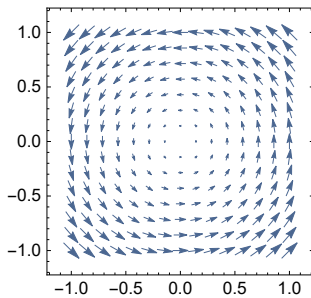


Pensieve header: Drawing phase portraits and drawing the Willerton Fish on April 4, 2016.

From <http://www.math.toronto.edu/drorbn/classes/16-1750-ShamelessMathematica/About.html>: **Possible Topics** (in no particular order). Whatever you may suggest, and the ~~Fibonacci numbers; the Jones polynomial; a more efficient Jones algorithm; a riddle on spheres; Khovanov homology; Γ -calculus; the Hopf fibration; Hilbert's 13th problem; non-commutative Gaussian elimination; free Lie algebras; the Baker-Campbell-Hausdorff formula; wacky numbers; an order 4 torus; the Schwarz Lantern; knot colourings; the Temperley-Lieb pairing; the dodecahedral link; sound experiments; barycentric subdivisions; a Peano curve; braid closures and Vogel's algorithm; the insolubility of the quintic; phase portraits; the Mandelbrot set; shadows of the Cantor Aerogel; quilt plots; some image transformations; De Bruijn graphs; the Riemann series theorem; finite type invariants and the Willerton fish.~~

Linear Phase Portraits

```
VectorPlot[{-y, x}, {x, -1, 1}, {y, -1, 1}]
```



$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}^T$$

```
{{1, 2}, {3, 4}}
```

```
?? ParametricPlot
```

`ParametricPlot[{fx, fy}, {u, umin, umax}]` generates a parametric plot of a curve with x and y coordinates f_x and f_y as a function of u .

`ParametricPlot[{fx, fy}, {gx, gy}, ..., {u, umin, umax}]` plots several parametric curves.

`ParametricPlot[{fx, fy}, {u, umin, umax}, {v, vmin, vmax}]` plots a parametric region.

`ParametricPlot[{fx, fy}, {gx, gy}, ..., {u, umin, umax}, {v, vmin, vmax}]` plots several parametric regions.

`ParametricPlot[... , {u, v} ∈ reg]` takes parameters $\{u, v\}$ to be in the geometric region reg . \gg

```
Attributes[ParametricPlot] = {HoldAll, Protected, ReadProtected}
```

```
Options[ParametricPlot] =
```

```
{AlignmentPoint → Center, AspectRatio → Automatic, Axes → True, AxesLabel → None,
  AxesOrigin → Automatic, AxesStyle → {}, Background → None, BaselinePosition → Automatic,
  BaseStyle → {}, BoundaryStyle → Automatic, ColorFunction → Automatic,
  ColorFunctionScaling → True, ColorOutput → Automatic, ContentSelectable → Automatic,
  CoordinatesToolOptions → Automatic, DisplayFunction → $DisplayFunction,
  Epilog → {}, Evaluated → Automatic, EvaluationMonitor → None, Exclusions → Automatic,
  ExclusionsStyle → None, FormatType → TraditionalForm, Frame → Automatic,
  FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic, FrameTicksStyle → {},
  GridLines → None, GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All,
  ImageSize → Automatic, ImageSizeRaw → Automatic, LabelStyle → {},
  MaxRecursion → Automatic, Mesh → Automatic, MeshFunctions → Automatic, MeshShading → None,
  MeshStyle → Automatic, Method → Automatic, PerformanceGoal → $PerformanceGoal,
  PlotLabel → None, PlotLegends → None, PlotPoints → Automatic, PlotRange → Automatic,
  PlotRangeClipping → True, PlotRangePadding → Automatic, PlotRegion → Automatic,
  PlotStyle → Automatic, PlotTheme → $PlotTheme, PreserveImageOptions → Automatic,
  Prolog → {}, RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
  TextureCoordinateFunction → Automatic, TextureCoordinateScaling → Automatic,
  Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}
```

```
Set[a, b]
```

```
b
```

```
a
```

```
b
```

```
Set[Evaluate[a], c]
```

```
c
```

```
a
```

```
c
```

```
b
```

```
c
```

```
?? Set
```

lhs = rhs evaluates *rhs* and assigns the result to be the value of *lhs*. From then on, *lhs* is replaced by *rhs* whenever it appears.
{l1, l2, ...} = {r1, r2, ...} evaluates the *ri*, and assigns the results to be the values of the corresponding *li*. >

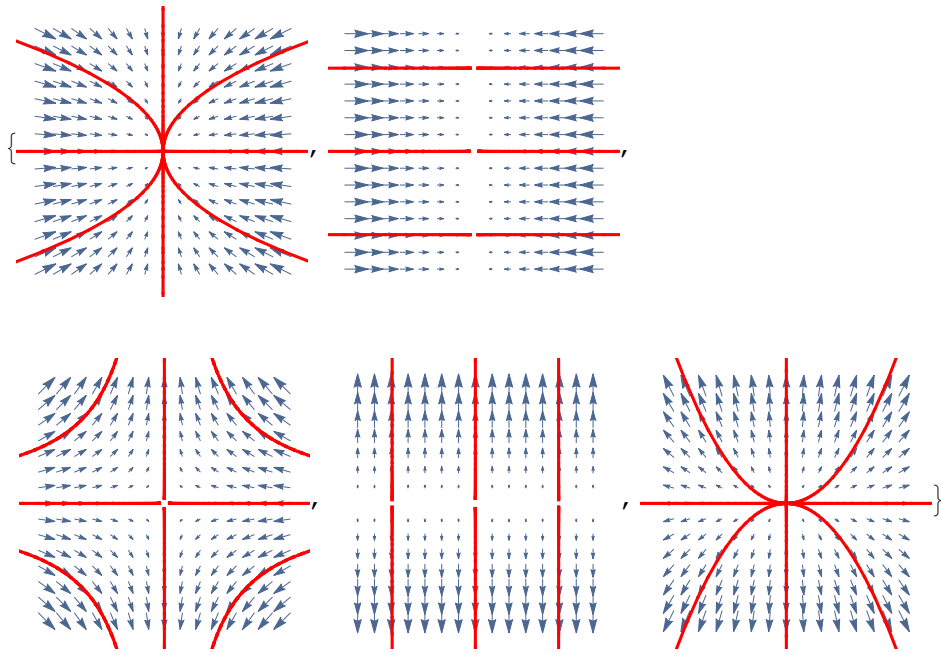
```
Attributes[Set] = {HoldFirst, Protected, SequenceHold}
```

```
With[{A =  $\begin{pmatrix} -2 & 0 \\ 0 & -1 \end{pmatrix}$ },
  Evaluate[Table[MatrixExp[t A]. $\begin{pmatrix} \text{Cos}[\theta] \\ \text{Sin}[\theta] \end{pmatrix}$ , { $\theta, \pi/4, 2\pi, \pi/4$ }]]]
  {{{  $\frac{e^{-2t}}{\sqrt{2}}$ ,  $\frac{e^{-t}}{\sqrt{2}}$  }, {0}, {e^{-t}}, {  $-\frac{e^{-2t}}{\sqrt{2}}$ ,  $\frac{e^{-t}}{\sqrt{2}}$  }, {  $-e^{-2t}$ , 0 } },
  {  $-\frac{e^{-2t}}{\sqrt{2}}$ ,  $-\frac{e^{-t}}{\sqrt{2}}$  }, {0}, {-e^{-t}}, {  $\frac{e^{-2t}}{\sqrt{2}}$ ,  $-\frac{e^{-t}}{\sqrt{2}}$  }, {  $e^{-2t}$ , 0 } }]
```

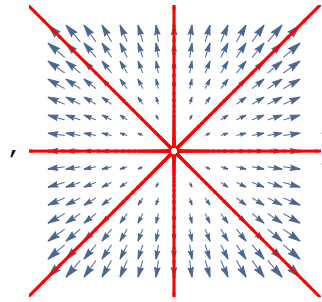
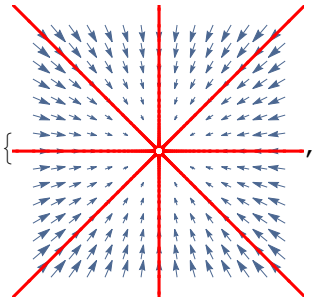
A
A

```
PP[A_] := Show[VectorPlot[A. $\begin{pmatrix} x \\ y \end{pmatrix}$ , {x, -1, 1}, {y, -1, 1}, Frame -> None],
  ParametricPlot[Evaluate[Table[MatrixExp[t A]. $\begin{pmatrix} \text{Cos}[\theta] \\ \text{Sin}[\theta] \end{pmatrix}$ , { $\theta, \pi/4, 2\pi, \pi/4$ }]]],
  {t, - $\pi$ ,  $\pi$ }, ColorFunction -> (Red &)],
  ImageSize -> 150]
```

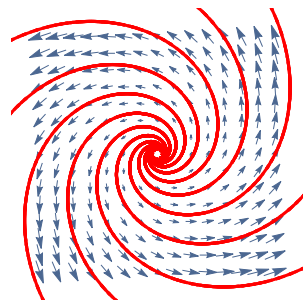
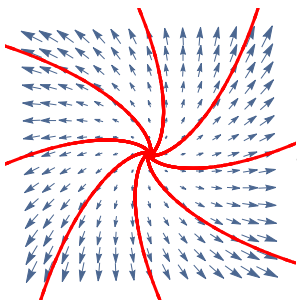
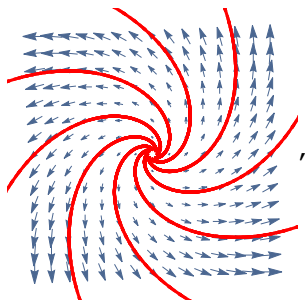
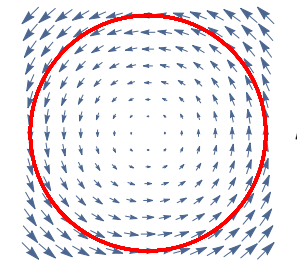
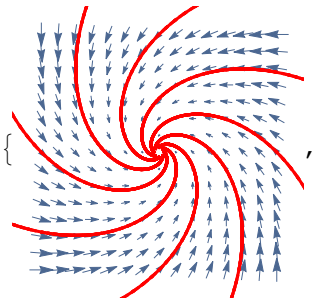
```
PP/@{ $\begin{pmatrix} -2 & 0 \\ 0 & -1 \end{pmatrix}$ ,  $\begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$ ,  $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ ,  $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ ,  $\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ }
```



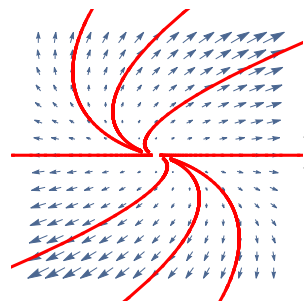
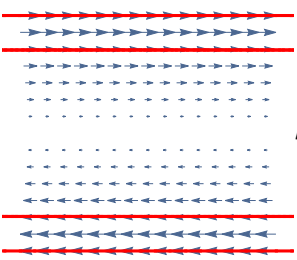
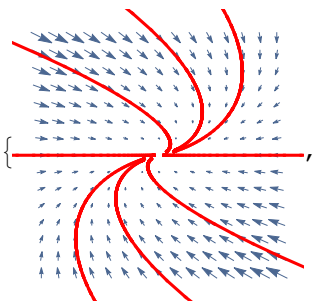
$$PP / @ \left\{ \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right\}$$



$$PP / @ \left\{ \begin{pmatrix} -1 & -1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 3 & -1 \\ 1 & 3 \end{pmatrix}, \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} \right\}$$



$$PP / @ \left\{ \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \right\}$$



Quadratic Phase Portraits

```

RandomQuadratic[] :=
  (a x^2 + sqrt[2] b x y + c y^2) /. a | b | c => RandomVariate[NormalDistribution[]];
RandomQuadratic[n_] := Table[RandomQuadratic[], {n}]

```

```
RandomQuadratic[2]
```

```
{2.08457 x^2 + 0.255033 x y - 0.646763 y^2, 1.88009 x^2 - 3.74949 x y + 1.19981 y^2}
```

```
Dynamic[$WorkingOn]
```

```
$WorkingOn
```

```
Off[NDSolve::ndsz, InterpolatingFunction::dmval];
```

```
quads = {};
```

```
Rasterize[GraphicsGrid[Table[
```

```
  quad = RandomQuadratic[2];
```

```
  AppendTo[quads, $WorkingOn = {{i, j} -> quad}];
```

```
  Show[Join[
```

```
    {VectorPlot[ $\frac{\text{quad}}{\text{Norm}[\text{quad}]^{3/4}}$ , {x, -1, 1}, {y, -1, 1}, Frame -> None]},
```

```
    Table[
```

```
      eqns = Join[
```

```
        Thread[{x'[t], y'[t]} == (quad /. {x -> x[t], y -> y[t]}),
```

```
        {x[0] == RandomReal[{-1, 1}], y[0] == RandomReal[{-1, 1]}}
```

```
      ];
```

```
      sol = NDSolve[eqns, {x, y}, {t, -1, 1}];
```

```
      ParametricPlot[
```

```
        Evaluate[{x[t], y[t]} /. sol],
```

```
        {t, -1, 1},
```

```
        PlotRange -> {{-1, 1}, {-1, 1}}, ColorFunction -> (Red &)
```

```
      ],
```

```
      {100}
```

```
    ]
```

```
  ]],
```

```
  {i, 5}, {j, 8}
```

```
]], ImageSize -> 960]
```

```
NDSolve::ndsz: At t == -0.564195, step size is effectively zero; singularity or stiff system suspected. >>
```

```
InterpolatingFunction::dmval:
```

```
Input value {-1.} lies outside the range of data in the interpolating function. Extrapolation will be used. >>
```

```
InterpolatingFunction::dmval:
```

```
Input value {-0.999959} lies outside the range of data in the interpolating function. Extrapolation will be used. >>
```

InterpolatingFunction::dmval :

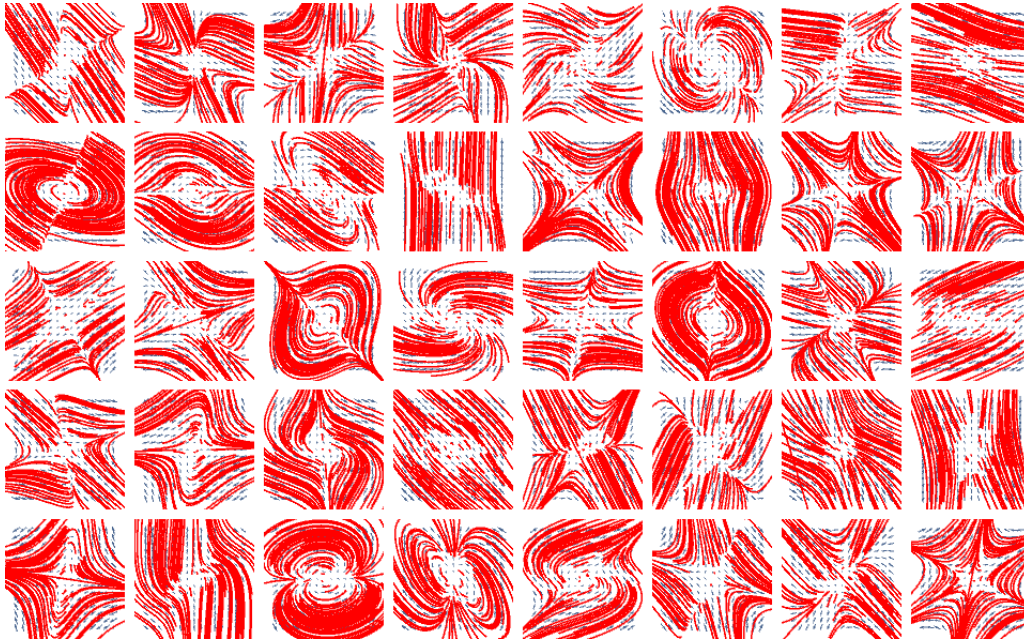
Input value {-0.999959} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

General::stop : Further output of InterpolatingFunction::dmval will be suppressed during this calculation. >>

NDSolve::npsz : At t == -0.622598, step size is effectively zero; singularity or stiff system suspected. >>

NDSolve::npsz : At t == 0.47818703543930857, step size is effectively zero; singularity or stiff system suspected. >>

General::stop : Further output of NDSolve::npsz will be suppressed during this calculation. >>



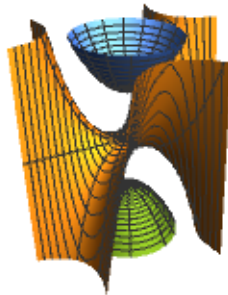
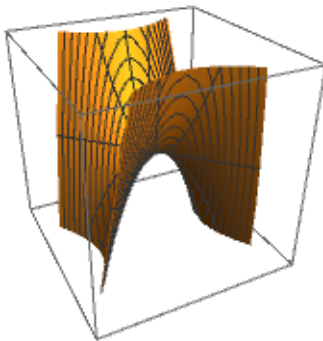
The Saddle and the Monkey Saddle

GraphicsRow @@

```

ParametricPlot3D[
  {r Cos[t], r Sin[t], 3 r^2 Cos[2 t]},
  {r, 0, 1}, {t, 0, 2 Pi},
  PlotRange -> {-1, 1}, Axes -> False,
  PlotPoints -> {10, 30},
  ViewPoint -> 0.8 {2.4, -1.2, 1.5}
]

ParametricPlot3D[{
  {r Cos[t], r Sin[t], 3 r^3 Cos[3 t]},
  {r Cos[t], r Sin[t], 1 + r^2} / 2,
  {r Cos[t], r Sin[t], -1 - r^2} / 2
}, {r, 0, 1}, {t, 0, 2 Pi},
PlotRange -> {-1, 1}, Boxed -> False,
Axes -> False,
PlotPoints -> {10, 30},
ViewPoint -> 0.8 {2.4, -1.2, 1.5}] // Rasterize
    
```



```
z = Assuming[x > 0 & y > 0, Re[(x + i y)^3 // Expand]]
```

$$-3 \operatorname{Im}[x^2 y] + \operatorname{Im}[y^3] + \operatorname{Re}[x^3 - 3 x y^2]$$

```
Simplify[Re[(x + i y)^3 // Expand], x > 0 & y > 0]
```

$$x^3 - 3 x y^2$$

$$z = x^3 - 3 x y^2$$

$$x^3 - 3 x y^2$$

```
quad = {-Dy z, Dx z}
```

$$\{6 x y, 3 x^2 - 3 y^2\}$$

```
Show[Join[
  {VectorPlot[ $\frac{\text{quad}}{\text{Norm}[\text{quad}]^{3/4}}$ , {x, -1, 1}, {y, -1, 1}, Frame -> None]},
  Table[
    eqns = Join[
      Thread[{x'[t], y'[t]} == (quad /. {x -> x[t], y -> y[t]}),
      {x[0] == RandomReal[{-1, 1}], y[0] == RandomReal[{-1, 1}]}
    ];
    sol = NDSolve[eqns, {x, y}, {t, -1, 1}];
    ParametricPlot[
      Evaluate[{x[t], y[t]} /. sol],
      {t, -1, 1},
      PlotRange -> {{-1, 1}, {-1, 1}}, ColorFunction -> (Red &)
    ],
    {100}
  ]
]]
```

NDSolve::ndsz : At t == 0.8092588132364427, step size is effectively zero; singularity or stiff system suspected. >>

NDSolve::ndsz : At t == -0.599128, step size is effectively zero; singularity or stiff system suspected. >>

NDSolve::ndsz : At t == 0.32333599670161095, step size is effectively zero; singularity or stiff system suspected. >>

General::stop : Further output of NDSolve::ndsz will be suppressed during this calculation. >>

InterpolatingFunction::dmval :

Input value {-1.} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

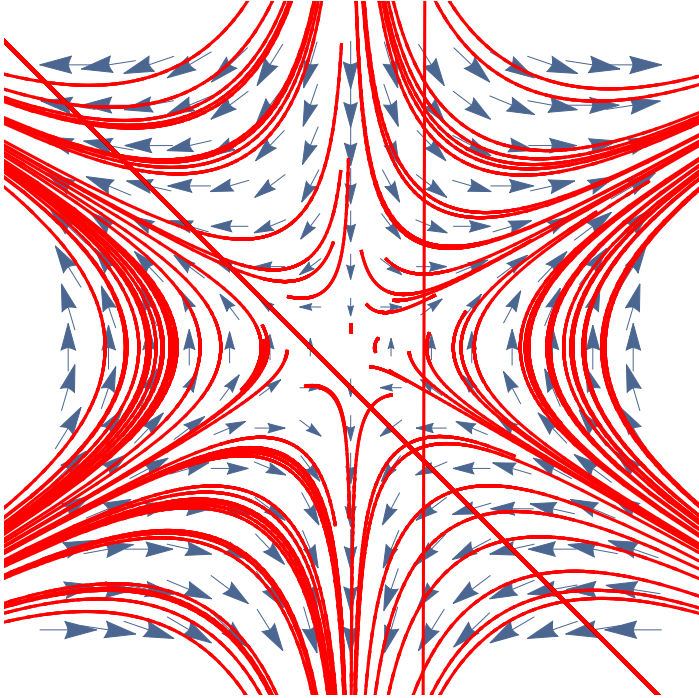
InterpolatingFunction::dmval :

Input value {-0.999959} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

InterpolatingFunction::dmval :

Input value {-0.999959} lies outside the range of data in the interpolating function. Extrapolation will be used. >>

General::stop : Further output of InterpolatingFunction::dmval will be suppressed during this calculation. >>



The Willerton Fish

Willerton: It is amazing to plot the values of v_2 against the values of v_3 on a large sample of knots.

```
<< KnotTheory`
```

```
GD[L_] :=
```

```
GD@@PD[L] /. X[i_, j_, k_, l_] => If[PositiveQ[X[i, j, k, l]], Ap[l, i], Am[j, i]];
Column[GD /@ AllKnots[{3, 6}]]
```

From Polyak-Viro's "Gauss Diagram Formulas for Vassiliev Invariants", IMRN 11 (1994) 445-453:

3.B THEOREM 1. *If G is any based Gauss diagram of a knot K then*

$$(4) \quad v_2(K) = \left\langle \left[\begin{array}{c} \circlearrowleft \\ \circlearrowright \end{array} \right], G \right\rangle,$$

where $v_2(K)$ is the Vassiliev invariant of degree 2 which takes values 0 on the unknot and 1 on a trefoil.

4.A THEOREM 2. *If G is a Gauss diagram of a knot K then*

$$(5) \quad v_3(K) = \left\langle \frac{1}{2} \left[\begin{array}{c} \circlearrowleft \\ \oplus \end{array} \right] + \left[\begin{array}{c} \circlearrowleft \\ \otimes \end{array} \right], G \right\rangle,$$

where $v_3(K)$ is the Vassiliev invariant of degree 3 which takes values 0 on the unknot, +1 on the right trefoil and -1 on the left trefoil.