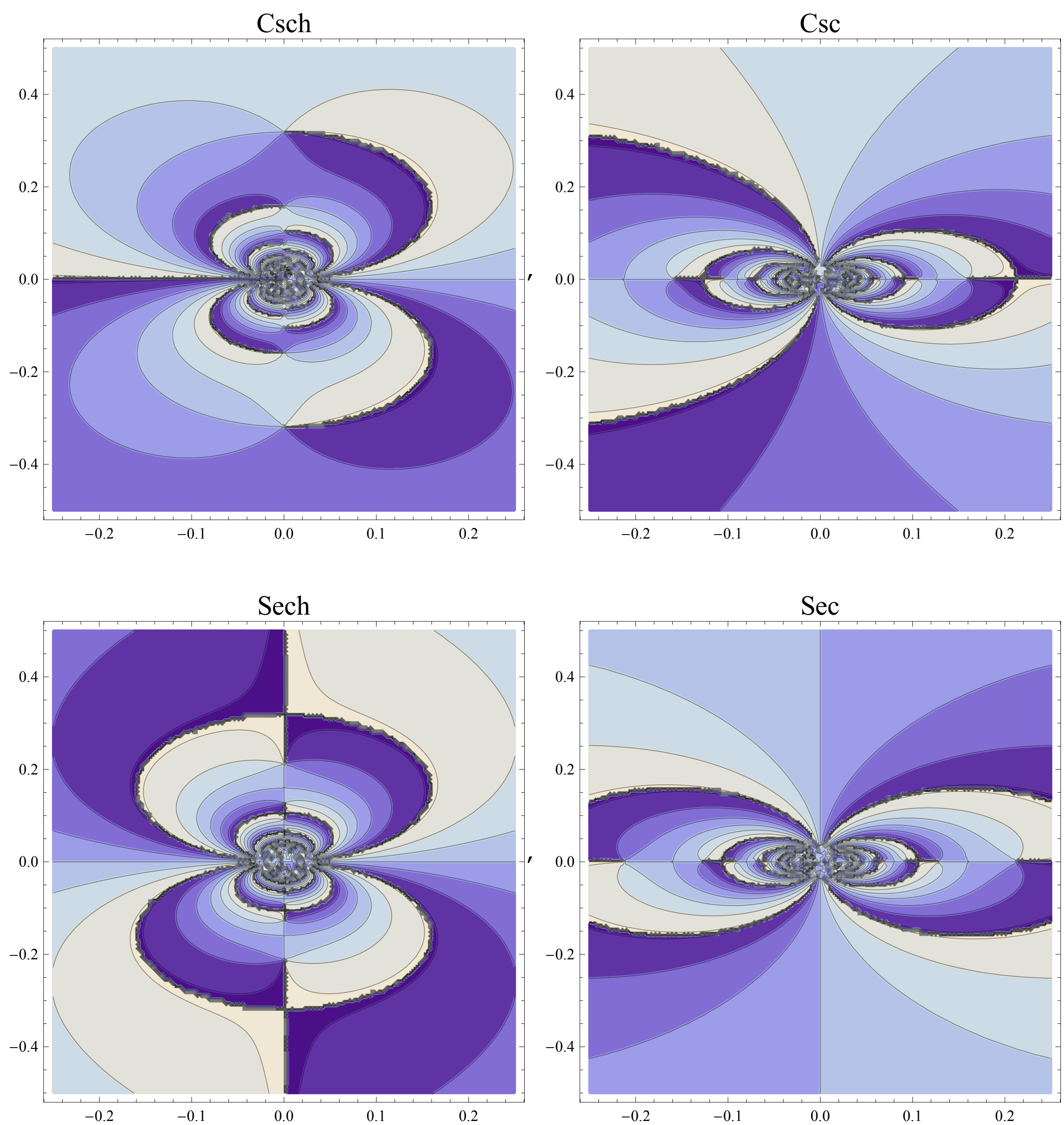


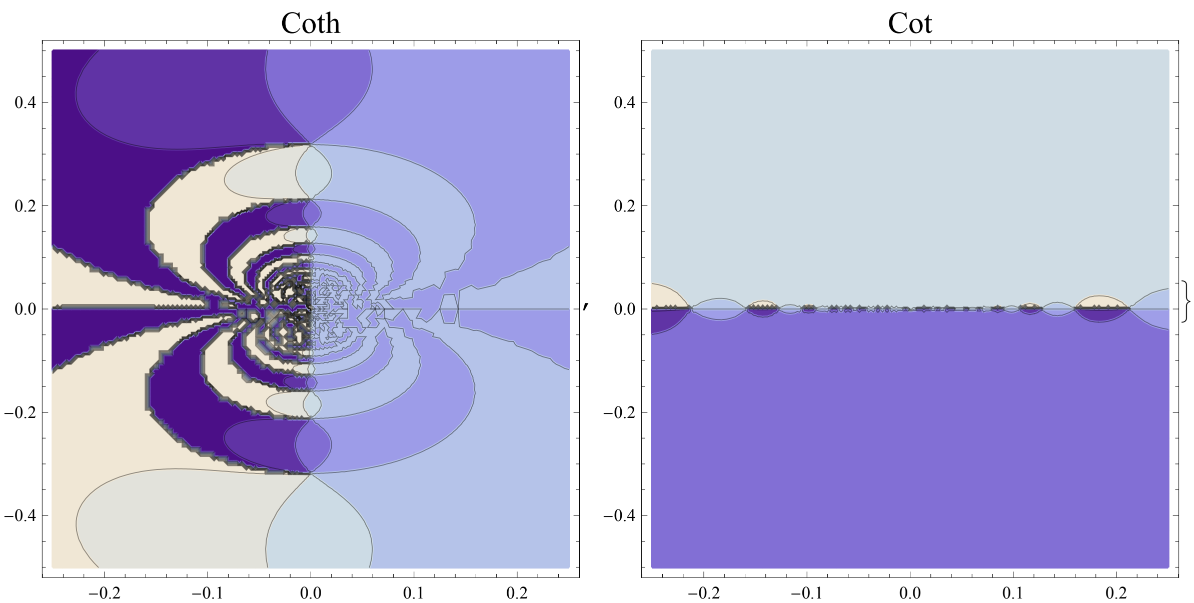
,



,


Cosh
Cos
Tanh
Tan
0.4
0.2
0.0
−0.2
−0.4
−0.2
−0.1
0.0
0.1
0.2


Csch
Csc
Sech
Sec
0.4
0.2
0.0
−0.2
−0.4
−0.2
−0.1
0.0
0.1
0.2


Coth
Cot
0.4
0.2
0.0
−0.2
−0.4
−0.2
−0.1
0.0
0.1
0.2