Chapter 1

Desolv Documentation

The **Desolv** package is implemented in the symbolic software, Maple. **Desolv** is simply a solver which attempts to find the solutions of a linear or nonlinear system of overdetermined differential equations of polynomial type. The implementation of **Desolv** is based on a heuristic procedure, which does not always promise to produce complete solutions of a system. If the heuristic procedure fails to simplify a system completely, it will transform the system of unsolved equations into a standard form. The heuristics is made up of seven modules which are the one-term module, the direct separation module, the solving unknown module, the indirect separation module, the integration module, the solving ODE module and the decoupling module; these have been described in Chapter Three.

The development of **Desolv** is based on the symbolic package **LIE** written by Alan Head [**?**] in MuMATH; **LIE** is a solver for systems of linear PDEs. **Desolv** has a simpler structure than **LIE**. **Desolv** partly adopts the methods of separation and integration in **CRACK** [**?**] (Thomas Wolf and Andre Brand) written in REDUCE and the method of decoupling in both **standard form** [**?**] (Gregory Reid and Allan Wittkopf) and **diffgrob2** [**?**] (Elizabeth Mansfield), which are both written in Maple. **Desolv** is designed to handle large systems efficiently without running into memory problems.

Desolv includes other algorithms which help to analyse Lie point symmetries, non-classical symmetries and potential symmetries.

The **Desolv** package runs on Maple Version V, Release 3 or later releases.

The descriptions of main functions in **Desolv** are given in this chapter. The global parameters of **Desolv** which can be controlled by users are also described in details. Examples are used to demonstrate how to find point symmetries, non-classical symmetries and potential factors. A classification problem of point symmetries is illustrated in section (6.6). Section (6.7) shows that the function *decouple*() can work differently with and without changing the global parameters.

1.1 The Main Function pdesolv()

The function *pdesolv*() solves a system of overdetermined differential equations. It requires three arguments and outputs a list of four elements. The input and output of

pdesolv(*leqns,lfn,lvar*)

are as follows:

INPUT:
- *leqns* is a list of differential expressions
- *lfn* is a list of unknown functions
- *lvar* is a list of variables

OUTPUT: a list of 4 elements [*reqns,linq,lsol,lunk*]
- *reqns* is a list of differential expressions
- *linq* is a list of differential expressions
- *lsol* is a list of solutions of functions in *lfn*
- *lunk* is a list of unknowns occuring in *reqns, linq* and *lsol*

The equations in the list *leqns* are in linear homogeneous form. The functions and the constants in *lfn* are the unknowns in *leqns*. The variables in *lvar* are the variables on which the differential equations *leqns* depend. In the cases that **pdesolv**() does not solve the system *leqns* completely, a list of unsolved equations is returned as *reqns*. During any calculation, **pdesolv**() normally makes the assumptions that some expressions are non-zero. These expression are called inequations. If these inequations contain any unknown functions/constants, they are stored in the list *linq*. If the input list *leqns* is a linear system of differential equations, then *linq* is an empty list. The list *lsol* contains the solutions to the unknowns in *lfn*. The list *lunk* consists of unknown functions or constants appearing in *reqns*, *linq* and *lsol*. Any unknowns arising during the simplification process are denoted as **F_i** for functions and **C_k** for constants, where i and k are integers. The highest values of i and k are stored in the global variables *desolv_function* and *desolv_constant* respectively.

1.2 The Parameters in Desolv

Desolv has some parameters that allows the user to have control of the simplification process.

desolv_module - The number of modules used in the simplification process. The function *pdesolv*() consists of seven modules which are in the following order: the solving one-term module, the direct separating module, the solving unknown module, the indirect separating module, the integrating module,

the solving ordinary DE module and the decoupling module. By default, desolv_module is set to be 7. For example, if desolv_module is set to be 4, then only the first four modules are used to simplify a system of differential equations.

desolv_modstat - A list of integers which tells how many times each module is successfully applied.

desolv_Nterm - The number of terms in an equation. There are two different ways of scanning a system of differential equations through the modules. One way is to take the whole system and scan it through each module. This is called *horizontal scan*. The other way is take one equation at a time and scan it through each module. This is called *vertical scan*. *pdesolv*() splits a system into two lists, one list **S** contains equations having **desolv_Nterm** terms or less and one list **L** contains equation having more than desolv_Nterm terms. *pdesolv*() uses *vertical scan* for the list **S** and *horizontal scan* for the list **L**. The default value is 1.

desolv_function - The number of new unknown functions which are introduced into the system during the call of *pdesolv*. **desolv_function** is set to be 1 each time the function *pdesolv*() is called.

desolv_constant - The number of new unknown constants which are introduced into the system during the call of *pdesolv*(). **desolv_constant** is set to be 1 each time the function *pdesolv*() is called.

desolv_reduce - The default value is *true*. When an integrability condition has been computed, it is immediately reduced with respect to the current system. If **desolv_reduce** is set to be *false*, no reduction is done at this stage.

desolv_sublimit - The default value is 10^5. During the calculation of the decoupling module, the coefficient of the highest derivative of an equation may get very large. If the length of the coefficient is greater than **desolv_sublimit**, the coefficient is replaced by an arbitrary function which has the same dependencies as the coefficient. The arbitrary function replacement of the coefficient prevents the system from large expansion.

desolv_efactor - The expansion factor. When the solution of an unknown function or constant is found in the simplification process, its solution is substituted into equations that contain the function. As a substitution is made to an equation, the function *pdesolv*() compares the length of the equation before and after the substitution. If the equation is expanded after the substitution by a factor greater than **desolv_efactor**, then the substitution will not be done to the equation and delay until all the modules have

unsuccessfully applied to the equation. **desolv_efactor** is set to 1000 by default.

desolv_nzassume - A list of expressions that are assumed to be non-zero during the simplification process.

desolv_liassume - A list of lists of expressions that are assumed to be linearly independent of each other during the simplification process.

infolevel[pdesolv] - The setting of **infolevel**[pdesolv] causes the display of intermediate results during the calculation of the function *pdesolv*(). Higher values of **infolevel**[pdesolv] will cause more information to be displayed. Other options for **infolevel** are provided, such as, infolevel[gendef], infolevel[genvec], infolevel[comtab], infolevel[separation], infolevel[decouple].

1.3 Classical symmetries

As a prototypical example of how to find point symmetry with the help of the **Desolv** package, the linear heat equation

$$u_{xx} - u_t = 0 \tag{1.1}$$

is considered. The defining equations of the point symmetry can be generated by the function *gendef*() which requires three input arguments, as follows:

gendef(*deqns, dvar, ivar*)

INPUT:
- *deqns* is a list of differential equations
- *dvar* is a list of dependent variables
- *ivar* is a list of independent variables

OUTPUT: a list of 3 elements [*leqns, lfn, lvar*]
- *leqns* is a list of homogeneous differential equations
- *lfn* is a list of infinitesimal functions
- *lvar* is a list of variables

The differential equations in *deqns* usually have the highest derivatives on the left hand side (lhs). If the highest derivatives are not on the left hand sides of the equations, *gendef*() will automatically put the equations into the solved form for the highest derivatives. The system of defining equations *leqns* are in the homogeneous form. The system *leqns* has been reduced with respect to its one-term equations. The list *lfn* contains infinitesimal functions corresponding to the variables in *lvar*. The infinitesimal function of an independent variable x is denoted by **xi**[x] in maple-syntax whereas the infinitesimal function of a

dependent variable u is denoted by **eta**$[u]$. In the MapleV Release 3 and later versions, the name **xi**[**x**] is displayed as ξ_x. The symmetry generator of the heat equation is expressed as

$$v = xi[x]\frac{\partial}{\partial x} + xi[t]\frac{\partial}{\partial t} + eta[u]\frac{\partial}{\partial u} \tag{1.2}$$

Example: Generation of the defining equations for the linear heat equation:

```
> read desolv:
> de := [ diff(u(x,t),x,x) - diff(u(x,t),t) = 0 ];
```

$$de := \left[\left(\frac{\partial^2}{\partial x^2}u(x,t)\right) - \left(\frac{\partial}{\partial t}u(x,t)\right) = 0\right]$$

```
> le := gendef(de,[u],[x,t]);
```

$$\begin{aligned} le := \Bigg[\Bigg[& \frac{\partial}{\partial u}\xi_x(x,t,u), \frac{\partial}{\partial u}\xi_t(x,t,u), \frac{\partial}{\partial x}\xi_t(x,t,u), \frac{\partial^2}{\partial u^2}\eta_u(x,t,u), \\ & -2\left(\frac{\partial}{\partial x}\xi_x(x,t,u)\right) + \left(\frac{\partial}{\partial t}\xi_t(x,t,u)\right), -\left(\frac{\partial}{\partial t}\eta_u(x,t,u)\right) + \left(\frac{\partial^2}{\partial x^2}\eta_u(x,t,u)\right), \\ & 2\left(\frac{\partial^2}{\partial u \partial x}\eta_u(x,t,u)\right) + \left(\frac{\partial}{\partial t}\xi_x(x,t,u)\right) - \left(\frac{\partial^2}{\partial x^2}\xi_x(x,t,u)\right)\Bigg], \\ & [\xi_x(x,t,u), \xi_t(x,t,u), \eta_u(x,t,u)], [x,t,u]\Bigg] \end{aligned}$$

This system of overdetermined linear PDEs can be simplified by the function *pdesolv*(). The function *pdesolv*() returns one unsolved equation. There is no assumption made on any unknown functions or constants in the solution. The solution contains one unknown function $F\_4(x,t)$ and six unknown constants, C_1 to C_6.

Example: (Heat equation)

```
> sol := pdesolv(le[1],le[2],le[3]);
```

$$\begin{aligned} sol := \Bigg[\Bigg[& -\left(\frac{\partial}{\partial t}F\_4(x,t)\right) + \left(\frac{\partial^2}{\partial x^2}F\_4(x,t)\right)\Bigg], [\,], \\ & \Bigg[\xi_x(x,t,u) = \frac{1}{2}xC\_4 + xtC\_5 + \frac{1}{2}C\_1 + \frac{1}{2}tC\_2, \end{aligned}$$

$$\xi_t(x,t,u) = C\_3 + tC\_4 + t^2C\_5,$$
$$\eta_u(x,t,u) = F\_4(x,t) - \frac{1}{4}uC\_5x^2 - \frac{1}{4}uC\_2x - \frac{1}{2}utC\_5 + \frac{1}{8}uC\_6\Big],$$
$$[F\_4(x,t), C\_1, C\_2, C\_3, C\_4, C\_5, C\_6]\Big]$$

¿From the solutions of the infinitesimal functions, Lie vectors can be formed by using the function *genvec*(). The function *genvec*() has three arguments and returns a list of vectors, as follows:

genvec(*lsol, lpar, lvar*)

INPUT:
- *lsol* is a list of solutions
- *lpar* is a list of parameters
- *lvar* is a list of variables

OUTPUT:
- a list of vectors.

The left hand side of a solution in *lsol* must be a function. Each function on the left hand side in *lsol* corresponds to a variable in *lvar*. The corresponding pair of function and variable must have the same position in *lsol* and *lvar*. For example,

$$\begin{aligned} leqns &= [xi[x](\cdots) = \cdots, xi[t](\cdots) = \cdots, eta[u](\cdots) = \cdots] \\ lvar &= [x,t,u] \end{aligned}$$

A vector is expressed in the form of a sum of a list of a coefficient and a list of variables. As an example,

$$x\frac{\partial}{\partial t} + sin(u)\frac{\partial}{\partial x} - t^2\frac{\partial}{\partial u} \longrightarrow [x,[t]] + [sin(u),[x]] + [-t^2,[u]]$$

```
> lv := genvec(sol[3],sol[4],le[3]);
```

$$\begin{aligned} lv := \Big[& [F\_4(x,t),[u]], [1,[x]], \left[\frac{1}{2}t,[x]\right] + \left[-\frac{1}{4}ux,[u]\right], \\ & [1,[t]], [1,[u]], \left[\frac{1}{2}x,[x]\right] + [t,[t]], \\ & [xt,[x]] + \left[t^2,[t]\right] + \left[-\frac{1}{4}ux^2 - \frac{1}{2}ut,[u]\right]\Big] \end{aligned}$$

Thus in our example the Lie-vectors of the heat equation are

$$\frac{\partial}{\partial x}\,,\quad \frac{\partial}{\partial t}\,,\quad \frac{\partial}{\partial u}\,,$$
$$\frac{t}{2}\frac{\partial}{\partial x}-\frac{xu}{4}\frac{\partial}{\partial u}$$
$$\frac{x}{2}\frac{\partial}{\partial x}+t\frac{\partial}{\partial t}\,,$$
$$xt\frac{\partial}{\partial x}+t^2\frac{\partial}{\partial t}-\frac{u}{4}(x^2+2t)\frac{\partial}{\partial u}\,,\quad F\_4(x,t)\frac{\partial}{\partial u}$$

The function *comtab*() calculates the table of commutators of a list of vectors in the form that generates from *genvec*(). The function *comtab*() requires two arguments and returns a square matrix or a square matrix with a list of equations, as follows:

comtab(*lvec*,*lvar*)

INPUT:
- *lvec* is a list of vectors
- *lvar* is a list of variables

OUTPUT:
- a square matrix or a square matrix and a list of equations.

Each element of the output matrix is expressed linearly in terms of **psi**[i] which denotes the i^{th}-vector in *lvec*. The following Maple session shows the calculation of the commutator table of the vectors *lv* without the vector, $[F\_4(x,t),[u]]$, because it contains the unknown function $F\_4(x,t)$.

Example: (Heat equation)

```
> comtab(subsop(1=NULL,lv),le[3]);
```

$$\begin{bmatrix} 0 & -\frac{1}{4}\psi_6 & 0 & \frac{1}{2}\psi_1 & 2\psi_2 & 0 \\ \frac{1}{4}\psi_6 & 0 & -\frac{1}{2}\psi_1 & -\frac{1}{2}\psi_2 & 0 & 0 \\ 0 & \frac{1}{2}\psi_1 & 0 & \psi_3 & 2\psi_4-\frac{1}{2}\psi_6 & 0 \\ -\frac{1}{2}\psi_1 & \frac{1}{2}\psi_2 & \psi_3 & 0 & \psi_5 & 0 \\ -2\psi_2 & 0 & -2\psi_4+\frac{1}{2}\psi_6 & -\psi_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

When a commutator of two vectors cannot be expressed as a linear combination of **psi**[i], it is replaced by a variable, called **gamma**[k] (γ_k) in the commutator table where k is an integer. For this case, **comtab**() outputs is a matrix of commutator and a list of expressions of **gamma**[k].

Example: (Heat equation)

```
> ct := comtab(subsop(1=NULL,lv),le[3]):
> print(ct[1]);
```

$$\begin{bmatrix} 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \psi_1 \\ -\gamma_1 & 0 & -\frac{1}{4}\psi_6 & 0 & \frac{1}{2}\psi_1 & 2\psi_2 & 0 \\ -\gamma_2 & \frac{1}{4}\psi_6 & 0 & -\frac{1}{2}\psi_1 & -\frac{1}{2}\psi_2 & 0 & 0 \\ -\gamma_3 & 0 & \frac{1}{2}\psi_1 & 0 & \psi_3 & 2\psi_4 - \frac{1}{2}\psi_6 & 0 \\ -\gamma_4 & -\frac{1}{2}\psi_1 & \frac{1}{2}\psi_2 & \psi_3 & 0 & \psi_5 & 0 \\ -\gamma_5 & -2\psi_2 & 0 & -2\psi_4 + \frac{1}{2}\psi_6 & -\psi_5 & 0 & 0 \\ -\psi_1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> ct[2];
```

$$\left[\gamma_1 = \left[-\left(\frac{\partial}{\partial x}F\_4(x,t)\right),[u]\right], \gamma_2 = \left[-\frac{1}{4}F\_4(x,t)x - \frac{1}{2}t\left(\frac{\partial}{\partial x}F\_4(x,t)\right),[u]\right],\right.$$
$$\gamma_3 = \left[-\left(\frac{\partial}{\partial t}F\_4(x,t)\right),[u]\right], \gamma_4 = \left[-\frac{1}{2}x\left(\frac{\partial}{\partial x}F\_4(x,t)\right) - t\left(\frac{\partial}{\partial t}F\_4(x,t)\right),[u]\right],$$
$$\left.\gamma_5 = \left[-\frac{1}{4}F\_4(x,t)x^2 - \frac{1}{2}F\_4(x,t)t - xt\left(\frac{\partial}{\partial x}F\_4(x,t)\right) - t^2\left(\frac{\partial}{\partial t}F\_4(x,t)\right),[u]\right]\right]$$

1.3.1 Arbitrary Functions of Derivatives

In classification problems, some equations may contain arbitrary functions of derivatives of dependent variables. When the defining equations are calculated, these arbitrary functions must be treated carefully. In most of the available symbolic packages, defining equation generators have problems in dealing with such equations, but it is not the case for the function *gendef*(). A demonstration of **Desolv** calculating the point symmetry of such an equation is shown. The quasilinear hyperbolic equation of the general form is

$$v_{tt} = f(x, v_x)\, v_{xx} + g(x, v_x) \tag{1.3}$$

This equation admits a symmetry group of 3 parameters.

$$\frac{\partial}{\partial t}\,,\ \frac{\partial}{\partial v}\,,\ t\frac{\partial}{\partial v}$$

Example: the PDE $v_{tt} = f(x, v_x)\, v_{xx} + g(x, v_x)$

```
> V := v(x,t):
> de := [ diff(V,t,t) = f(x,diff(V,x))*diff(V,x,x) + g(x,diff(V,x)) ];
```

$$de := \left[\left(\frac{\partial^2}{\partial t^2} v(x,t)\right) = f\left(x, \frac{\partial}{\partial x} v(x,t)\right)\left(\frac{\partial^2}{\partial x^2} v(x,t)\right) + g\left(x, \frac{\partial}{\partial x} v(x,t)\right)\right]$$

```
> le := gendef(de,[v],[x,t]);
```

$$le := \left[\left[\xi_x(x,t,v),\ \frac{\partial}{\partial t}\xi_t(x,t,v),\ \frac{\partial}{\partial v}\eta_v(x,t,v),\ \frac{\partial}{\partial x}\eta_v(x,t,v),\ \frac{\partial}{\partial v}\xi_t(x,t,v),\ \frac{\partial}{\partial x}\xi_t(x,t,v),\ \frac{\partial^2}{\partial t^2}\eta_v(x,t,v)\right],\ \left[\xi_x(x,t,v), \xi_t(x,t,v), \eta_v(x,t,v)\right],\ [x,t,v]\right]$$

```
> sol := pdesolv(op(le));
```

$$sol := [[\,],[\,],[\xi_x(x,t,v) = 0,\ \xi_t(x,t,v) = C\_1,\ \eta_v(x,t,v) = C\_2 + tC\_3]\,,\ [C\_1, C\_2, C\_3]$$

```
> genvec(sol[3],sol[4],le[3]);
```

$$[[1,[t]],\ [1,[v]],\ [t,[v]]]$$

1.4 Nonclassical Symmetries

For the Burgers' equation [**?**]

$$u_t = u_{xx} + 2uu_x \tag{1.4}$$

the infinitesimal generator is

$$\mathbf{v} = xi[x]\frac{\partial}{\partial x} + xi[t]\frac{\partial}{\partial t} + eta[u]\frac{\partial}{\partial u} \tag{1.5}$$

The invariant surface condition (ISC) is

$$eta[u] = xi[t]u_t + xi[x]u_x \tag{1.6}$$

For the case 1 with $xi[t] = 1$, the ISC becomes

$$eta[u] = u_t + xi[x]u_x \tag{1.7}$$

and for the case 2 with $xi[x] = 1, xi[t] = 0$, the ISC is

$$eta[u] = u_x \tag{1.8}$$

The function *gennc*() generates the defining equations of the nonclassical symmetry. *gennc*() requires four arguments with the fourth argument to be optional and returns a list of elements, as follows:

gennc(*deqns, dvar, ivar, lcase*)

INPUT:
- *deqns* is a list of differential equations
- *dvar* is a list of dependent variables
- *ivar* is a list of independent variables
- *lcase* is a list of case-number (Optional)

OUTPUT: a list of 3 elements [*leqns,lfn,lvar*]
- *leqns* is a list of lists of differential expressions
- *lfn* is a list of infinitesimal functions
- *lvar* is a list of variables

gennc() will generate all the cases for the nonclassical method when the argument *lcase* is omitted. The case-numbers in *lcase* represent the cases that the function *gennc*() will output. The following Maple session shows the calculation of defining equations of nonclassical symmetry for the Burgers' equation with the case 1.

Example: Burger's equation

```
> read desolv:
> U := u(x,t):
> de := [ diff(U,t) = diff(U,x,x) + 2*U*diff(U,x) ];
```

$$de := \left[\frac{\partial}{\partial t}u(x,t) = \frac{\partial^2}{\partial x^2}u(x,t) + 2u(x,t)\frac{\partial}{\partial x}u(x,t)\right]$$

```
> le := gennc(de,[u],[x,t],[1]):
> le[1][1];
```

$$\left[\frac{\partial^2}{\partial u^2}\xi_x(x,t,u)\,,\ \xi_t(x,t,u) - 1, -2\left(\frac{\partial}{\partial x}\xi_x(x,t,u)\right)\eta_u(x,t,u) + \left(\frac{\partial^2}{\partial x^2}\eta_u(x,t,u)\right)\right.$$
$$\left. - \left(\frac{\partial}{\partial t}\eta_u(x,t,u)\right) + 2u\left(\frac{\partial}{\partial x}\eta_u(x,t,u)\right) 4u\left(\frac{\partial}{\partial u}\xi_x(x,t,u)\right) + \left(\frac{\partial^2}{\partial u^2}\eta_u(x,t,u)\right)\right.$$

$$+2\xi_x(x,t,u)\left(\frac{\partial}{\partial u}\xi_x(x,t,u)\right)-2\left(\frac{\partial^2}{\partial u\partial x}xi_x(x,t,u)\right),2u\left(\frac{\partial}{\partial x}\xi_x(x,t,u)\right)$$
$$+2\left(\frac{\partial^2}{\partial u\partial x}\eta_u(x,t,u)\right)-\left(\frac{\partial^2}{\partial x^2}\xi_x(x,t,u)\right)+2\xi_x(x,t,u)\left(\frac{\partial}{\partial x}\xi_x(x,t,u)\right)$$
$$-2\eta_u(x,t,u)\left(\frac{\partial}{\partial u}\xi_x(x,t,u)\right)+2\eta_u(x,t,u)+\left(\frac{\partial}{\partial t}\xi_x(x,t,u)\right)\Bigg]$$

This system of defining equations is nonlinear in $xi[x], xi[t]$ and $eta[u]$. When the function *pdesolv*() is used to simplify an system of nonlinear differential equations, it is better to exclude the decoupling module from the simplification process. The decoupling module very often runs out of memory space when dealing with nonlinear systems because of the complexity and expansion. The global parameter **desolv_module** which is set to 6, will force the function *pdesolv*() to use the first 6 modules, without the decoupling module.

Example: Burger's equation

```
> desolv_module := 6:
> sol := pdesolv(le[1][1],le[2],le[3]):
```

This Maple session yields three sets of solutions for the nonclassical symmetry of the Burgers' equation with the case 1. The first solution contains 2 unsolved equations.

$$4F\_5(t)^3+\left(\frac{\partial^2}{\partial t^2}F\_5(t)\right)+6F\_5(t)\left(\frac{\partial}{\partial t}F\_5(t)\right)=0$$

$$-\left(\frac{\partial^2}{\partial t^2}F\_6(t)\right)-4F\_5(t)^2F\_6(t)-4F\_5(t)\left(\frac{\partial}{\partial t}F\_6(t)\right)-2F\_6(t)\left(\frac{\partial}{\partial t}F\_5(t)\right)=0$$

$$\begin{aligned}\xi_t(x,t,u) &= 1\\ \xi_x(x,t,u) &= F\_5(t)x-F\_6(t)\\ \eta_u(x,t,u) &= -\frac{1}{2}x\left(\frac{\partial}{\partial t}F\_5(t)\right)+\frac{1}{2}\left(\frac{\partial}{\partial t}F\_6(t)\right)\\ &\quad -F\_5(t)^2x+F\_5(t)F\_6(t)-uF\_5(t)\end{aligned}$$

subject to $F\_5(t)x-F\_6(t)\neq 0$.

The second solution is

$$\begin{aligned}\xi_t(x,t,u) &= 1\\ \xi_x(x,t,u) &= -2u\\ \eta_u(x,t,u) &= 0\end{aligned}$$

The third solution contains three unsolved equations subject to one condition, $F\_1(x,t) \neq 0$.

$$\begin{gathered}-\left(\frac{\partial^2}{\partial x^2}F\_3(x,t)\right)+\left(\frac{\partial}{\partial t}F\_3(x,t)\right)+2\left(\frac{\partial}{\partial x}F\_1(x,t)\right)F\_3(x,t)=0\\ \left(\frac{\partial}{\partial t}F\_4(x,t)\right)-\left(\frac{\partial^2}{\partial x^2}F\_4(x,t)\right)-2\left(\frac{\partial}{\partial x}F\_3(x,t)\right)+2\left(\frac{\partial}{\partial x}F\_1(x,t)\right)F\_4(x,t)=0\\ \left(\frac{\partial}{\partial t}F\_1(x,t)\right)-2\left(\frac{\partial}{\partial x}F\_4(x,t)\right)-\left(\frac{\partial^2}{\partial x^2}F\_1(x,t)\right)+2F\_1(x,t)\left(\frac{\partial}{\partial x}F\_1(x,t)\right)=0\end{gathered}$$

$$\begin{aligned}\xi_t(x,t,u) &= 1\\ \xi_x(x,t,u) &= F\_1(x,t)+u\\ \eta_u(x,t,u) &= -u^3-F\_1(x,t)u^2-F\_3(x,t)-uF\_4(x,t)\end{aligned}$$

1.5 Potential symmetries

An important calculation in the case of potential symmetry is to find *potential factors*. Potential factors are needed to put an equation into a potential form, from which a potential system can be derived. In this section, an example of calculating potential factors using **Desolv** is demonstrated.

Example: Consider the nonlinear diffusion equation $S_0\{u\}$ [**?**] given by

$$u_t = (K(u)u_x)_x \tag{1.9}$$

The Fréchet derivative of S_0 can be computed by the function *frechet*():

frechet(*leqns, dvar, ivar*)

INPUT:
- *leqns* is a list of differential equations
- *dvar* is a list of dependent variables
- *ivar* is a list of independent variables

OUTPUT:
- a square matrix

The output is a matrix of vectors which are expressed in the form of a sum of lists.

```
> de := [ diff(u(x,t),t) = diff( K(u(x,t))*diff(u(x,t),x),x ) ];
```

$$de := \left[\frac{\partial}{\partial t}u(x,t) = D(K)(u(x,t))\left(\frac{\partial}{\partial x}u(x,t)\right)^2 K(u(x,t))\left(\frac{\partial^2}{\partial x^2}u(x,t)\right)\right]$$

```
> F := frechet(de,[u],[x,t]);
```

$$\begin{aligned} F := & \left[[-K(u(x,t)),[x,x]] + \left[-2D(K)(u(x,t))\left(\frac{\partial}{\partial x}u(x,t)\right),[x]\right] + [1,[t]]\right. \\ & \left. + \left[-D^{(2)}(K)(u(x,t))\left(\frac{\partial}{\partial x}u(x,t)\right)^2 - D(K)(u(x,t))\left(\frac{\partial^2}{\partial x^2}u(x,t)\right),1\right]\right] \end{aligned}$$

The adjoint of the Fréchet derivative F can be calculated by calling the function *adjointD*():

adjointD(*M, lvar*)

INPUT:
- *M* is a square matrix of vectors
- *lvar* is a list of variables

OUTPUT:
- a square matrix

```
> adjointD(F,[x,t]);
```

$$[[-K(u(x,t)),[x,x]] + [-1,[t]]]$$

The function *genfac*() generates the system of defining equations for potential factors of a system of differential equations. The function *genfac*() requires three arguments, a list of equations, a list of dependent variables and a list of independent variables:

genfac(*deqns, dvar, ivar*)

INPUT:
- *deqns* is a list of differential equations
- *dvar* is a list of dependent variables
- *ivar* is a list of independent variables

OUTPUT: a list of 3 elements [*leqns,lfn,lvar*]
- *leqns* is a list of differential expressions
- *lfn* is a list of potential factor-functions
- *lvar* is a list of variables

The potential factor-functions are denoted as **lambda[i]** by default where i are integers.

```
> le := genfac(de,[u],[x,t]);
```

$$le := \left[\left[\frac{\partial}{\partial u}\lambda_1(x,t,u), -K(u)\left(\frac{\partial^2}{\partial x^2}\lambda_1(x,t,u)\right) - \left(\frac{\partial}{\partial t}\lambda_1(x,t,u)\right)\right], [\lambda_1(x,t,u)], [x,t,u]\right]$$

```
> sol := pdesolv(op(le));
```

$$sol := [[\,], [\,], [\lambda_1(x,t,u) = C\_1 + xC\_2], [C\_1, C\_2]]$$

Therefore the potential factors of the system S_0 are $\{1, x\}$. The potential factor $\lambda_1 = x$ yields the potential system S_1**{u,v}**,

$$v_x = xu \tag{1.10}$$
$$v_t = x(L(u))_x - L(u) \tag{1.11}$$

where $K(u) = L'(u)$. By repeating the above steps, we find that the solutions for the potential factors of S_1**{u,v}** are $(\lambda_1, \lambda_2) = (0, x^{-2})$.

```
> U := u(x,t): V := v(x,t):
> de := [ diff(V,x) = x*U, diff(V,t) = x*diff( L(U),x) - L(U) ];
```

$$de := \left[\frac{\partial}{\partial x}v(x,t) = xu(x,t), \frac{\partial}{\partial t}v(x,t) = xD(L)(u(x,t))\left(\frac{\partial}{\partial x}u(x,t)\right) - L(u(x,t))\right]$$

```
> F := frechet(de,u,v],[x,t]);
```

$$F := [[-x,1],[1,[x]]]\left[[-xD(L)(u(x,t)),[x]]+\left[-xD^{(2)}(L)(u(x,t))\left(\frac{\partial}{\partial x}u(x,t)\right)+D(L)(u(x,t)),1\right],[1,[t]]\right]$$

```
> adjointD(F,[x,t]);
```

$$\begin{bmatrix} [-x,1] & [xD(L)(u(x,t)),[x]]+[2D(L)(u(x,t)),1] \\ [-1,[x]] & [-1,[t]] \end{bmatrix}$$

```
> le := genfac(de,[u,v],[x,t]);
```

$$le := \left[\left[\frac{\partial}{\partial u}\lambda_2(x,t,u,v),-\left(\frac{\partial}{\partial u}\lambda_1(x,t,u,v)\right)-\left(\frac{\partial}{\partial v}\lambda_2(x,t,u,v)\right)x\left(\frac{\partial}{\partial u}L(u)\right),\right.\right.$$
$$-x\lambda_1(x,t,u,v)+x\left(\frac{\partial}{\partial u}L(u)\right)\left(\frac{\partial}{\partial x}\lambda_2(x,t,u,v)\right)+x^2\left(\frac{\partial}{\partial u}L(u)\right)$$
$$\left(\frac{\partial}{\partial v}\lambda_2(x,t,u,v)\right)u+2\left(\frac{\partial}{\partial u}L(u)\right)\lambda_2(x,t,u,v),$$
$$-\left(\frac{\partial}{\partial x}\lambda_1(x,t,u,v)\right)-\left(\frac{\partial}{\partial v}\lambda_1(x,t,u,v)\right)xu-\left(\frac{\partial}{\partial t}\lambda_2(x,t,u,v)\right)$$
$$\left.\left.+\left(\frac{\partial}{\partial v}\lambda_2(x,t,u,v)\right)L(u)\right],[\lambda_1(x,t,u,v),\lambda_2(x,t,u,v)],[x,t,u,v]\right]$$

```
> sol := pdesolv(op(le));
```

$$sol := \left[[\,],[\,],\left[\lambda_1(x,t,u)=0,\lambda_2(x,t,u)=\frac{C\_1}{x^2}\right],[C\_1]\right]$$

1.6 Classification

Example: The nonhomogenous Monge-Ampère equation contains an arbitrary function $a(x,y)$,

$$u_{xx}u_{yy}-u_{xy}^2+a^2=0 \tag{1.12}$$

Different values of the arbitrary function give different symmetry groups (see, e.g., the section 9.4 in the handbook of Lie group analysis of differential equations Vol.1, N.H.Ibragimov [**?**]). The case where $a=x^\beta y^\gamma$ is examined:

```
> U := u(x,y):
> de := [diff(U,x,x)*diff(U,y,y)-diff(U,x,y)^2+(x^beta*y^gamma)^2=0];
```

$$de := \left[\left(\frac{\partial^2}{\partial x^2}u(x,y)\right)\left(\frac{\partial^2}{\partial y^2}u(x,y)\right) - \left(\frac{\partial^2}{\partial y \partial x}u(x,y)\right)^2 + \left(x^{\beta}\right)^2 \left(y^{\gamma}\right)^2 = 0\right]$$

```
> le := gendef(de,[u],[x,y]);
```

$$le := \left[\left[\frac{\partial}{\partial u}\xi_y(x,y,u),\ \frac{\partial}{\partial u}\xi_x(x,y,u),\ \frac{\partial^2}{\partial x^2}\xi_y(x,y,u),\ \frac{\partial^2}{\partial y \partial x}\eta_u(x,y,u),\right.\right.$$
$$\frac{\partial^2}{\partial y^2}\eta_u(x,y,u),\ \frac{\partial^2}{\partial u^2}\eta_u(x,y,u),\ \frac{\partial^2}{\partial x^2}\eta_u(x,y,u),\ \frac{\partial^2}{\partial y^2}\xi_x(x,y,u),$$
$$2\left(\frac{\partial^2}{\partial u \partial y}\eta_u(x,y,u)\right) - \left(\frac{\partial^2}{\partial y^2}\xi_y(x,y,u)\right),\ -\left(\frac{\partial^2}{\partial u \partial x}\eta_u(x,y,u)\right)$$
$$+\left(\frac{\partial^2}{\partial y \partial x}\xi_y(x,y,u)\right),\ \left(\frac{\partial^2}{\partial u \partial y}\eta_u(x,y,u)\right) + \left(\frac{\partial^2}{\partial y \partial x}\xi_x(x,y,u)\right),$$
$$2\left(\frac{\partial^2}{\partial u \partial x}\eta_u(x,y,u)\right) - \left(\frac{\partial^2}{\partial x^2}\xi_x(x,y,u)\right),\ -\left(\frac{\partial}{\partial y}\xi_y(x,y,u)\right) yx$$
$$\left.-\left(\frac{\partial}{\partial x}\xi_x(x,y,u)\right) yx + \left(\frac{\partial}{\partial u}\eta_u(x,y,u)\right) yx - \xi_y(x,y,u)\gamma x - \xi_x(x,y,u)\beta y\right],$$
$$\left.\left[\xi_x(x,y,u), \xi_y(x,y,u), \eta_u(x,y,u)\right], [x,y,u]\right]$$

```
> sol := pdesolv(op(le));
```

$$sol := \left[[\,],[\,],\left[\xi_x(x,y,u) = xC\_10,\right.\right.$$
$$\xi_y(x,y,u) = -\frac{y(C\_14 + C\_10 + C\_10\beta)}{\gamma + 1},$$
$$\left.\eta_u(x,y,u) = -C\_13 - uC\_14 + yC\_1 + xC\_3\right],$$
$$\left.[C\_1, C\_3, C\_10, C\_13, C\_14]\right]$$

```
> lv := genvec(sol[3],sol[4],le[3]);
```

$$lv := [[y,[u]],[x,[u]],[x + x\gamma,[x]] + [-y - y\beta,[y]],[1,[u]],[x,[x]] + [u + u\beta,[u]]]$$

The equation admits 5 symmetries,

$$y\frac{\partial}{\partial u}\,,\ x\frac{\partial}{\partial u}\,,\ \frac{\partial}{\partial u}\,,$$

$$(1+\gamma)x\frac{\partial}{\partial x}-(1+\beta)y\frac{\partial}{\partial y}\,,\ x\frac{\partial}{\partial x}+(1+\beta)u\frac{\partial}{\partial u}$$

During the simplification process, the function **pdesolv**() assumes some expressions to be non-zero such as denominators and coefficients of the leading derivatives. When **pdesolv**() applies the method of direct separation to an equation, the expressions which depend on the separating variable are assumed to be linearly independent of each other. The function **classify**() displays non-zero assumptions and linearly independent assumptions. The arguments of the function **classify**(), which are optional, are names of arbitrary functions or constants.

```
> classify(gamma,beta);
```

$$\begin{gathered}\textit{non-zero assumptions } \gamma\\ \beta\\ \gamma+1\\ \beta+1\\ \beta+2+\gamma\\ \textit{linearly independent assumptions}\end{gathered}$$

If these expressions of β and γ are equal to zero, the equation may admit different sets of symmetries. These expressions are stored in the global variable, **desolv_nzassume** and **desolv_liassume**. For $\gamma=\beta=-1$, the equation becomes

$$u_{xx}u_{yy}-u_{xy}^2+x^{-2}y^{-2}=0 \tag{1.13}$$

This equation admits 7 symmetries instead of 5 in the general case.

```
> U := u(x,y):
> de := [ diff(U,x,x)*diff(U,y,y) - diff(U,x,y)^2 + 1/(x*y)^2 = 0];
> le := gendef(de,[u],[x,y]);
> sol := pdesolv(op(le));
```

$$sol := \left[[\,],[\,],\left[\xi_x(x,y,u) = -x^2C\_12 - xC\_17 + xyC\_2,\right.\right.$$
$$\xi_y(x,y,u) = y^2C\_2 + yC\_15 - xyC\_12,$$
$$\left.\left.\eta_u(x,y,u) = C\_5 + yC\_9 + yuC\_2 + xC\_7 - xuC\_12\right],\right.$$
$$\left.[C\_2, C\_5, C\_7, C\_9, C\_12, C\_15, C\_17]\right]$$

```
> lv := genvec(sol[3],sol[4],le[3]);
```

$$lv := [[x,[u]],[1,[u]],[xy,[x]]+[y^2,y]+[yu,[u]],[y,[u]],[y,[y]],$$
$$[x^2,[x]]+[xy,[y]]+[xu,[u]],[x,[u]]]$$

1.7 Decoupling

A system of differential equations which contain arbitrary functions (or classified functions) often cause difficulties for simplification. A system of nonlinear telegraph equations [**?**] which contains two arbitrary functions f and g is

$$v_t = u_x \tag{1.14}$$
$$v_x = f(u)u_t + g(u) \tag{1.15}$$

The 8 defining equations for point symmetries are

$$-\frac{\partial\xi_x}{\partial u} + \frac{\partial\xi_t}{\partial v} = 0$$
$$-\frac{\partial\xi_t}{\partial u} + f\frac{\partial\xi_x}{\partial v} = 0$$
$$\frac{\partial\eta_u}{\partial u} + \frac{\partial\xi_t}{\partial t} - \frac{\partial\eta_v}{\partial v} + \frac{\partial\xi_x}{\partial x} = 0$$
$$\frac{\partial\eta_u}{\partial x} - \frac{\partial\eta_v}{\partial t} + g\frac{\partial\xi_x}{\partial t} + g\frac{\partial\eta_u}{\partial u} = 0$$
$$-\frac{\partial\xi_t}{\partial x} - g\frac{\partial\xi_t}{\partial v} - \frac{\partial\eta_v}{\partial u} + f\frac{\partial\xi_x}{\partial t} + g\frac{\partial\xi_x}{\partial u} + f\frac{\partial\eta_u}{\partial v} = 0$$
$$-g\frac{\partial\xi_x}{\partial u} - g\frac{\partial\xi_t}{\partial v} - f\frac{\partial\eta_u}{\partial v} + \frac{\partial\eta_v}{\partial u} + f\frac{\partial\xi_x}{\partial t} - \frac{\partial\xi_t}{\partial x} = 0$$
$$g\frac{\partial\eta_v}{\partial v} - g^2\frac{\partial\xi_x}{\partial v} + \frac{\partial\eta_v}{\partial x} - g'\eta_v - f\frac{\partial\eta_u}{\partial t} - g\frac{\partial\xi_x}{\partial x} = 0$$

None of these equations can be simplified by the first six modules of the function **pdesolv**(). Thus the defining system needs to be transformed into the standard form by the decoupling module. The function **decouple**() takes in three arguments, a list of equations, a list of unknowns and a list of variables, as follows,

decouple(*leqns, lfn, lvar*)

INPUT:
- *leqns* is a list of differential equations
- *lfn* is a list of unknown functions
- *lvar* is a list of variables

OUTPUT:
- a list of differential equations

As the function **decouple**() simplifies the above system, the calculation does not get very far before the computer is hung up due to a great expansion of the system. The parameter **infolevel**[decouple] can be set to be 3 to display intermediate computation of the function **decouple**().

```
> U := u(x,t): V := v(x,t):
> de := [ diff(V,t) = diff(U,x), diff(V,x) = f(U)*diff(U,t) + g(U)]:
> le := gendef(de,[u,v],[x,t]):
> infolevel[decouple] := 3:
> s  := decouple(le[1],le[2],le[3]):
```

The large expansion is caused by equations whose coefficients of highest derivatives depend on arbitrary functions $f(u)$ and $g(u)$. When these equations reduce another equation, they may be differentiated with respect to u a number of times. One of the techniques used by the function **decouple**() to overcome the expansion problem is to replace large coefficients of the highest derivatives with arbitrary functions, namely AAA_k where k are integers. The functions AAA_k have the same dependencies as the coefficients. A replacement is done depending on how large the coefficient is. The global variable *desolv_sublimit* allows the user to control such replacements. If the length of a coefficient is larger than the value of **desolv_sublimit**, a replacement will take place. By default, **desolv_sublimit** is set to be at 10^5.

Another technique of avoiding the expansion is to delay the reduction of integrability conditions. When an integrability condition is computed, immediately it is reduced with respect to the current system. The reduction can be delayed until all integrability conditions are calculated by setting the parameter **desolv_reduce** to be *false*. By default, the parameter **desolv_reduce** is set to be

true. The following Maple session shows how the function *decouple*() simplifies the above defining equations by setting **desolv_reduce** = *false* and **desolv_sublimit** = 100.

Example:

```
> read desolv:
> de := [ diff(v(x,t),t) = diff(u(x,t),x),
diff(v(x,t),x) = f(u(x,t))*diff(u(x,t),t) + g(u(x,t)) ]:
> le := gendef(de,[u,v],[x,t]):
> desolv_reduce := false:
> desolv_sublimit := 100:
> s1 := decouple(le[1],le[2],le[3]);
```

$$s1 := \left[\eta_u(x,t,u,v),\ \frac{\partial}{\partial v}\xi_x(x,t,u,v),\ \frac{\partial}{\partial v}\xi_t(x,t,u,v),\ \frac{\partial}{\partial v}\eta_v(x,t,u,v),\right.$$
$$\frac{\partial}{\partial u}\xi_x(x,t,u,v),\ \frac{\partial}{\partial u}\xi_t(x,t,u,v),\ \frac{\partial}{\partial t}\xi_x(x,t,u,v),\ \frac{\partial}{\partial u}\eta_v(x,t,u,v),$$
$$\frac{\partial}{\partial t}\xi_t(x,t,u,v),\ \frac{\partial}{\partial t}\eta_v(x,t,u,v),\ \frac{\partial}{\partial x}\xi_t(x,t,u,v),$$
$$\left.\frac{\partial}{\partial x}\xi_t(x,t,u,v),\ \frac{\partial}{\partial x}\eta_v(x,t,u,v)\right]$$

```
> sol := pdesolv(s1,le[2],le[3]);
```

$$sol := [[\,],\ [\,],\ [\xi_x(x,t,u,v) = C\_3,$$
$$\xi_t(x,t,u,v) = C\_1,\ \eta_u(x,t,u,v) = 0,\ \eta_v(x,t,u,v) = C\_2],$$
$$[C\_1,\ C\_2,\ C\_3]]$$

```
bytes used=82878192, alloc=2489912, time=547.67
```

This Maple session took 547 seconds of CPU time to complete. The intermediated calculation of the function *decouple*() can be displayed by setting the variable **infolevel**[**decouple**]. The higher the value of **infolevel**[**decouple**], the more calculation is displayed.