Content

This is an example of a MAPLE 7 worksheet for normalization, finding invariants and reduction of the following system:
Dx = Mx + f_2(x) + higher order terms , where:

$$M := \begin{bmatrix} -\alpha_1 & \alpha_1 - \alpha_2 & \alpha_2 \\ -2\,\alpha_1 & \alpha_1 - \alpha_2 - \alpha_3 & \alpha_2 + \alpha_3 \\ -2\,\alpha_1 - \alpha_2 + \alpha_3 & \alpha_1 - 2\,\alpha_3 - \alpha_2 & \alpha_2 + \alpha_3 \end{bmatrix}$$

f[2] := vector([x[1]^2, 0, 0])

Disclaimer

While our testing, as well as computations of examples, indicate correctness and reliability of the programs, the authors cannot guarantee the correctness of any routine.
Program developed by S. Mayer (email: mayer@mathA.rwth-aachen.de)

Initialization

```
with(linalg):
```

The Message: "Warning, the protected names norm and trace have been redefined and unprotected" will not inflict the following calculations. It derives from using the linalg package instead of the LinearAlgebra package, for the "linalg package is useful for doing computations in abstract linear algebra" (Maple glossary). This allows parameter dependent calculations.

Primitive operations

Procedures

Input

The vector field v is given in the form v(x)=Mx+f[2][x]+... where M is an nxn matrix and the f[j] are homogeneous polynomials C^n -> C^n of degree j. The following routines are capable of working with parameters, so you might want to apply them for parameter dependent vector fields. Note that maple assumes non-degeneraces of all parameters.

Enter the linearization M of the vector field. The following calculations assume that M is semisimple.

```
> M:=matrix(3,3,[0,alpha[1],alpha[2],-alpha[1],0,alpha[3],-alph
  a[2],-alpha[3],0]):
  T:=matrix(3,3,[1,0,0,1,1,0,1,1,1]):
  M:=multiply(T,M,inverse(T));
```

$$M := \begin{bmatrix} -\alpha_1 & \alpha_1 - \alpha_2 & \alpha_2 \\ -2\,\alpha_1 & \alpha_1 - \alpha_2 - \alpha_3 & \alpha_2 + \alpha_3 \\ -2\,\alpha_1 - \alpha_2 + \alpha_3 & \alpha_1 - 2\,\alpha_3 - \alpha_2 & \alpha_2 + \alpha_3 \end{bmatrix}$$

Let targetspace be the desired degree of the normal form. targetspace then is also the degree of the taylor expansion of the vector field.

```
> targetspace:=2;
```

$$targetspace := 2$$

Just evaluate the following lines.

```
> f:=vector(targetspace,0):
> dim:=rowdim(M);
```

$$dim := 3$$

```
> for i from 1 to targetspace do
>     f[i]:=vector(dim,0);
> od:
> f[1]:=Matrix2Polynom(M);
```

$$f_1 := [-\alpha_1 x_1 + \alpha_1 x_2 - \alpha_2 x_2 + \alpha_2 x_3, -2\,\alpha_1 x_1 + \alpha_1 x_2 - \alpha_2 x_2 - \alpha_3 x_2 + \alpha_2 x_3 + x_3 \alpha_3,$$
$$-2\,\alpha_1 x_1 - \alpha_2 x_1 + \alpha_3 x_1 + \alpha_1 x_2 - 2\,\alpha_3 x_2 - \alpha_2 x_2 + \alpha_2 x_3 + x_3 \alpha_3]$$

Enter the higher order terms of the Taylor expansion of the vector field: f[2]:=vector([...]); f[3]:=... f[targetspace]:=vector([...]);

```
> f[2]:=vector([x[1]^2,0,0]);
```

$$f_2 := [x_1^{\,2}, 0, 0]$$

Enter the maximal degree of calculated invariant polynomials.

```
> invariantsdegree:=1;
```

$$invariantsdegree := 1$$

Automatic initialization

```
> dim:=rowdim(M):
> minimalpoly:=minpoly(M,tau);
```

$$minimalpoly := (\alpha_2^{\,2} + \alpha_1^{\,2} + \alpha_3^{\,2})\,\tau + \tau^3$$

```
> sigmaset:=symmetriceigenvalues(minimalpoly,dim):
```

Determining annihilating polynomials

Annihilating polynomials for the action of ad M on P_r: (Remember that x -> p(-x) annihilates the action of ad M on P_0 if p annihilates M.)

```
> liebracketannihilator:=AnnihilatingPolynomials(sigmaset,subs(
  tau=-tau,minimalpoly),minimalpoly,dim,targetspace+1):
"Finished with step ", 1, "."
"Finished with step ", 2, "."
"Finished with step ", 3, "."
```

Now liebracketannihilator[j] annihilates the action of ad M on P_{j-1}.

```
> invpoly:=AnnihilatingPolynomials(sigmaset,minimalpoly,minimal
  poly,dim,invariantsdegree):
"Finished with step ", 1, "."
```

Now invpoly[j] annihilates the action of L_M on S_j.

Output

```
> for j from 1 to targetspace+1 do
>     print("Annihilating polynomial for ad M on homogeneous
  polynomials of degree",j-1);
>     print(liebracketannihilator[j]);
> od;
```

"Annihilating polynomial for ad M on homogeneous polynomials of degree", 0

$$\tau\,(\alpha_2^{\,2} + \alpha_1^{\,2} + \alpha_3^{\,2} + \tau^2)$$

"Annihilating polynomial for ad M on homogeneous polynomials of degree", 1

$$(4\,\alpha_2^{\,2}+4\,\alpha_3^{\,2}+\tau^2+4\,\alpha_1^{\,2})\,\tau\,(\alpha_2^{\,2}+\alpha_1^{\,2}+\alpha_3^{\,2}+\tau^2)$$

"Annihilating polynomial for ad M on homogeneous polynomials of degree", 2

$$(4\,\alpha_2^{\,2}+4\,\alpha_3^{\,2}+\tau^2+4\,\alpha_1^{\,2})(\tau^2+9\,\alpha_3^{\,2}+9\,\alpha_2^{\,2}+9\,\alpha_1^{\,2})\,\tau\,(\alpha_2^{\,2}+\alpha_1^{\,2}+\alpha_3^{\,2}+\tau^2)$$

```
> print("Annihilating polynomial for L_M on homogeneous
  polynomials of degree",0);print(1);
  for j from 1 to invariantsdegree do
>     print("Annihilating polynomial for L_M on homogeneous
  polynomials of degree",j);
>     print(invpoly[j]);
> od;
```

"Annihilating polynomial for L_M on homogeneous polynomials of degree", 0

$$1$$

"Annihilating polynomial for L_M on homogeneous polynomials of degree", 1

$$\tau\,(\alpha_2^{\,2}+\alpha_1^{\,2}+\alpha_3^{\,2}+\tau^2)$$

Higher-order normalization

```
> dgl:=matrix(targetspace,targetspace): #the differential
  equation in the various transformed states
> Trafo:=vector(targetspace):Trafo[1]:=vector(dim,0):   #the
  transformation is exp(Trafo).
> for i from 1 to targetspace do
>     dgl[1,i]:=eval(f[i]);
> od:
> for i from 2 to targetspace do
      #print("starting with
  ",collect(expand(liebracketannihilator[i+1]),tau));
>
  erg:=semisimplecalculation(M,collect(expand(liebracketannihil
  ator[i+1]),tau),f[i]);
      #print("transformation and transformed part in normal
  form calculated for one degree higher");
>     for j from 1 to i-1 do
>         dgl[i,j]:=eval(dgl[i-1,j]);
>     od;
      #print("some terms are copied (degree < r)");
>     Trafo[i]:=eval(erg[1]);
>     dgl[i,i]:=eval(erg[2]);
      #print("some terms are chosen as it was calculated
  before");
>     for j from i+1 to targetspace do
>         dgl[i,j]:=eval(dgl[i-1,j]);
>         k:=1;
>         while j >= k*(i-1)+1 do
>               hlp:=scalarmul(dgl[i-1,j-k*(i-1)],1/k!);
>               for l from 1 to k do
>                   hlp:=lie(erg[1],hlp);
>               od;
>               dgl[i,j]:=matadd(dgl[i,j],hlp);
>               k:=k+1;
>         od;
```

```
            #print("transformation of higher terms up to degree
  ",j," done in this step")
>     od;
>     for j from i to targetspace do
>         f[j]:=eval(dgl[i,j]);
>     od;
      #print("transformation succesfull up to degree ",i);
> od:
```

The The previous lines compute Poincare-Dulac normal form up to degree targetspace. It is stored in the variable dgl. dgl is a field of dimension targetspace^2: dgl[i,1]+dgl[i,2]+...+dgl[i,targetspace] is the Poincare-Dulac-normal form up to degree i. The corresponding transformation is given by exp(ad(Trafo[i])) exp(ad(Trafo[i-1])) ... exp(ad(Trafo[2])). An output of the results is provided in the next steps:

+ ***Output of the Poincare-Dulac normal form***

+ ***Output of intermediate results***

− Invariants at fixed degree

```
> eval(invariantsdegree);
```

$$1$$

Choose a degree invdegree <= invariantsdegree;

```
> invdegree:=invariantsdegree;
```

$$invdegree := 1$$

− *Invariant providing map*

A map from the space of homogeneous polynomials of invdegree onto the invariant polynomials of invdegree is calculated.

```
> eval(invpoly);
```

$$[\tau\,(\alpha_2^{\,2}+\alpha_1^{\,2}+\alpha_3^{\,2}+\tau^2)]$$

Note that the command InvariantsProvidingPolynomial(invpoly[deg],M); calculates the invariant providing map for degree deg, if deg<invariantsdegree.

```
> invpolymap:=InvariantsProvidingPolynomial(invpoly[invdegre
  e],M);
```

$$invpolymap := \alpha_2^{\,2}+\alpha_1^{\,2}+\alpha_3^{\,2}+\tau^2$$

− *Finding the invariants*

A basis of invariants of degree ’invariantsdegree’ is calculated using ’invpolymap’. If you already know the dimension of the space of invariants from some theoretical argument, set ’number’ to some positive value (in order to save calculation time) and thus only do ’number’ tries for invariants. Another call of invariants(...,eval(thistry),...) will then try more invariants...

```
> Invarianten:={}: InvZahl:=0: firsttry:=vector(dim,0):
> firsttry[dim]:=invdegree:number:=-1:eval(firsttry);
```

$$[0,\,0,\,1]$$

```
> thistry,Invarianten,InvZahl:=invariants(invpolymap,eval(In
  varianten),InvZahl,eval(firsttry),number,invdegree,dim,M);
```

$$thistry,\ Invarianten,\ InvZahl := [2,\,0,\,0],\ \{x_3\,\alpha_1^{\,2}+\alpha_1\,x_3\,\alpha_3-\alpha_1\,\alpha_2\,x_3-\alpha_1^{\,2}\,x_2-\alpha_3\,\alpha_2\,x_2 + \alpha_2^{\,2}\,x_2-\alpha_3\,\alpha_1\,x_2+\alpha_2\,\alpha_1\,x_1+\alpha_3\,\alpha_1\,x_1+\alpha_3^{\,2}\,x_1-\alpha_2^{\,2}\,x_1\},\ 1$$

If g is an invariant, and c a constant, c*g is also an invariant. So it is ’legal’ to try to simplify further calculations by eliminating parameters.

```
> Inva:=vector(InvZahl):
```

```
> for i from 1 to InvZahl do

  Inva[i]:=factor(polynomialsimplify(Invarianten[i],dim));
  od;
```

$$Inva_1 := \alpha_1 x_2 - \alpha_1 x_3 - \alpha_2 x_1 - \alpha_3 x_1 + \alpha_2 x_2$$

Reduced vector field

```
> Inva1:=-Inva[1]:
> Inva2:=factor(lieableitung(f[2],Inva1)):
> Inva3:=factor(lieableitung(f[2],Inva2)):
> pbasis:=vector([eval(Inva1^3),eval(Inva1*Inva2)]):
> LK:=LinearCoefficients(pbasis,Inva3,dim);
```

$$LK := \left[-2 \frac{(\alpha_2^{\,2} - \alpha_2 \alpha_3 + \alpha_1^{\,2})(\alpha_2 + \alpha_3)\alpha_3^{\,3}}{(\alpha_2^{\,2} + \alpha_1^{\,2} + \alpha_3^{\,2})^4}, 2 \frac{\alpha_3}{\alpha_2^{\,2} + \alpha_1^{\,2} + \alpha_3^{\,2}} \right]$$

```
>
```

The reduced equation is given by
Dx1=x2
Dx2=LK[1]*x1^3 + LK[2] * x1 * x2