```
> restart:with(LinearAlgebra):with(VectorCalculus):with(DEtools)
  :with(plots):with(plottools):with(stats):with(statplots):with
  (CurveFitting):
```

We will assume a DE of the form $\frac{d}{dt} x_1 = x_2, \frac{d}{dt} x_2 = f(x_1, x_2)$,
and use the van der Pol oscillator to generate data. The van der Pol oscillator can be written as $\frac{d}{dt} x_1 = x_2, \frac{d}{dt} x_2 = x_2 - x_2 x_1^2 - x_1$,
and we generate a time series by integrating it forward and extracting values.

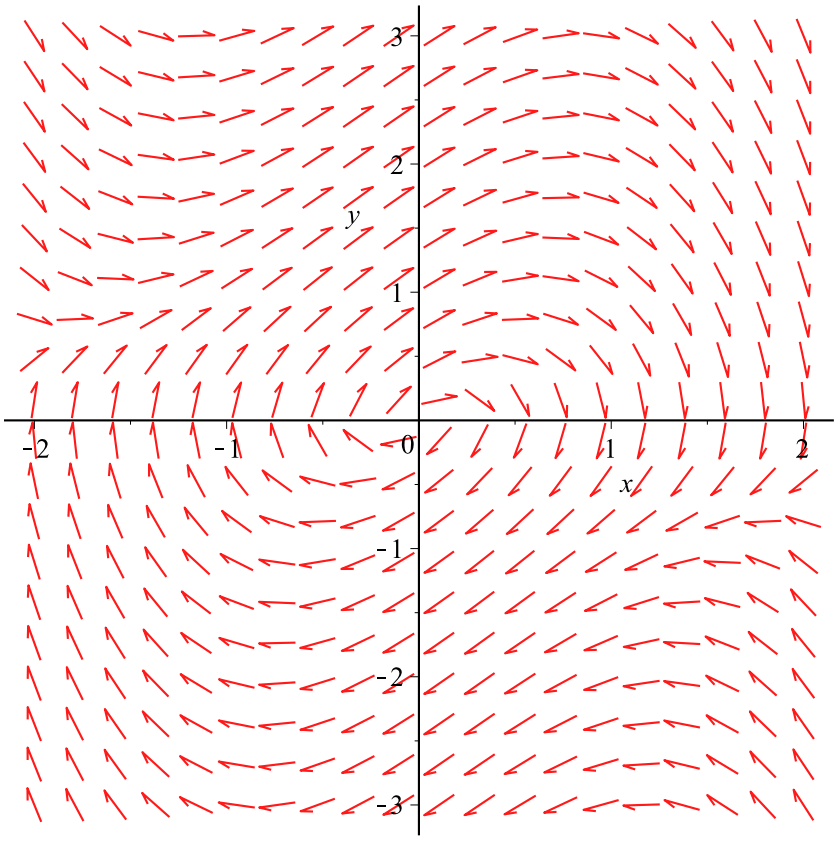
```
> kmax:=400:tmax:=kmax:
  VDP:=diff(x(t),t,t)=diff(x(t),t)-diff(x(t),t)*x(t)^2-x(t);
  VDP1:=diff(x(t),t)=y(t);
  VDP2:=diff(y(t),t)=y(t)*(1-x(t)^2)-x(t);
```

$$VDP := \frac{d^2}{dt^2} x(t) = \frac{d}{dt} x(t) - \left(\frac{d}{dt} x(t) \right) x(t)^2 - x(t)$$

$$VDP1 := \frac{d}{dt} x(t) = y(t)$$

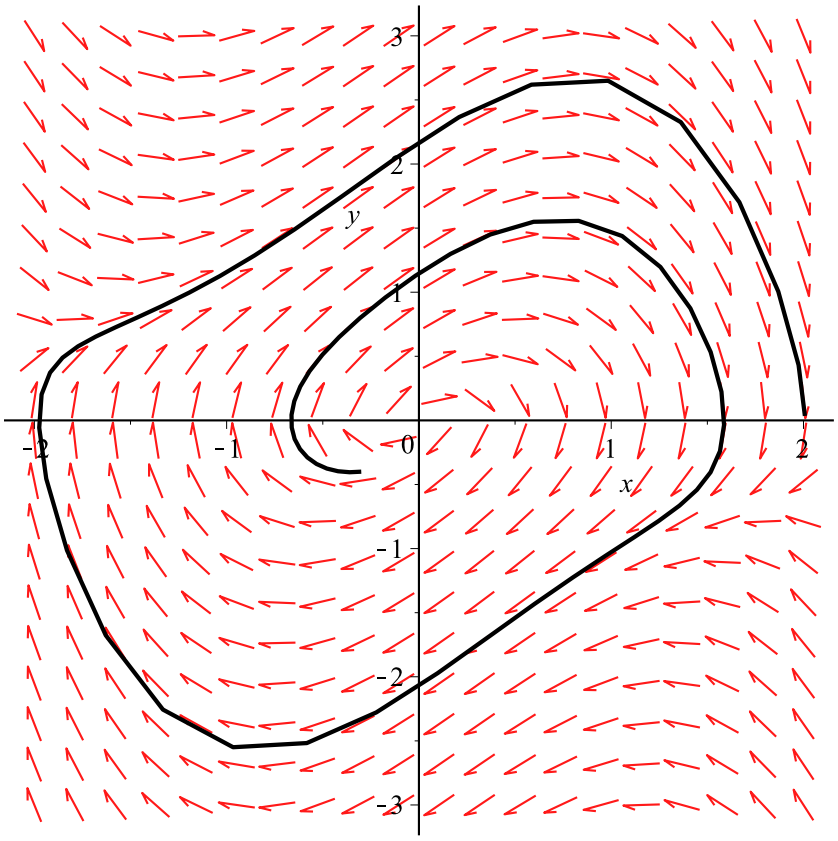
$$VDP2 := \frac{d}{dt} y(t) = y(t) \left(1 - x(t)^2\right) - x(t) \qquad (1)$$

```
> dfieldplot([VDP1,VDP2],[x(t),y(t)],t=0..tmax,x=-2..2,y=-3..3);
```




```
> AA:=phaseportrait([VDP1,VDP2],[x(t),y(t)],t=0..tmax,[[x(0)=-0.3,y
  (0)=-0.4]],x=-2..2,y=-3..3,linecolor=black,thickness=2,stepsize=
  0.15);display(AA);
```

$$AA := PLOT(\dots)$$



The initial conditions

```
> VDPIC:={x(0)=-0.3,y(0)=-0.4};
```

$$VDPIC := \{x(0) = -0.3, y(0) = -0.4\} \quad (2)$$

The system written as two first order differential equations

```
> VDPsys:={diff(x(t),t)=y(t),diff(y(t),t)=y(t)*(1-x(t)^2)-x(t)};
```

$$VDPsys := \left\{ \frac{d}{dt} x(t) = y(t), \frac{d}{dt} y(t) = y(t) \left(1 - x(t)^2\right) - x(t) \right\} \quad (3)$$

Now we set up a procedure to numerically compute the data and store it in a list:

```
> VDPsol:=dsolve(VDPsys union VDPIC, numeric, output=listprocedure,
  range=0..tmax,maxfun=0);
```

$$VDPsol := [t = \mathbf{proc}(t) \ \ldots \ \mathbf{end\ proc}, x(t) = \mathbf{proc}(t) \ \ldots \ \mathbf{end\ proc}, y(t) = \mathbf{proc}(t) \quad (4)$$

...

end proc]

Let the variable x(t) hold the x data and y(t) hold the numeric y data

```
> VDPx := subs(VDPsol,x(t)): VDPy := subs(VDPsol,y(t)):
```

```
> VDPx(1), VDPy(1);
```

$$-0.638431464897280,\ -0.172688426477271 \qquad (5)$$

Here we actually compute the data and store it, with initial time t=1/2 and incrementing in steps of 1/2

```
> VDPXA:=Array([seq( VDPx(k/10) , k = 0..kmax)]):
  VDPYA:=Array([seq( VDPy(k/10) , k = 0..kmax)]):
  VDPXX:=convert(VDPXA,list):VDPYY:=convert(VDPYA,list):
```

Write the data to a file, then read it in:

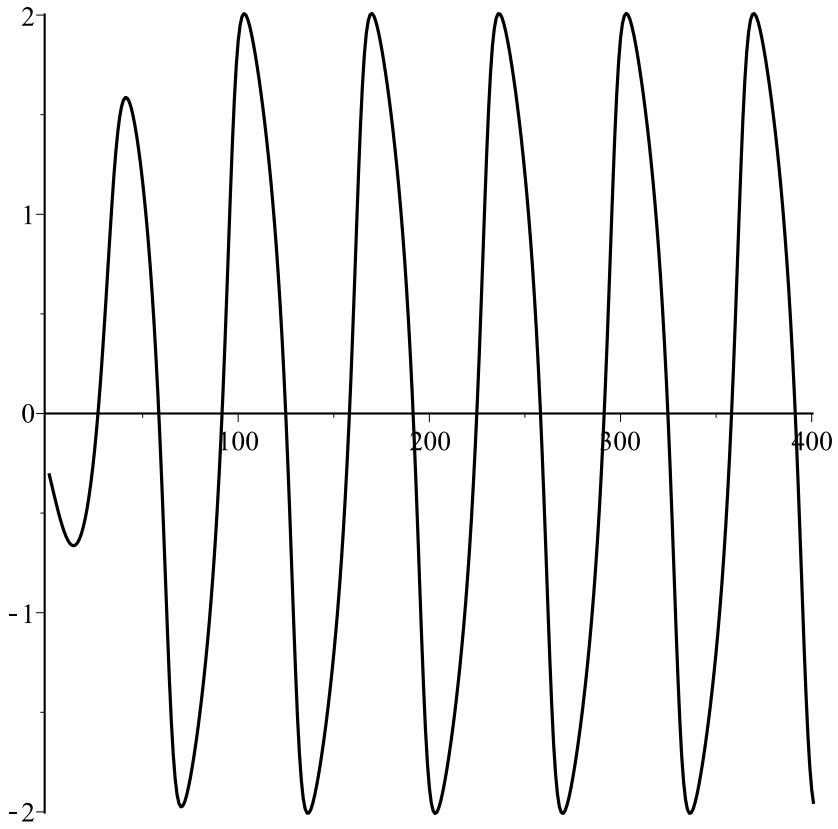
```
> writedata("c:/VDP.dat",VDPXX,float);
  L:=readdata("C:/VDP.dat",1):n:=nops(L);
```

$$n := 401 \qquad (6)$$

Plot it

```
> LPA:=listplot(L);
  display(LPA);
```

$$LPA := PLOT(\ldots)$$



```
> f := k -> L[k];
```

$$f := k \rightarrow L_k \tag{7}$$

The delay is given

```
> dly:=8;
```

$$dly := 8 \tag{8}$$

Here we form a three dimensional delay vector. We start at one more than two times the delay so that we don't try to sample before the beginning of the data set. P, N and Q are the delay vectors.

```
> for a from 2*dly+1 to nops(L) do P[a]:=f(a) od:
  for a from 2*dly+1 to nops(L) do N[a]:=f(a-dly) od:
  for a from 2*dly+1 to nops(L) do Q[a]:=f(a-2*dly) od:
```

```
> Pa:=convert(P,list):Na:=convert(N,list):Qa:=convert(Q,list):nops
  (Pa);nops(Na);nops(Qa);
  whattype(Pa);
```

$$385$$

$$385$$

$$385$$

$$list \tag{9}$$

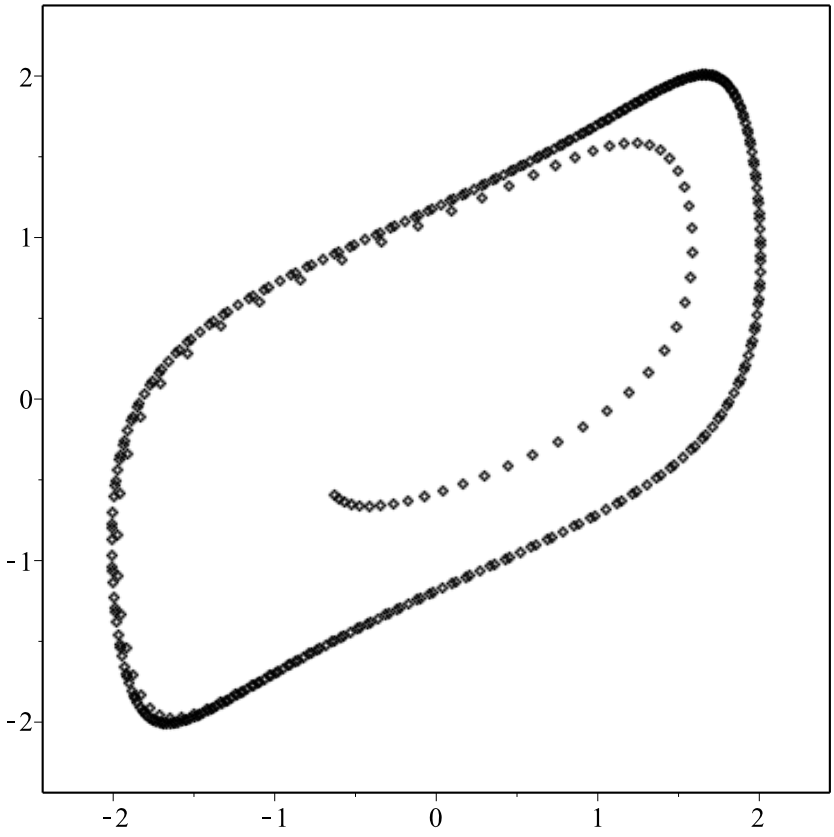
```
> Pa[3];
```

$$-0.5687727611 \tag{10}$$

A scatterplot of the data in a 2D phase space

```
> VDPP1:=scatterplot(Pa,Na,axes=boxed,thickness=3,color=black);
  display(VDPP1);
```

$$VDPP1 := PLOT(\ldots)$$



This array holds the data and the numbers 0 through kmax for spline fitting

```
> PT:=Array([seq( [k,Pa[k]] , k = 1..300)]):
  NT:=Array([seq( [k,Na[k]] , k = 1..300)]):
```

```
> PT[4][2];
```

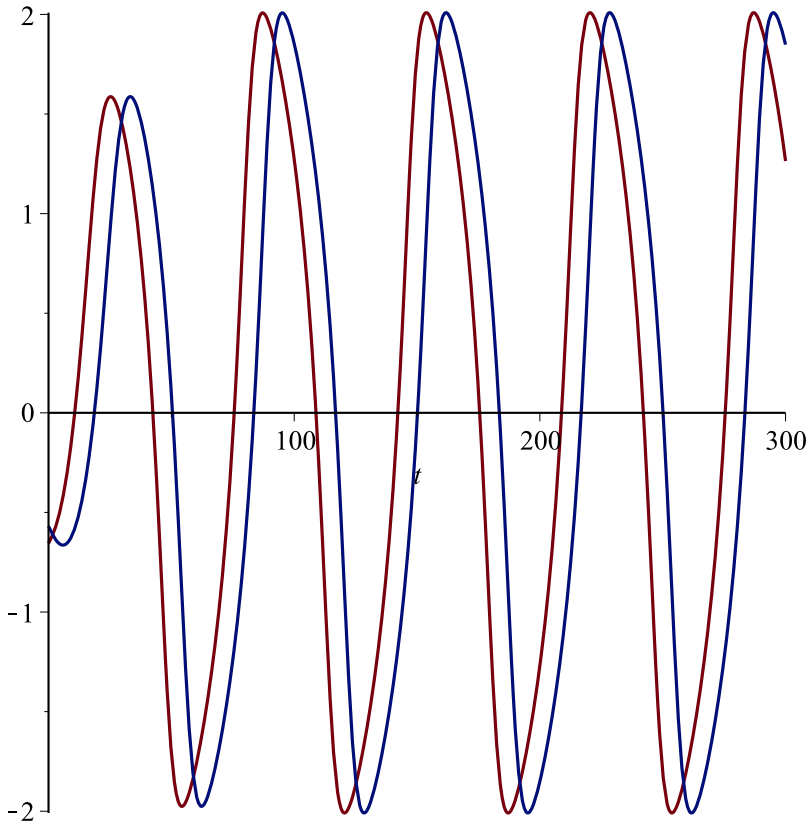
$$-0.5261427422 \qquad (11)$$

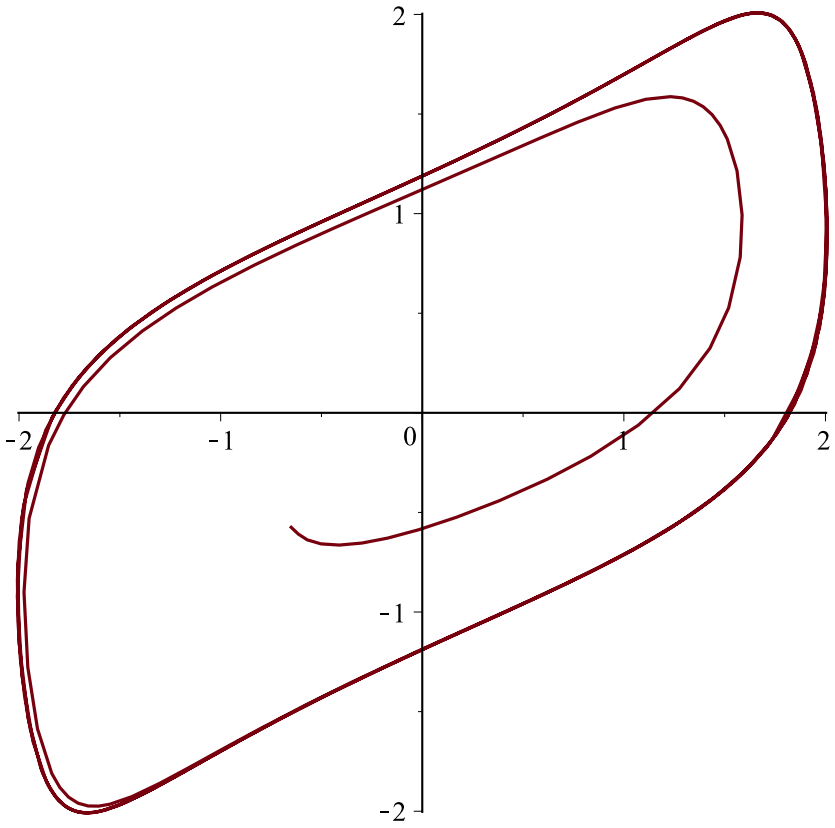
Spline it up

```
> XS:=Spline(PT, t):
  YS:=Spline(NT, t):
  PA:=plot({XS,YS},t=0..300);
  PPA:=plot([XS,YS,t=0..300]);
  display(PA);display(PPA);
```

$$PA := PLOT(\ldots)$$

$$PPA := PLOT(\ldots)$$


2
1
0
-1
-2
100
200
300
t

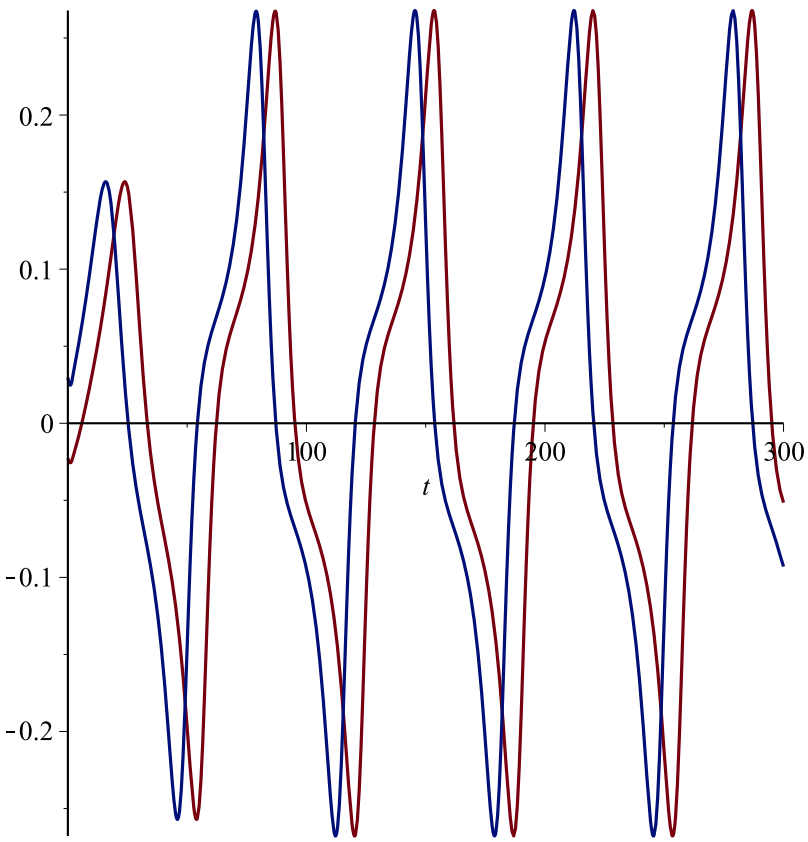


Take the derivative of the spline, and we have the beginnings of a vector field... hmmm, I'm a little suspicious of the green plot. Is that a decent approximation of the true dynamics?

```
> XT:=diff(XS,t):
  YT:=diff(YS,t):
  DYDX:=YT/XT:
```

```
> PB:=plot({XT,YT},t=0..300);
  display(PB);
```

$$PB := PLOT(\dots)$$



We can extract values from the derivative of the spline this way:

```
> eval(XT,t=.999);
```

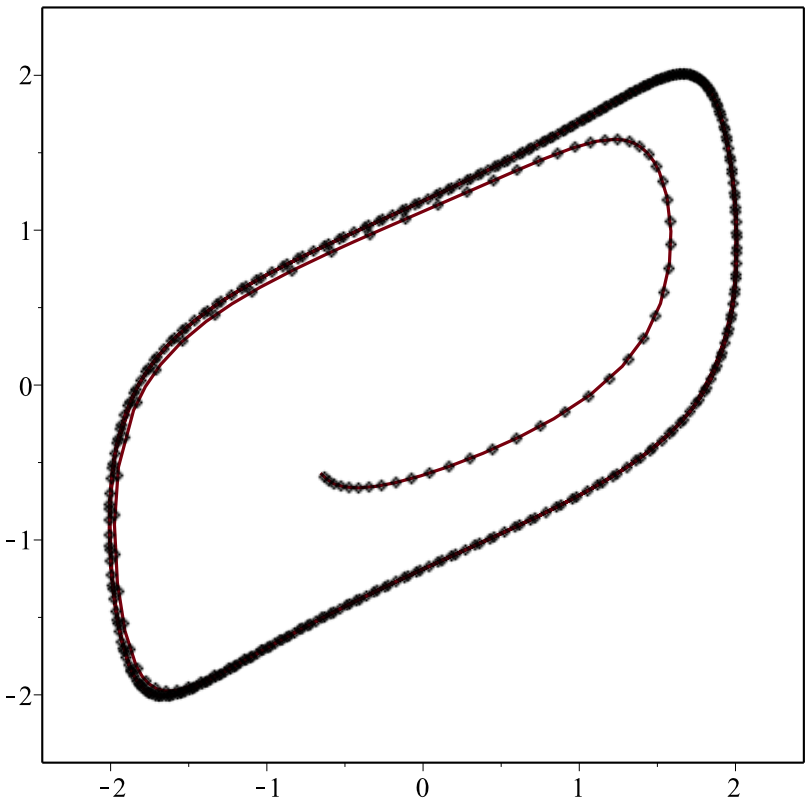
0.02471030461 **(12)**

Now we form arrays of the data from the spline fits and the derivatives of the spline fits

```
> VDPXS:=Array([seq( eval(XS,t=k+1.001) , k = 0..100)]):
  VDPYS:=Array([seq( eval(YS,t=k+1.001) , k = 0..100)]):
  VDPXT:=Array([seq( eval(XT,t=k+1.001) , k = 0..100)]):
  VDPYT:=Array([seq( eval(YT,t=k+1.001) , k = 0..100)]):
  SVDP:=VDPYT/VDPXT:
  SVDP[11];
  LVDPXS:=convert(VDPXS,list):
  LVDPYS:=convert(VDPYS,list):
  LVDPXT:=convert(VDPXT,list):
  LVDPYT:=convert(VDPYT,list):
  LSVDP:=convert(SVDP,list):
```

```
display(PPA,VDPP1);
```

0.3239899556



This is the least squares fit to the derivative data

```
> XVDPdata:=LVDPXS:
  YVDPdata:=LVDPYS:
  ZVDPdata:=LSVDP:
  ZVDPXdata:=LVDPXT:
  ZVDPYdata:=LVDPYT:
  poly:=z=aa*x^3+bb*x^2*y+cc*x*y^2+dd*y^3+ee*x*y+ff*x^2+gg*y^2+hh*
  x+yi*i+jj;
  vars:={aa,bb,cc,dd,ee,ff,gg,hh,ii,jj};
```

$$poly := z = aa\,x^3 + bb\,x^2\,y + cc\,x\,y^2 + dd\,y^3 + ee\,x\,y + ff\,x^2 + gg\,y^2 + hh\,x + yi\,i + jj$$

$$vars := \{aa, bb, cc, dd, ee, ff, gg, hh, ii, jj\} \quad \textbf{(13)}$$

```
> FVDP:=fit[leastsquare[[x,y,z], poly,vars ]]([XVDPdata, YVDPdata,
  ZVDPdata]);
```

$$FVDP := z = 0.2842466593\,x^3 + 1.021916265\,x^2\,y - 1.571684531\,x\,y^2 + 0.3952584471\,y^3 - 0.5147300762\,x\,y + 1.681618155\,x^2 - 1.107149987\,y^2 - 0.1211295753\,x + 0.7156421235 \quad (14)$$

```
> FVDPX:=fit[leastsquare[[x,y,z], poly, vars]]([XVDPdata, YVDPdata,
  ZVDPXdata]);
```

$$FVDPX := z = 0.01978730421\,x^3 - 0.1381802407\,x^2\,y + 0.2298156170\,x\,y^2 - 0.1208800678\,y^3 + 0.04381634620\,x\,y - 0.02467853773\,x^2 - 0.04345841020\,y^2 + 0.01358759149\,x + 0.06923855855 \quad (15)$$

```
> FVDPY:=fit[leastsquare[[x,y,z], poly, vars]]([XVDPdata, YVDPdata,
  ZVDPYdata]);
```

$$FVDPY := z = 0.02070014256\,x^3 - 0.02241201150\,x^2\,y + 0.06155748164\,x\,y^2 - 0.05894100635\,y^3 + 0.01077289793\,x\,y - 0.006316080816\,x^2 - 0.01069447370\,y^2 + 0.05550762709\,x + 0.01723321335 \quad (16)$$

```
> FVDPS:={diff(X(t),t)=Y(t),diff(Y(t),t)=subs({x=X(t),y=Y(t)},rhs
  (FVDP))};
  DFVDPX:=diff(X(t),t)=subs({x=X(t),y=Y(t)},rhs(FVDPX));
  DFVDPY:=diff(Y(t),t)=subs({x=X(t),y=Y(t)},rhs(FVDPY));
```

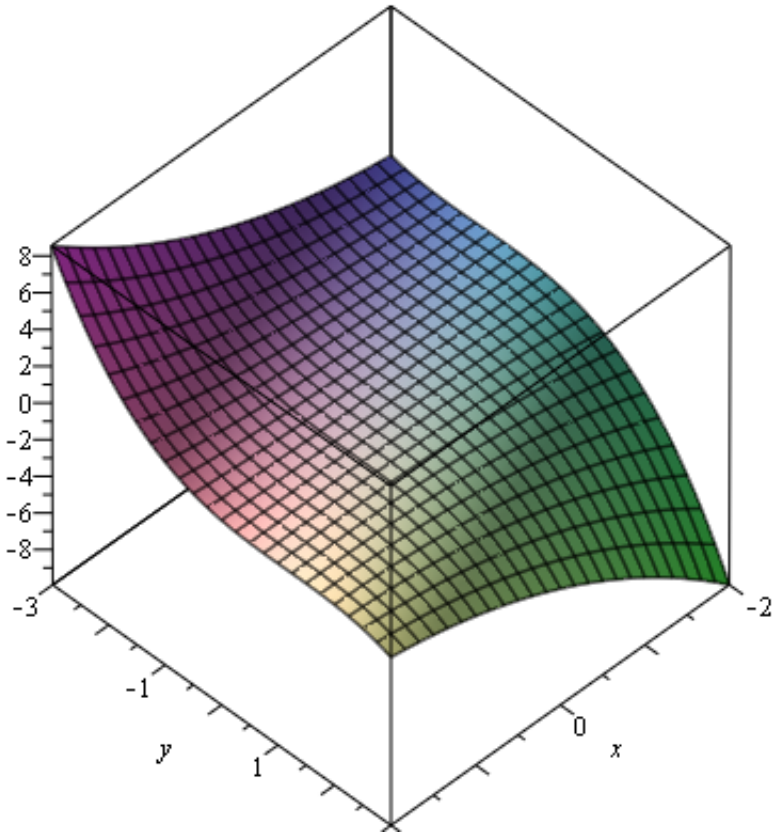
$$FVDPS := \left\{ \frac{d}{dt}X(t) = Y(t),\ \frac{d}{dt}Y(t) = 0.2842466593\,X(t)^3 + 1.021916265\,X(t)^2\,Y(t) - 1.571684531\,X(t)\,Y(t)^2 + 0.3952584471\,Y(t)^3 - 0.5147300762\,X(t)\,Y(t) + 1.681618155\,X(t)^2 - 1.107149987\,Y(t)^2 - 0.1211295753\,X(t) + 0.7156421235 \right\}$$

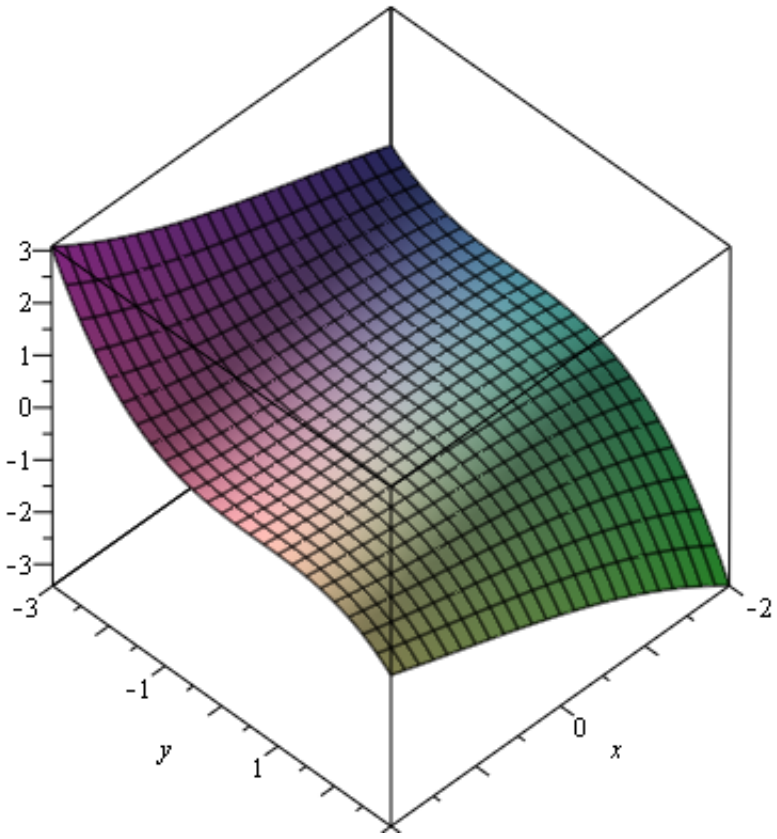
$$DFVDPX := \frac{d}{dt}X(t) = 0.01978730421\,X(t)^3 - 0.1381802407\,X(t)^2\,Y(t) + 0.2298156170\,X(t)\,Y(t)^2 - 0.1208800678\,Y(t)^3 + 0.04381634620\,X(t)\,Y(t) - 0.02467853773\,X(t)^2 - 0.04345841020\,Y(t)^2 + 0.01358759149\,X(t) + 0.06923855855$$

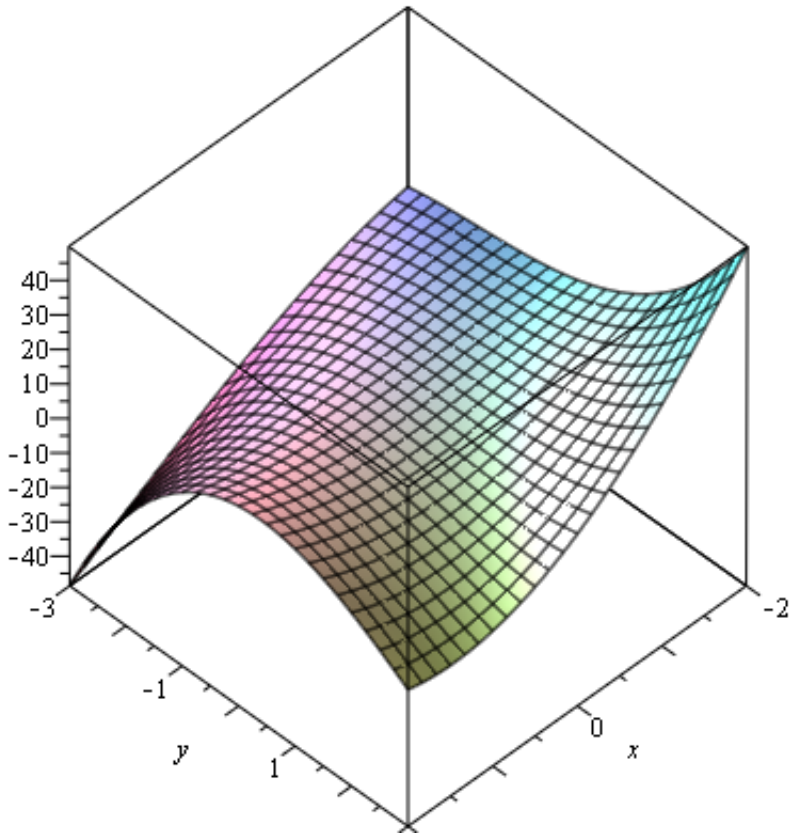
$$DFVDPY := \frac{d}{dt}Y(t) = 0.02070014256\,X(t)^3 - 0.02241201150\,X(t)^2\,Y(t) + 0.06155748164\,X(t)\,Y(t)^2 - 0.05894100635\,Y(t)^3 + 0.01077289793\,X(t)\,Y(t) - 0.006316080816\,X(t)^2 - 0.01069447370\,Y(t)^2 + 0.05550762709\,X(t) + 0.01723321335 \quad (17)$$

```
> plot3d(rhs(FVDPX),x=-2..2,y=-3..3,axes=boxed);
  plot3d(rhs(FVDPY),x=-2..2,y=-3..3,axes=boxed);
```

```
plot3d(rhs(FVDP),x=-2..2,y=-3..3,axes=boxed);
```



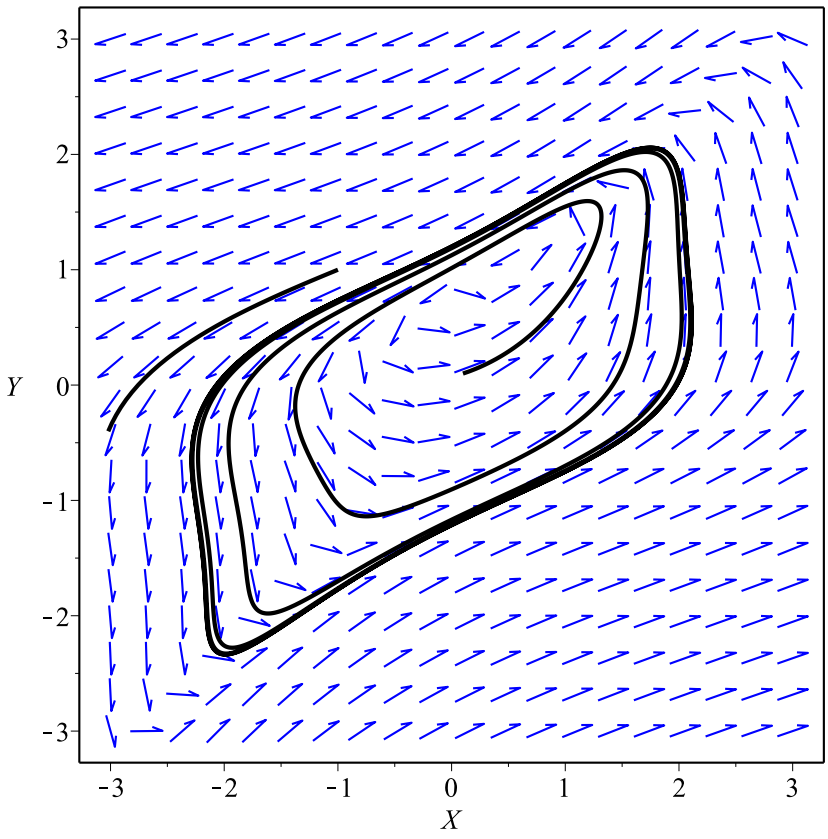

3
2
1
0
-1
-2
-3
-3
-1
y
1
0
x
-2




```
> PRD:=DEplot([DFVDPX,DFVDPY],[X(t),Y(t)],t=0..tmax,[[X(0)=0.1,Y(0)
  =0.1],[X(0)=-1,Y(0)=1]],X=-3..3,Y=-3..3,color=blue,linecolor=
  black,thickness=2,axes=boxed,stepsize=0.1);
```

$$PRD := PLOT(\dots) \qquad (18)$$

```
> display(PRD);
```


3
2
1
Y 0
-1
-2
-3
-3
-2
-1
0
1
2
3
X