```
> restart;
  with(plots):
  with(plottools):
  with(DEtools):
  alias(alpha=LambertW(exp(-1)));
```

```
Warning, the name changecoords has been redefined

Warning, the name arrow has been redefined

Warning, the name translate has been redefined
```

α

Lyapunov functions should be proper

About this worksheet

Author and Date:

Matthias Kawski
Arizona State University
http://math.asu.edu/~kawski
Original version: November 2006, release 8

>

Content, Purpose and Use

This worksheet constructs a pictorial counter example that forcefully illustrates how a system may fail to be asy stable even though there is a positive definite function whose

derivative along solution curves is strictly negative definite.

>

The main objective is to provide a powerful picture that makes everyone aware of the need of technical hypotheses on Lyapunov functions to conclude (global) asymptotic stability (being proper, or "radially unbounded")

>

The implementation is interesting by itself as e.g. simpler attempts using quadratically

growing Lyapunov functions did not provide pictures. Also, the construction of a

suitable system for the Lyapunov function is an interesting ploy mixing gradient and "Hamiltonian" fields.
Note, this is still somewhat incomplete as we have not yet checked that the solution indeed moves FAST enough twds infinity along the positive x-axis before getting too close to it ... this is essential as the "lying outside the basin of attraction" is based on the integral of Vdot w.r.t. TIME being bounded carefully....

>

Updates and log of modifications.

none yet.

>

Constructing a suitable, easily viewable Lyapunov function

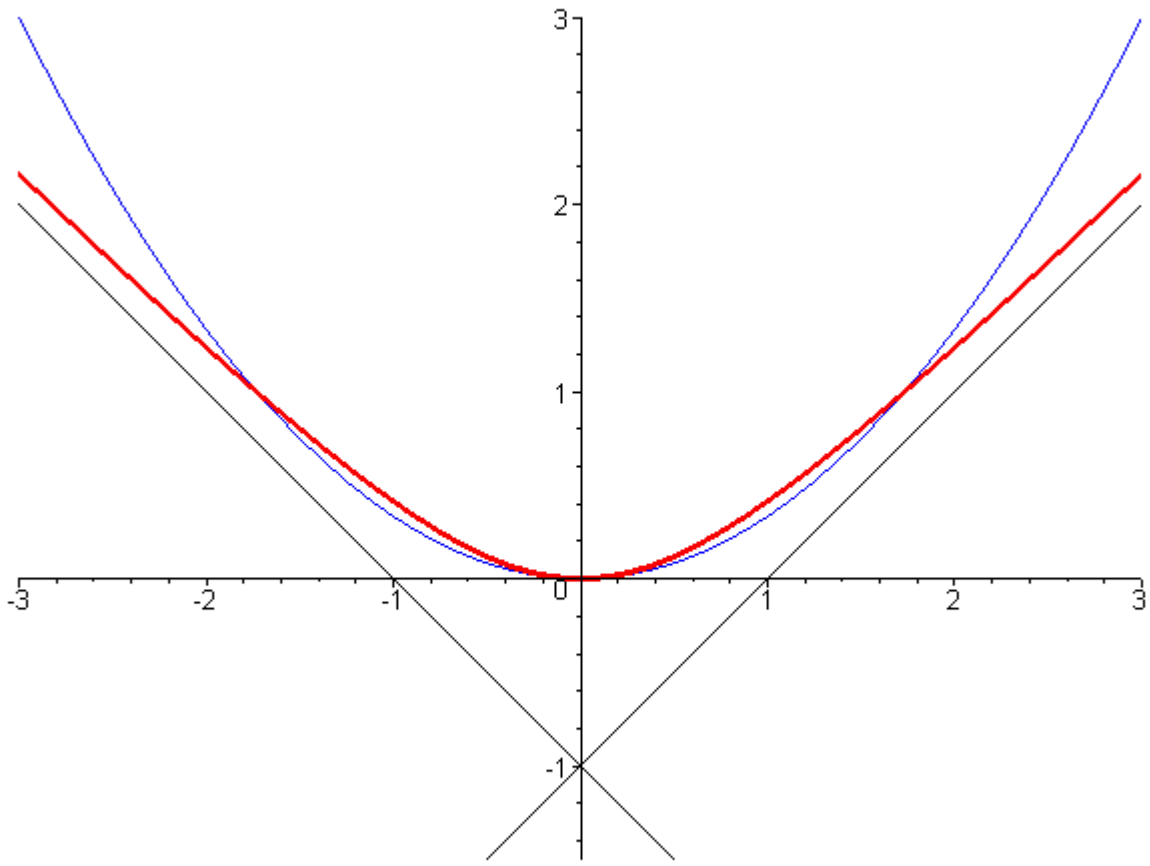
Preliminary steps --- just for those interested in designing graphs

For 3D-viewing purposes it turns out that functions which exhibit linear rather than quadratic growth-rates are aesthetically more appealing and easier to read.
Start with a hyperbola (not parabola) in the transverse direction:

```
> fy:=y->sqrt(y^2+1)-1;
  plot([fy,s->s^2/3,t->t-1,t->-t-1],-3..3,-3/2..3,
```

```
color=[red,blue,black,black],thickness=[3,1,1,1],
scaling=constrained);
```

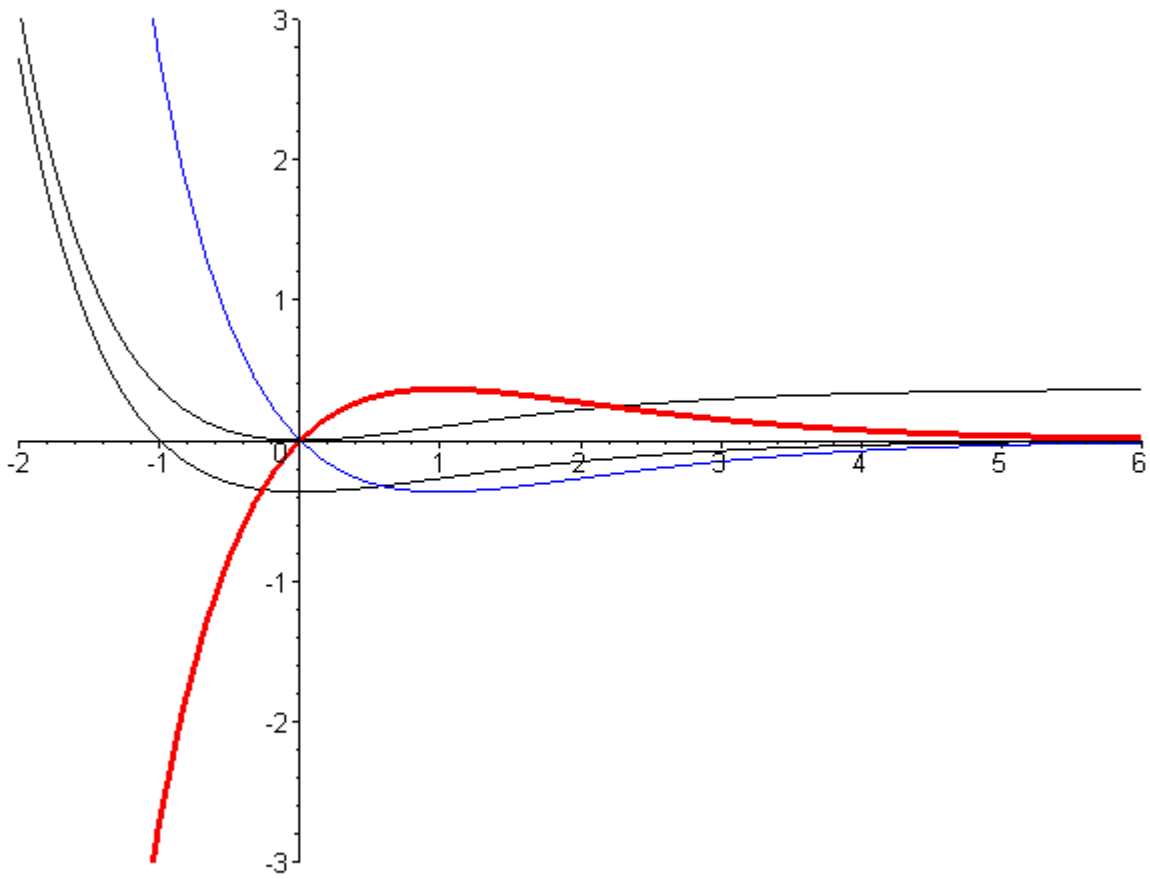
$$fy := y \to \sqrt{y^2 + 1} - 1$$



In the other direction we want a function which is bounded and unbounded, respectively, as the input grows to plus and minus infinity. It also should have a single global extremum. The natural basic building block is $x \to x\,\mathbf{e}^{(-x)}$, which is easily flipped over, and translated so that we get a positive definite function whose global minimum is at the origin.

```
> fx:=x->x*exp(-x);
  plot([fx,-fx,t->-fx(t+1),t->-fx(t+1)+exp(-1)],-2..6,-3..3,
       color=[red,blue,black,black],thickness=[3,1,1,1],
       scaling=constrained);
```

$$fx := x \to x\,\mathbf{e}^{(-x)}$$



To get nicer pictures, we would like a more pronounced "dip", while reducing the growth rate for negative x. Simply dividing through by ($1+e^{(-x)}$) will achieve this second objective.

```
> ff:=x->(-fx(x+1)+exp(-1))/(1+exp(-x)):
  'ff(x)'=ff(x);
```

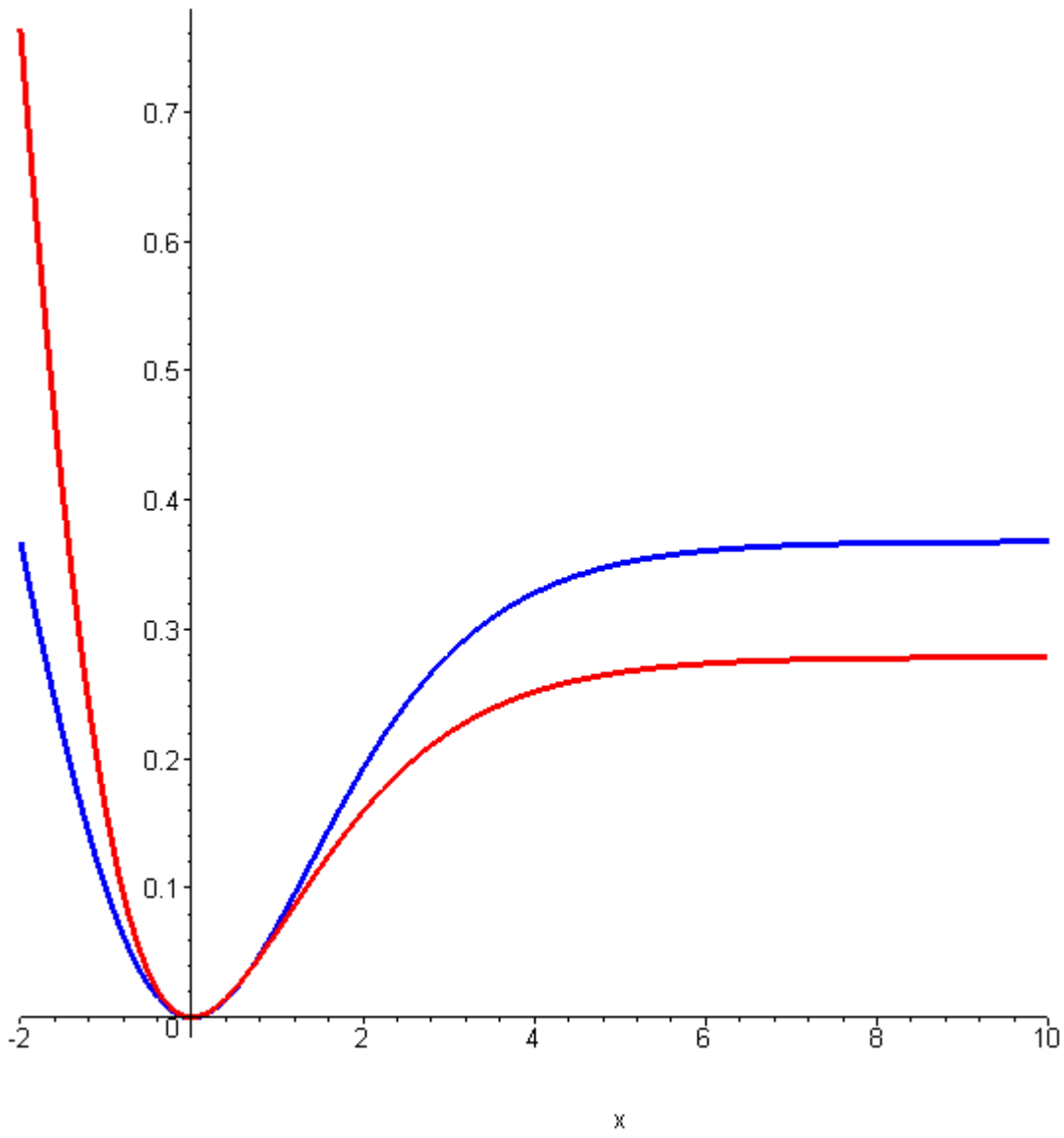
$$ff(x) = \frac{-(x+1)\,e^{(-x-1)} + e^{(-1)}}{1+e^{(-x)}}$$

The original worksheet used the similar function (just slightly different scaling) resulting in shifts (horizontal, vertical) that involve the LambertW function

```
> f:=alpha
    -(x+1+alpha)*exp(-x-1-alpha)/(1+exp(-x-1-alpha));
```

$$f := \alpha - \frac{(x+1+\alpha)\,e^{(-x-1-\alpha)}}{1+e^{(-x-1-\alpha)}}$$

```
> plot([f(x),ff(x)],x=-2..10,
      color=[red,blue],thickness=3);
```



>

The not-so Lyapunov function and its graph

```
> alpha=evalf(alpha);
  f:=alpha-(x+1+alpha)*exp(-x-1-alpha)/(1+exp(-x-1-alpha)):
  V:=f+(sqrt(y^2+1)-1)/2;
   Vfcn:=unapply(V,(x,y));
```

$$\alpha = 0.2784645428$$

$$V := \alpha - \frac{(x+1+\alpha)\,\mathbf{e}^{(-x-1-\alpha)}}{1+\mathbf{e}^{(-x-1-\alpha)}} + \frac{\sqrt{y^2+1}}{2} - \frac{1}{2}$$

$$Vfcn := (x,y) \to \alpha - \frac{(x+1+\alpha)\,\mathbf{e}^{(-x-1-\alpha)}}{1+\mathbf{e}^{(-x-1-\alpha)}} + \frac{1}{2}\sqrt{y^2+1} - \frac{1}{2}$$

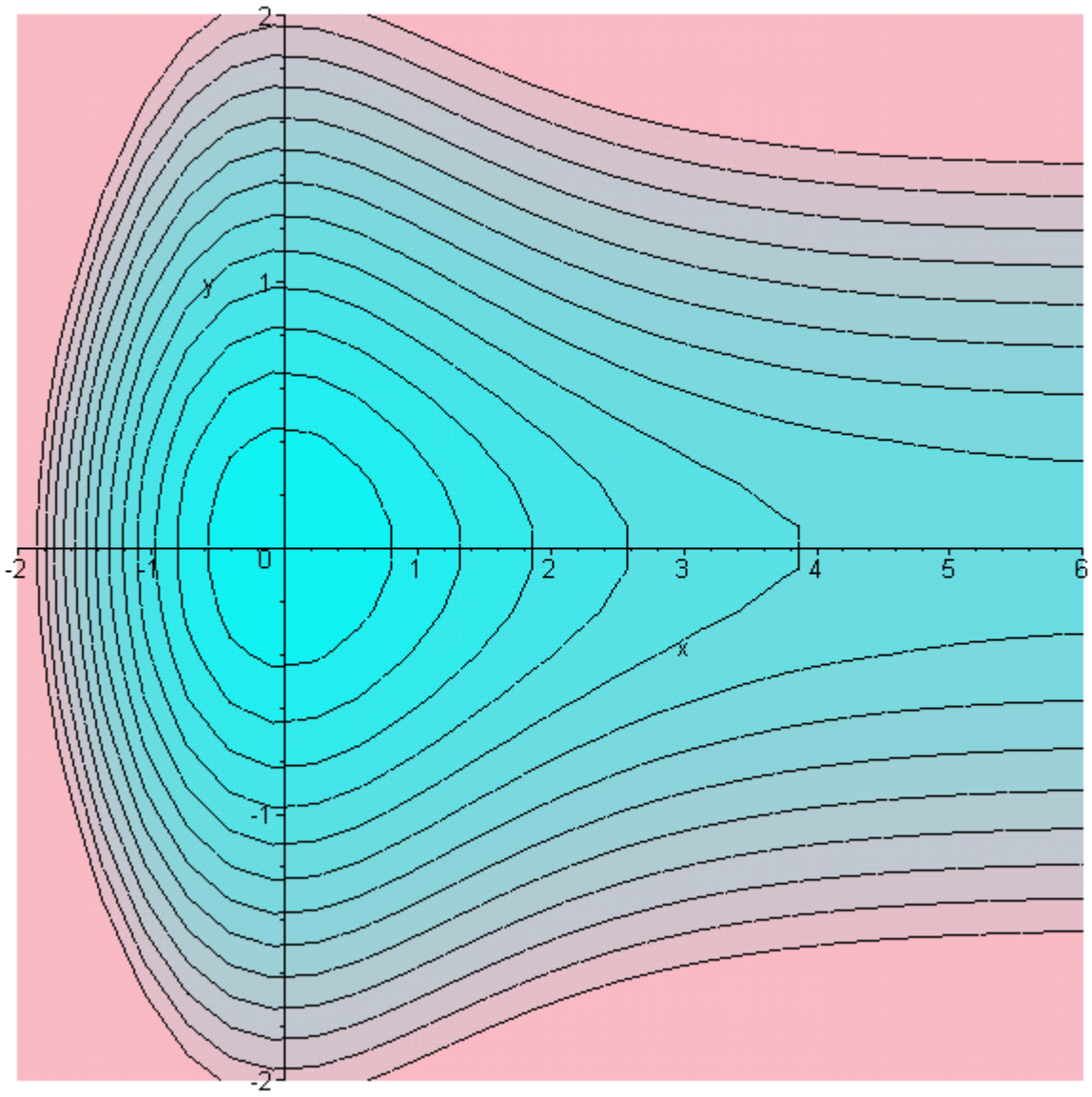
plot range (domain)

```
> a:=2:
```

lift arrows above the surface

```
> eps:=0.01:
```

```
> cplot:= contourplot(V,x=-a..3*a,y=-a..a,contours=[seq(k/20,k=0..13)],
      filled=true,coloring=[cyan,pink],thickness=1):
  display(cplot);
```

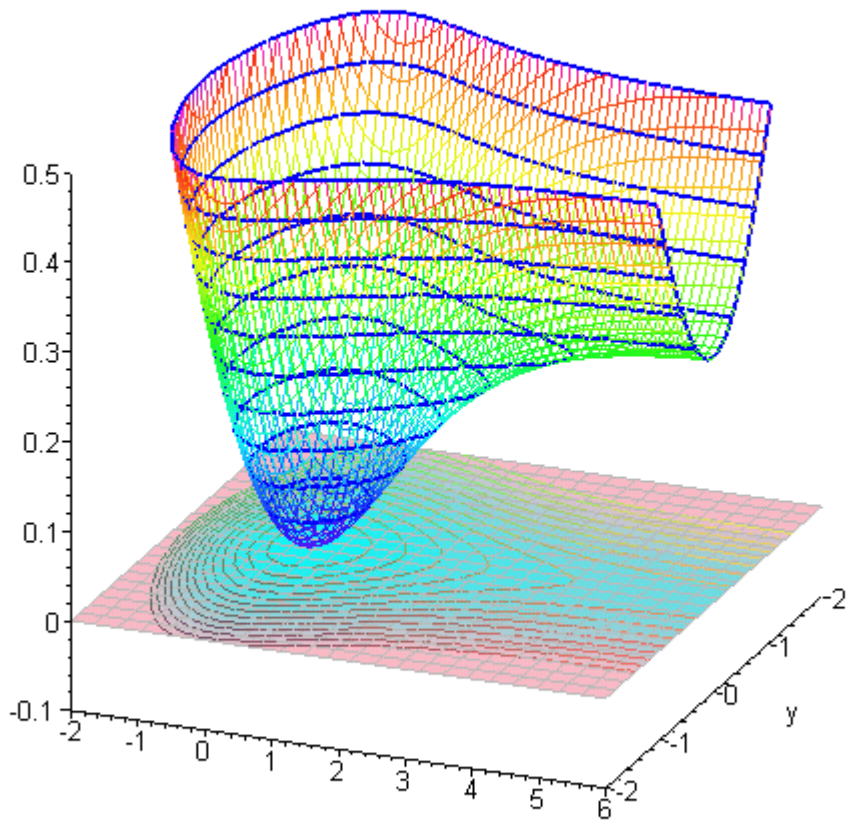


Procedures from the plot-tools package to help display together 2d- and 3d plot structures at various heights, even variable heights --- lifted to the graph of the Lyapunov function)

```
> trafo:= transform((x,y) ->[x,y,0]):
  traf5:= transform((x,y) ->[x,y,eps]):
  lift := transform((x,y) ->[x,y,Vfcn(x,y)]):
```

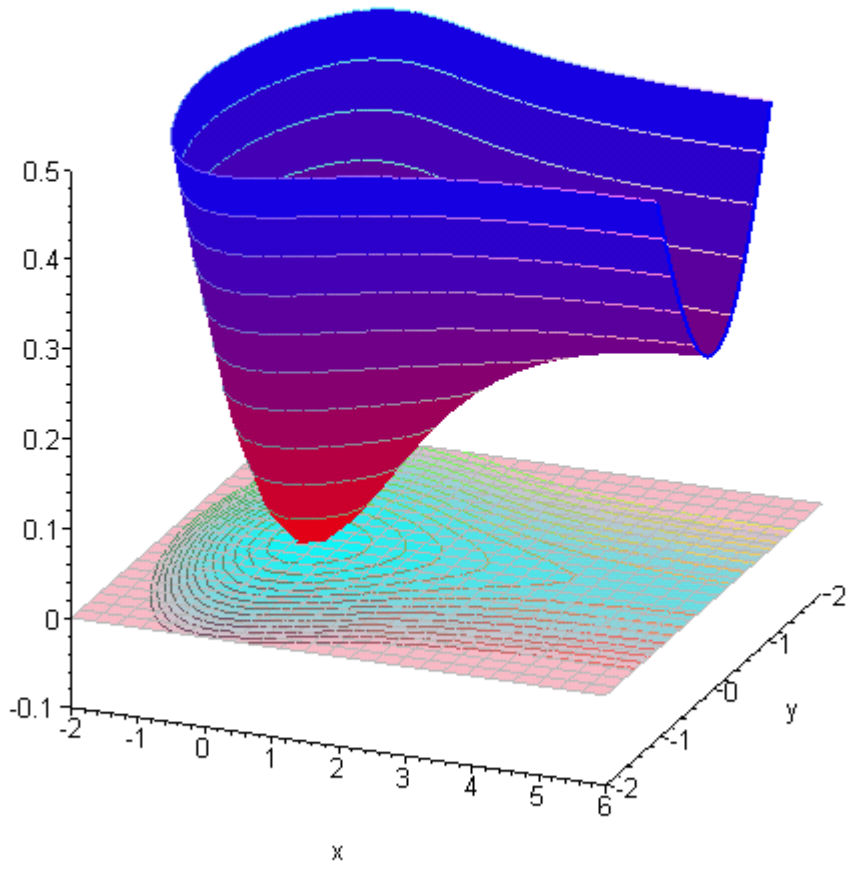
An absolutely beautiful image --- great proportions, cut-offs, viewing angle ....
Ideas for improvements: Tone down the color -- grey shades may even be better to SEE foreground vs background surface....

```
> display([
      trafo(cplot),
      plot3d(V,x=-a..3*a,y=-a..a,style=wireframe,shading=ZHUE,grid=[60,48]),
      plot3d(V,x=-a..3*a,y=-a..a,style=contour,contours=[seq(k/20,k=0..10)],
          color=blue,thickness=2),
      spacecurve([3*a,y,subs(x=3*a,V)],y=-a..a,color=blue,thickness=2),
      plot3d(0,x=-a..3*a,y=-a..a,style=wireframe,color=grey,grid=[24,24])],
    view=[-a..3*a,-a..a,-0.1..0.5],axes=frame,orientation=[-68,69]);
```



Using solid surfaces instead -- maybe lighting or transparency might make this even better

```
> display([trafo(cplot),
   contourplot3d(V,x=-a..3*a,y=-a..a,style=wireframe,contours=[seq(k/20,k=0..10)],
      filled=true,coloring=[red,blue]),
   spacecurve([3*a,y,subs(x=3*a,V)],y=-a..a,color=blue,thickness=2),
   plot3d(0,x=-a..3*a,y=-a..a,style=wireframe,color=grey,grid=[24,24])],
        view=[-a..3*a,-a..a,-0.1..0.5],axes=frame,orientation=[-68,69]);
```



>

Constructing a matching system of DEs and its integral curves

Gradient system derived from the Lyapunov function

Given a scalar function one immediate, natural, vector field (differential equation) associated with it is obtained from the gradient of the scalar function.

```
> gradV:=map(q->simplify(diff(V,q)),[x,y]);
```

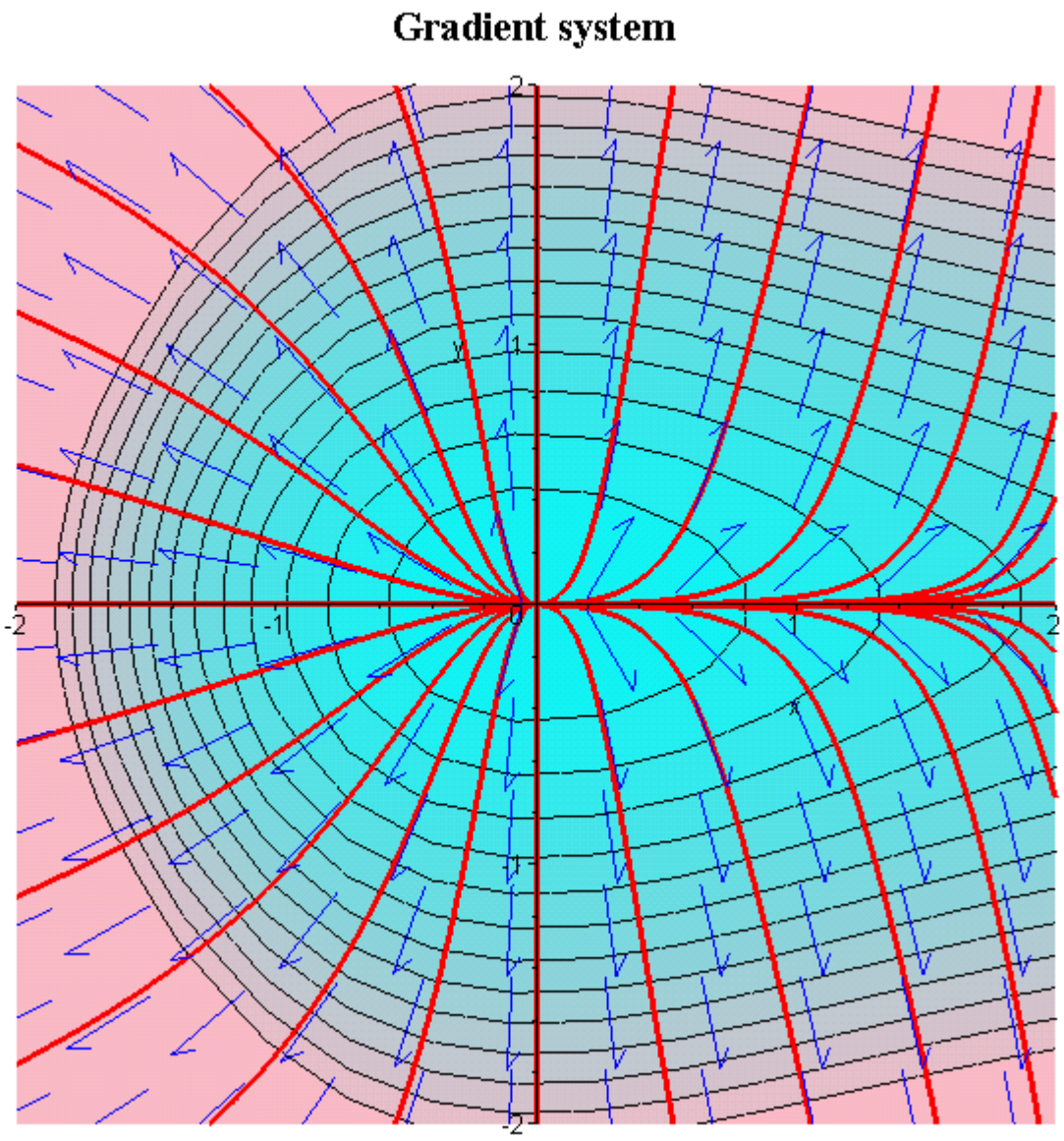
$$gradV := \left[-\frac{e^{(-x-1)}\,\alpha\,(e^{(-x-1)}\,\alpha - x\,e^{(-1)} - \alpha\,e^{(-1)})}{(e^{(-1)} + e^{(-x-1)}\,\alpha)^2},\ \frac{y}{2\sqrt{y^2+1}} \right]$$

The system of differential equations associated to the gradient vector field

```
> gradsys:=zip((a,b)->diff(a(t),t)=subs(x=x(t),y=y(t),b),[x,y],gradV):
  op(gradsys);
```

$$\frac{d}{dt}x(t) = -\frac{e^{(-x(t)-1)}\,\alpha\,(e^{(-x(t)-1)}\,\alpha - x(t)\,e^{(-1)} - \alpha\,e^{(-1)})}{(e^{(-1)} + e^{(-x(t)-1)}\,\alpha)^2},\ \frac{d}{dt}y(t) = \frac{1}{2}\frac{y(t)}{\sqrt{y(t)^2+1}}$$

```
> display([
    DEplot(gradsys,[x(t),y(t)],t=-20..3,
      [seq([x(0)=2*cos(theta),y(0)=2*sin(theta)],
        theta=[seq(k*Pi/12,k=1..23),seq(k*Pi/36,k=-2..2)])],
      stepsize=.2,
      title=`Gradient system`,
      titlefont=[TIMES,BOLD,16],
      color=blue,
      linecolor=red,
      #arrows=MEDIUM,
      method=rkf45),
  cplot
  ],view=[-2..2,-2..2]);
```




```
>
```

Hamiltonian systems derived from the Lyapunov function

A second natural choice for scalar functions on the plane simple is the gradient field rotated by 90 degrees. Rather than perpendicular to the contours of the given scalar function like the gradient field, this field will be so that the contours of the scalar function are actually integral curves of the vector field / differential

equation.
In higher dimensions, there are many vector fields that are tangent to the level-(hyper)surfaces, and one obtains a distinguished "Hamiltonian vector field" when a distinguished "symplectic form" had been specified which takes the role of the form (dx wedge dy) in the plane.

```
> hamvf:=(q->[q[2],-q[1]])(gradV);
```

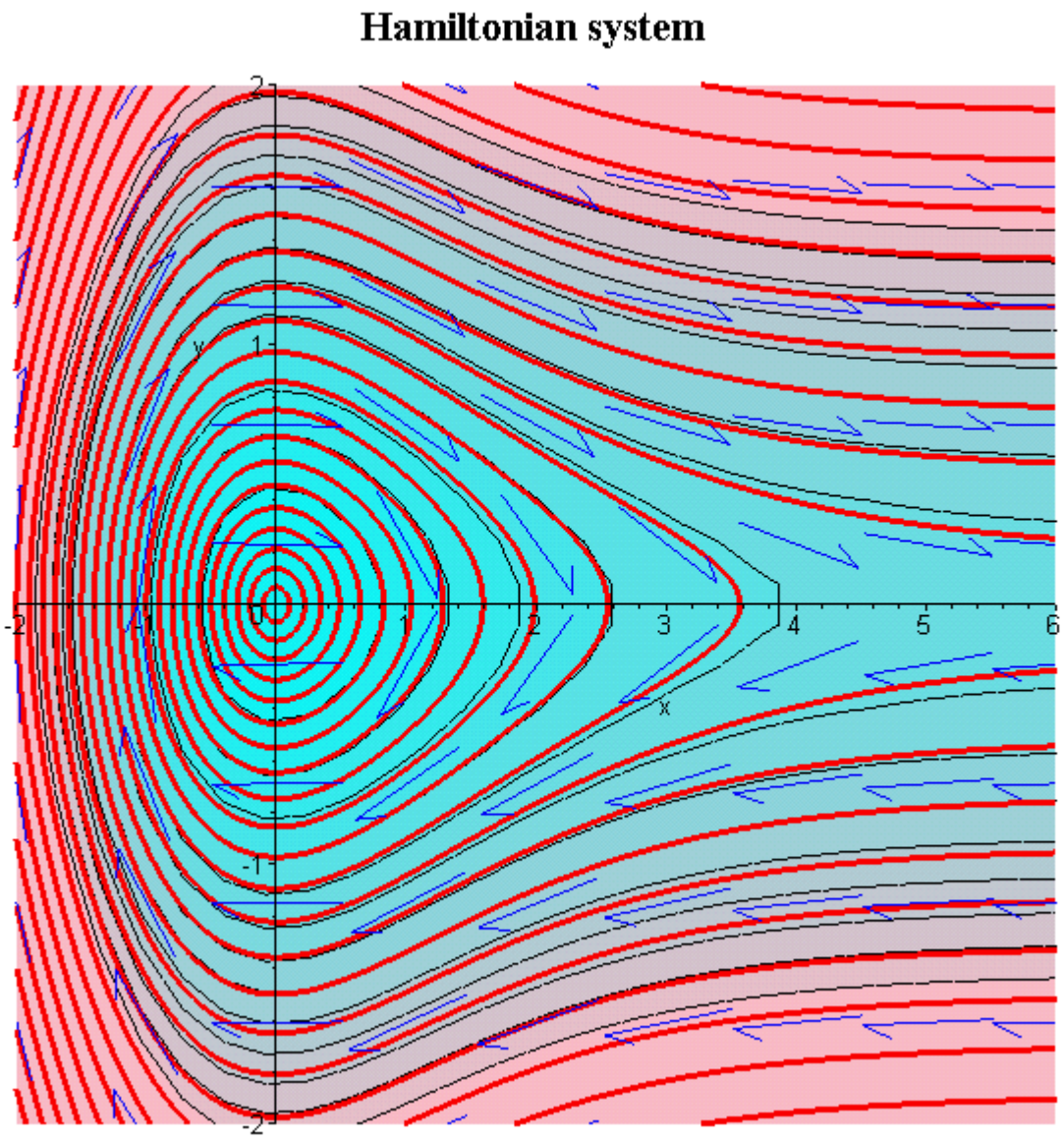
$$hamvf := \left[\frac{y}{2\sqrt{y^2+1}}, \frac{\mathbf{e}^{(-x-1)}\,\alpha\,(\mathbf{e}^{(-x-1)}\,\alpha - x\,\mathbf{e}^{(-1)} - \alpha\,\mathbf{e}^{(-1)})}{(\mathbf{e}^{(-1)} + \mathbf{e}^{(-x-1)}\,\alpha)^2} \right]$$

The system of differential equations associated to the Hamiltonian vector field

```
> hamsys:=zip((a,b)->diff(a(t),t)=subs(x=x(t),y=y(t),b),[x,y],hamvf):
  op(hamsys);
```

$$\frac{d}{dt}\mathrm{x}(t) = \frac{1}{2}\frac{\mathrm{y}(t)}{\sqrt{\mathrm{y}(t)^2+1}}, \frac{d}{dt}\mathrm{y}(t) = \frac{\mathbf{e}^{(-\mathrm{x}(t)-1)}\,\alpha\,(\mathbf{e}^{(-\mathrm{x}(t)-1)}\,\alpha - \mathrm{x}(t)\,\mathbf{e}^{(-1)} - \alpha\,\mathbf{e}^{(-1)})}{(\mathbf{e}^{(-1)} + \mathbf{e}^{(-\mathrm{x}(t)-1)}\,\alpha)^2}$$

```
> display([
DEplot(hamsys,[x(t),y(t)],t=-40..40,
    [seq([x(0)=-k/10,y(0)=0],k=1..30)],
    stepsize=.2,
    title=`Hamiltonian system`,
    titlefont=[TIMES,BOLD,16],
    color=blue,
    linecolor=red,
    #arrows=MEDIUM,
    method=rkf45),
cplot
],view=[-2..6,-2..2]);
```




```
>
```

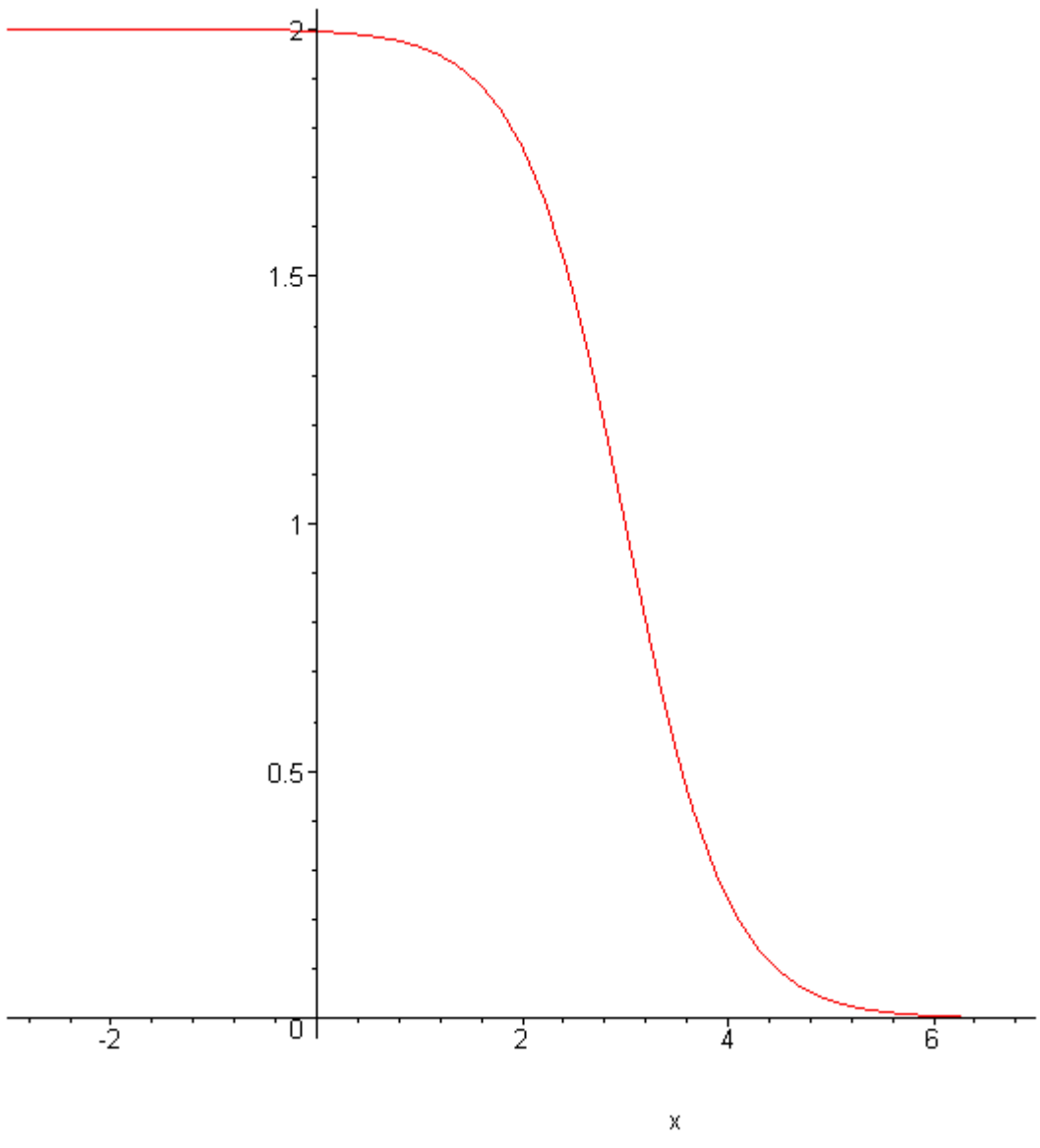
The desired system

The desired system is basically the Hamiltonian vector field with a very slight perturbation by adding a state-dependent negative multiple of the gradient field.
If this multiple is nowhere zero, then the time derivative of the Lyapunov function is strictly negative definite.

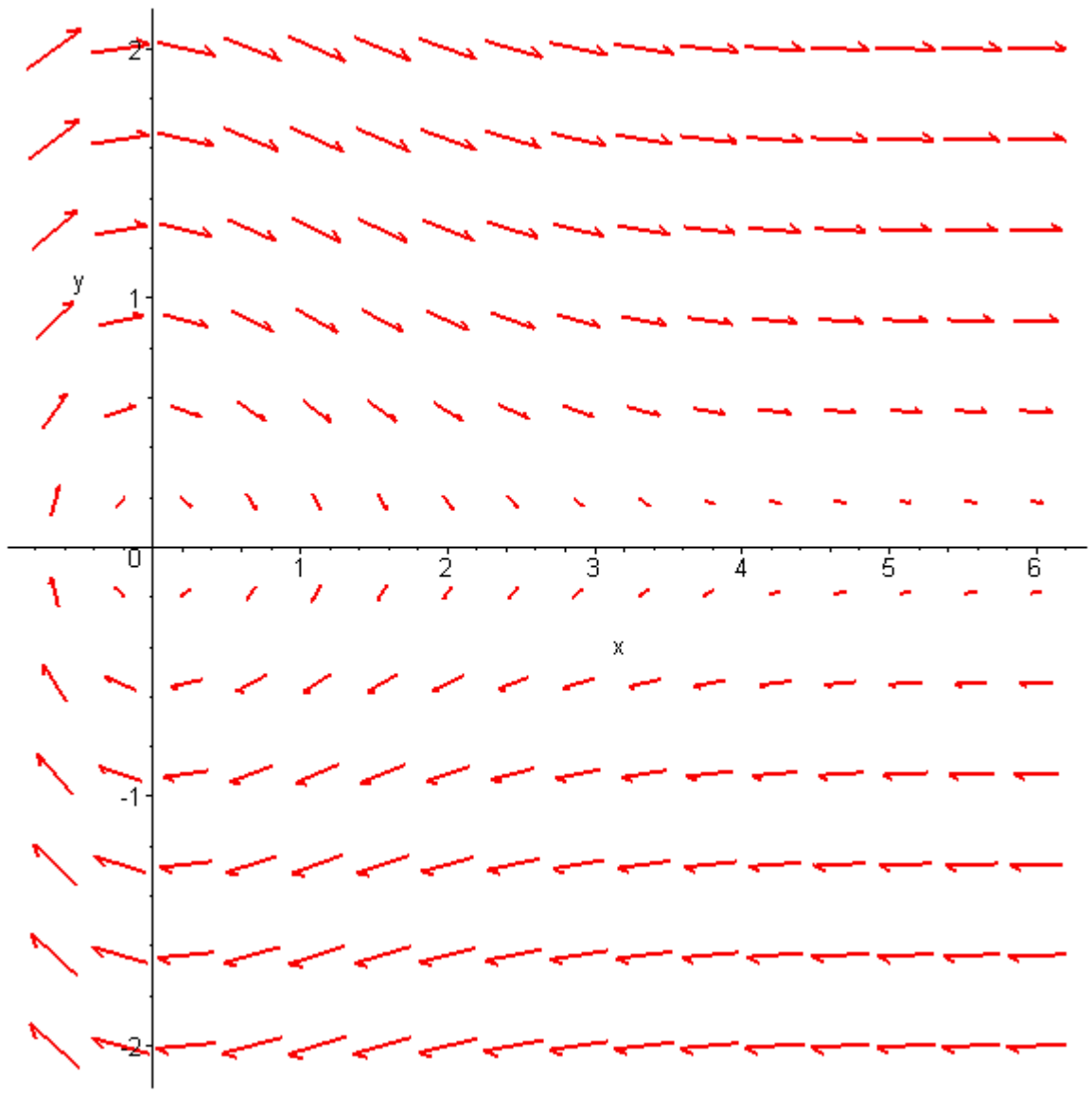
On the other hand we want solutions that climb down into the "trough" to do so ever so gingerly / slowly that they never reach the asymptotic minimal height of the trough. We may do this by choosing a multiple that has a well-defined finite integral as x goes to infinity ....

CAUTION: This is unfinished --- basic concern is that as y goes to zero, approaching the bottom of the trough, dV/dy also goes to zero, and hence the x-speed dx/dt goes to zero. May have to further adjust the new field by speeding it suitably up so that solutions go to infinity before getting to the maximal trough height....
ANYHOW, it can be done smoothly, and for us the general picture is what really matters.

```
> plot((1+tanh(-x+3)),x=-3..7);
```




```
> descvf:=zip((a,b)->a+(1+tanh(3-x))/50*(-b),hamvf,gradV):
  fieldplot(descvf,x=-a/3..3*a,y=-a..a,thickness=2,color=red,grid=[16,12]);
```



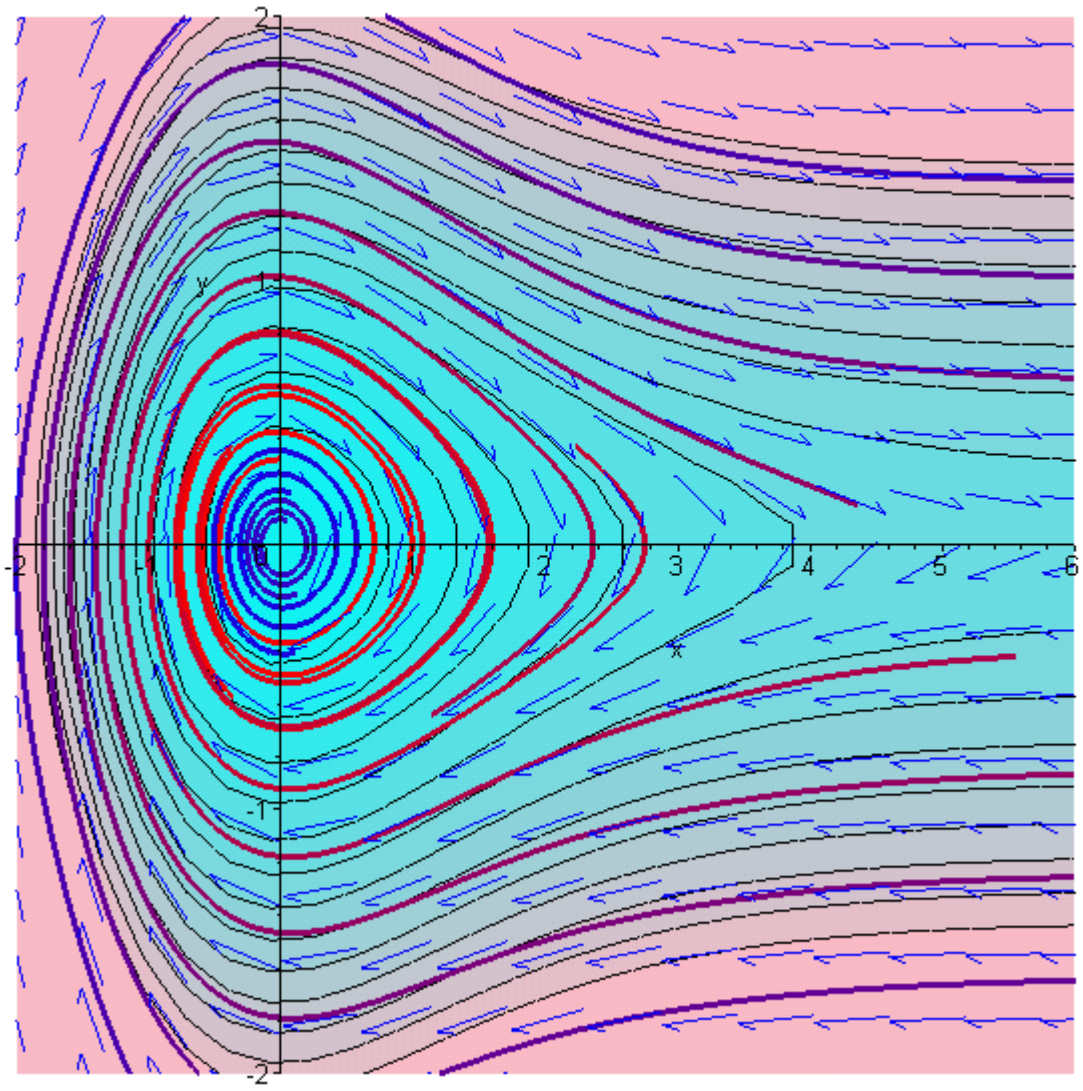

```
> descsys:=zip((a,b)->diff(a(t),t)=subs(x=x(t),y=y(t),b),[x,y],descvf):
  op(descsys);
```

$$\frac{d}{dt}x(t) = \frac{1}{2}\frac{y(t)}{\sqrt{y(t)^2+1}} + \frac{1}{50}\frac{(1-\tanh(x(t)-3))\,e^{(-x(t)-1)}\,\alpha\,(e^{(-x(t)-1)}\,\alpha - x(t)\,e^{(-1)} - \alpha\,e^{(-1)})}{2\,(e^{(-1)}+e^{(-x(t)-1)}\,\alpha)},$$

$$\frac{d}{dt}y(t) = \frac{e^{(-x(t)-1)}\,\alpha\,(e^{(-x(t)-1)}\,\alpha - x(t)\,e^{(-1)} - \alpha\,e^{(-1)})}{2\,(e^{(-1)}+e^{(-x(t)-1)}\,\alpha)} - \frac{1}{100}\frac{(1-\tanh(x(t)-3))\,y(t)}{\sqrt{y(t)^2+1}}$$

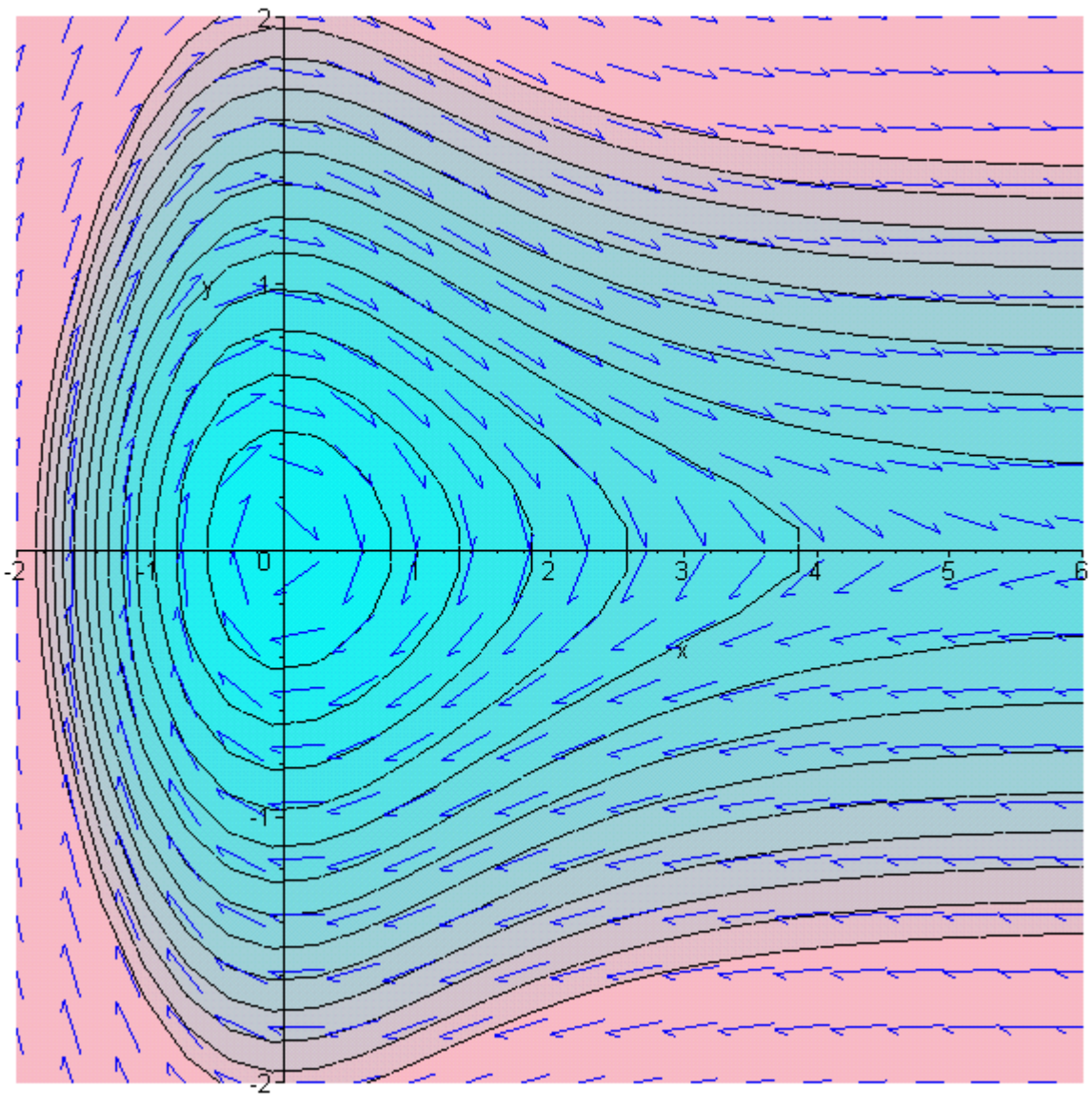
```
> N:=10:
  display([
    DEplot(descsys,[x(t),y(t)],t=-25..25,
        [seq([x(0)=-k/(N/2),y(0)=0],k=1..N)],
        stepsize=.2,
        title=`Strictly decaying: No closed curves, all spirals`,
        titlefont=[TIMES,BOLD,16],
        color=blue,
        linecolor=[seq(COLOR(RGB,k/N,0,1-k/N),k=1..N)],
        #arrows=MEDIUM,
        method=rkf45),
  cplot
  ],view=[-2..6,-2..2]);
```

Strictly decaying: No closed curves, all spirals



Just the vector field and contour-plot -- to later be inserted in a 3D-composite image

```
> flat:=display([
    dfieldplot(descsys,[x(t),y(t)],t=675..8709,x=-2..6,y=-2..2,
       color=blue),
    cplot
    ],view=[-2..6,-2..2]):
  display(flat);
```



Custom picked initial conditions for two distinct trajectories in the final 3D-pix:

```
> ic1:=x(0)=6,y(0)=-1/2;
  ic2:=x(0)=6,y(0)=-1;
```

$$ic1 := x(0) = 6, y(0) = \frac{-1}{2}$$

$$ic2 := x(0) = 6, y(0) = -1$$

and the associated numeric solutions of the corresponding initial value problems

```
> dsol1:=dsolve({op(descsys),ic1},numeric);
  dsol2:=dsolve({op(descsys),ic2},numeric);
```

$$dsol1 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$

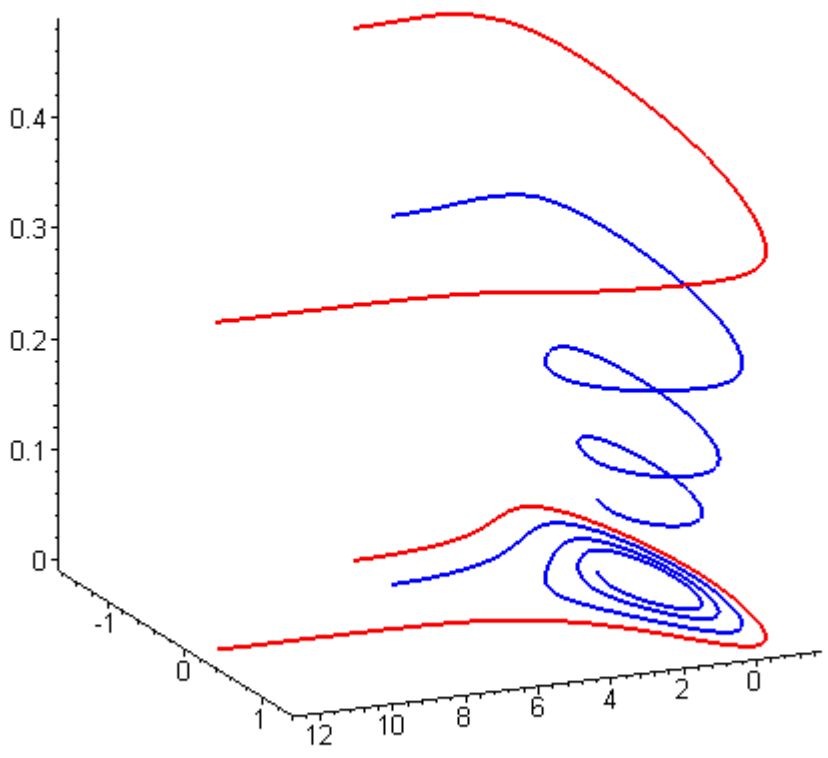
$$dsol2 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$

"Lifting" points from the xy-plane to the graph of the Lyapunov function

```
> llift:=q->subs(x=q[1],y=q[2],[x,y,V]):
```

First check: plot the solution curves together with their vertical lifts

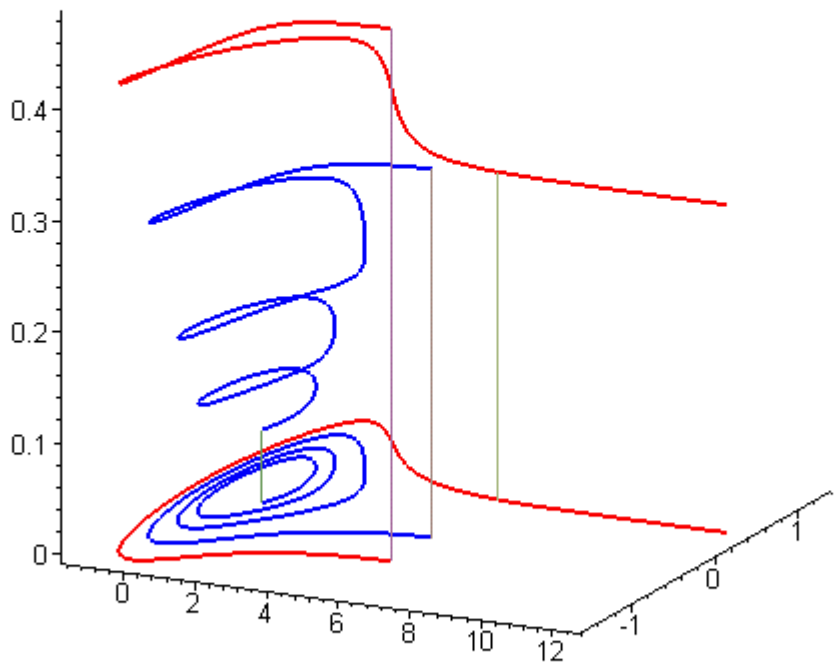
```
> just4curves:=display([
  spacecurve(map(tt->[seq(rhs(dsol1(tt)[k]),k=2..3),0],
  [seq(tau/10,tau=0..1000)]),thickness=2,color=blue),
  spacecurve(map(tt->[seq(rhs(dsol2(tt)[k]),k=2..3),0],
  [seq(tau/10,tau=0..1000)]),thickness=2,color=red),
  spacecurve(map(tt->llift([seq(rhs(dsol1(tt)[k]),k=2..3)]),
  [seq(tau/10,tau=0..1000)]),thickness=2,color=blue),
  spacecurve(map(tt->llift([seq(rhs(dsol2(tt)[k]),k=2..3)]),
  [seq(tau/10,tau=0..1000)]),thickness=2,color=red)
  ],axes=frame,orientation=[66,74]):
  just4curves;
```



Add some visual aids:

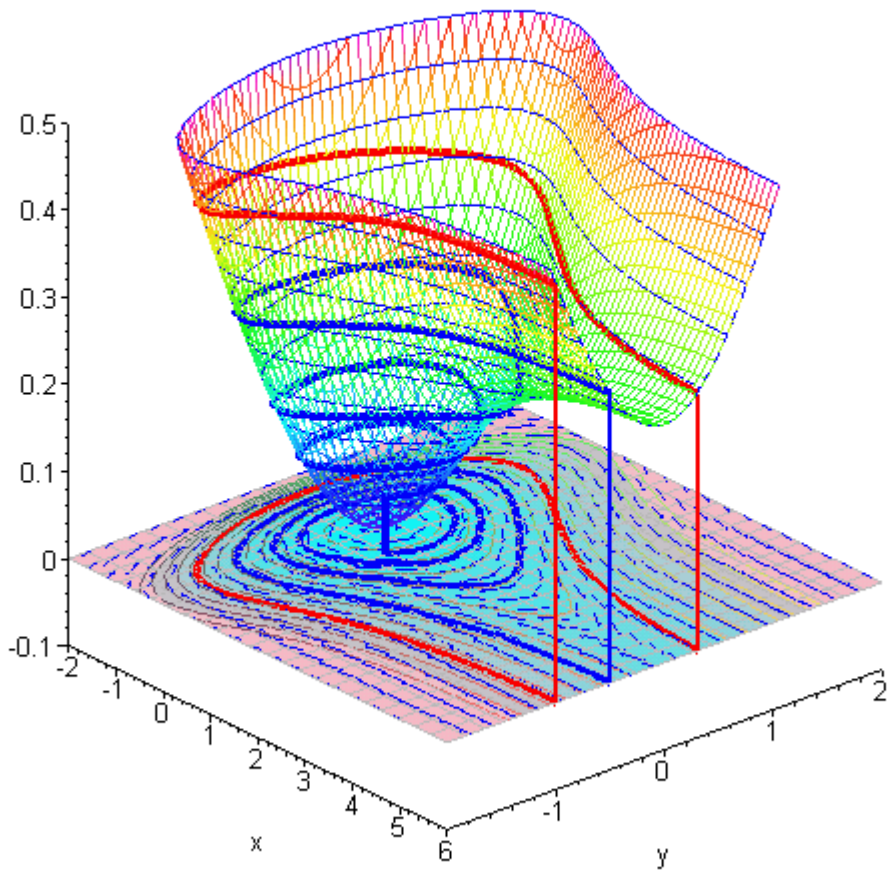
```
> ver1:=evalf(subs(ic1,[[x(0),y(0),0],llift([x(0),y(0)])])):
  ver2:=evalf(subs(ic2,[[x(0),y(0),0],llift([x(0),y(0)])])):
  ver3:=evalf([[seq(rhs(dsol1(100)[k]),k=2..3),0],llift([seq(rhs(dsol1(100)[k]),k=2..3)])]):
  ver4:=evalf([[seq(rhs(dsol2(50.5)[k]),k=2..3),0],llift([seq(rhs(dsol2(50.5)[k]),k=2..3)])]):

> display([just4curves,op(map(q->spacecurve(q),[ver1,ver2,ver3,ver4]))],
      orientation=[-64,74]);
```



The grand finale...

```
> display([
   trafo(flat),
   plot3d(V,x=-a..3*a,y=-a..a,style=wireframe,shading=ZHUE,grid=[60,48]),
   plot3d(V,x=-a..3*a,y=-a..a,style=contour,contours=[seq(k/20,k=0..10)],
    color=blue,thickness=1),
   spacecurve([3*a,y,subs(x=3*a,V)],y=-a..a,color=blue,thickness=1),
  spacecurve(ver1,color=blue,thickness=2),
  spacecurve(ver2,color=red,thickness=2),
  spacecurve(ver3,color=blue,thickness=4),
  spacecurve(ver4,color=red,thickness=2),
  spacecurve(map(tt->[seq(rhs(dsol1(tt)[k]),k=2..3),0],
  [seq(tau/10,tau=0..1000)]),thickness=4,color=blue),
  spacecurve(map(tt->[seq(rhs(dsol2(tt)[k]),k=2..3),0],
  [seq(tau/10,tau=0..1000)]),thickness=4,color=red),
  spacecurve(map(tt->llift([seq(rhs(dsol1(tt)[k]),k=2..3)]),
  [seq(tau/10,tau=0..1000)]),thickness=4,color=blue),
  spacecurve(map(tt->llift([seq(rhs(dsol2(tt)[k]),k=2..3)]),
  [seq(tau/10,tau=0..1000)]),thickness=4,color=red),
   plot3d(0,x=-a..3*a,y=-a..a,style=wireframe,color=grey,grid=[24,24])],
  view=[-a..3*a,-a..a,-0.1..0.5],axes=frame,orientation=[-41,65]);
```


0.5
0.4
0.3
0.2
0.1
0
-0.1
-2
-1
0
1
2
3
4
5
6
x
-1
0
1
2
y

>

>

>