Transversal

A Maple Package For Singularity Theory

Version 3.1

Neil P. Kirk

November 1998

Contents

Preface

Acknowledgements. The majority of the `Transversal` package was developed and written as part of my Ph.D. thesis. I would like to thank my supervisor J.W. Bruce for introducing me to the techniques of classification theory and for his help during the development and testing of the package. He was a major influence in the success of the project. Thanks are also due to W. Hawes, C.A. Hobbs, K. Houston, T. Nishimura, F. Tari and J.M. West for their help with testing. The package has since been extended and used in several large projects supported by the EPSRC grants GR/J28162, GR/L33948 and GR/L17245; computational work was supported by grant GR/H59855.

Department of Mathematical Sciences,
University of Liverpool,
P.O. Box 147,
Liverpool,
L69 3BX.
England.

Email: neilpk@liverpool.ac.uk

Chapter 1

Introduction

1.1 Background

Calculations which arise in local singularity theory lend themselves naturally to symbolic algebra methods. In this article we describe a package which deals with problems in classification and unfolding theory for the standard equivalence relations encountered in singularity theory. The package, called `Transversal`, consists of a collection of procedures which run under the the symbolic algebra system `Maple` [8].

We refer to the survey article of Wall [27] and the book of Martinet [18] for a comprehensive discussion of the singularity theory used in this article. The more recent advances in determinacy and classification theory are discussed in the articles by Bruce, Kirk, du Plessis and Wall [5, 7]. The techniques developed in these provide a very efficient, wide-ranging classification scheme involving algebraic calculations which may be reduced to finite dimensional symbolic problems. However, the calculations can become very intensive and repetitive which is where the need for a specialist computer package arises.

The applications we have in mind require the calculation of certain 'tangent spaces' in a jet-space. This calculation involves manipulation of truncated polynomial vectors and is therefore really just a problem in linear algebra which can be handled by a computer. For example, in classification problems the calculation can be reduced to the enumeration of the orbits of the jet-group. In this situation we are considering Lie groups acting on smooth manifolds and have powerful techniques such as *Mather's Lemma* [27, Lemma 1.1] and *Complete Transversals* [5] at our disposal. (In fact, we are dealing with algebraic groups over **R** or **C** acting algebraically on an affine space and stronger results can be established. Although of theoretical importance, we will not need such results in our present applications.) It turns out that all of the information that we require can be obtained from a calculation of the tangent spaces to the orbits of the jet-group in the jet-space. Calculations in unfolding theory can be reduced to similar symbolic manipulations. We do not have a Lie group action in this case (we only have the notion of 'extended equivalence' at the germ level) but unfolding theory allows us to work with the associated ‘extended tangent spaces’. Once we have concluded that a given germ is finitely-determined (using the above methods) we may perform unfolding calculations in a suitable jet-space. At the jet level, the calculation of these ‘extended tangent spaces’ involves identical symbolic manipulations to those required in classification calculations.

The ‘tangent spaces’ are given by the action of a space of vector fields L on a given germ or jet. For example, if L is the Lie algebra of a jet-group then the tangent space to the orbit through the jet f is given by the natural action of the Lie algebra and is denoted by $L \cdot f$. We will use ‘tangent space’ as a general term to refer to such spaces $L \cdot f$ (even though they are not necessarily tangent to some submanifold). The terminology is used both at the jet and germ level. (This notation was established in the more recent work [5, 7] as a preferred alternative to the ad-hoc notation $T\mathcal{G} \cdot f$ used previously.) The main feature of our package is its ability to calculate and manipulate the spaces $L \cdot f$. Our aim was to produce a package capable of performing the calculations over a wide range of equivalence relations. In particular, it must apply to the cases where $\mathcal{G}$, a subgroup of $\mathcal{K}$ defining the equivalence relation, is one of the standard Mather groups $\mathcal{R}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{K}$ [27]; or more generally one of Damon’s geometric subgroups [10] for which a set of generators of the Lie algebra $L\mathcal{G}$ can be written down explicitly.

Let us consider one of the important research areas in singularity theory, namely the case of $\mathcal{A}$-equivalence. Not only is this a natural generalisation of $\mathcal{R}$-equivalence but it has significant applications in geometry and related areas such as computer vision. For example, in such applications one often wants to consider the simultaneous contact between a submanifold and a whole family of model submanifolds, typically families of lines, planes, circles, spheres, and so forth. In these situations we must work with $\mathcal{A}$-equivalence rather than contact ($\mathcal{K}$) equivalence, the difference essentially being that contact between nearby fibres of the map is preserved under $\mathcal{A}$-equivalence, whereas $\mathcal{K}$-equivalence only relates to the contact class associated to one fibre. For a recent survey of geometrical applications of singularity theory we refer to the article of Bruce [4] and the extensive bibliography therein. A real obstruction in obtaining $\mathcal{A}$-classifications is the size of the computations involved in all but the simplest of examples. One only has to refer to the existing papers dealing with $\mathcal{A}$-classification to see this, for example, those of Mond, du Plessis and Rieger [20, 21, 24]. For such applications any useful package must be able to calculate $L \cdot f$ in a given jet-space for given jet f where $L = L\mathcal{A}$ (for applications of Mather’s Lemma, calculation of $\mathcal{A}$-invariants, moduli detection); $L = L\mathcal{A}_1$ with the possible inclusion of a nilpotent part (for determinacy and complete transversal calculations in classification problems); and $L = L\mathcal{A}_e$ (for unfolding calculations). The package achieves all of these requirements and we cite its success in $\mathcal{A}$-classifications as its single most important application. For example, the aforementioned results of Mond and Rieger were all reproduced in a matter of hours using `Transversal`. Recent applications of `Transversal` [6, 13, 14, 15, 16, 28] represent some of the most extensive classifications carried out to date. The package has been extended to deal with weighted homogeneous filtrations, multigerms and cases where the equivalence $\mathcal{G}$ derives from a set of liftable or lowerable vector fields (the latter providing new results in the theory of caustics and envelopes).

1.2 A Guide to Using the Package

The remainder of this manual is organised as follows. Chapter 2 reviews the techniques from singularity theory which the package relies upon. An overview of the package is provided in Chapter 3. The basic scheme of the algorithm is described followed by a discussion of some of the more important technical issues. Chapters 2 and 3 form the basis of an article describing the package [17]. Chapter 4 is a reference guide for the package and describes such features as function calls and how to initialise specific calculations. In the true spirit of reference manual it is written in a somewhat terse manner! This is in contrast to Chapter 5 which acts as a tutorial, taking the reader through worked examples of some standard calculations from singularity theory. We bring the reader's attention to the important remarks regarding the use of the package in Section 5.1.

We suspect that most users will want to use the package with the least amount of effort. With regard to this we recommend the following as the smoothest route for getting started. Glance through Chapter 2 to make sure you are familiar with the background singularity theory. These techniques form the basis of the package and we assume that the reader has a full grasp on them, following up the references if need be. Chapter 3 forms an excellent introduction to the package. Section 3.1 should be read in detail as the remainder of the manual assumes the reader is familiar with the details there. Section 3.2 is somewhat more technical in nature, though it is probably worth glancing through it on first reading, referring back as necessary. Following this the reader should be able to work through the tutorial section in Chapter 5. The examples there should provide enough details for the reader to progress onto their own calculations and projects. A more precise specification of the package is found in the reference manual, Chapter 4, which should be consulted as necessary (and should hopefully be a little more readable at this stage!). This user manual should, of course, be studied in greater detail to achieve a better understanding of the package, as required.

The notation used throughout the manual is standard, based on (some of) that developed in [18, 27]. In addition, we adopt the more systematic notation used in [5, 7] and clarify the following. The theory applies over both the real and complex numbers and $\mathbf{F}$ will denote either $\mathbf{R}$ or $\mathbf{C}$. (In addition, the classifications in these cases hardly differ. Minor simplifications occur in the $\mathbf{C}$ case due to the collapsing of orbits, most commonly resulting in the removal of a $\pm$ sign in the normal form.) The local ring of differentiable/analytic function-germs $\mathbf{F}^n, 0 \to \mathbf{F}$ is denoted by $\mathcal{E}_n$ and its maximal ideal by $\mathcal{M}_n$. The corresponding module of map-germs $\mathbf{F}^n, 0 \to \mathbf{F}^p$ is denoted $\mathcal{E}(n,p)$, those with zero target are therefore given by $\mathcal{M}_n\mathcal{E}(n,p)$. We define the standard kth-jet-space $J^k(n,p)$ to be $\mathcal{M}_n\mathcal{E}(n,p)/\mathcal{M}_n^{k+1}\mathcal{E}(n,p)$. This is identified with the space of p-tuples of polynomials in n indeterminates over

F, truncated to degree k; a germ f being identified with its Taylor expansion to degree k. Unless otherwise stated, $\mathcal{G}$ will denote a subgroup of the contact group $\mathcal{K}$, usually one of the standard Mather groups $\mathcal{R}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{K}$, but in principle one of Damon's geometric subgroups. We let $\mathcal{G}_k$ be the normal subgroup of $\mathcal{G}$ consisting of those germs whose k-jet is equal to that of the identity. The standard kth-jet-group $J^k\mathcal{G}$ is defined to be the quotient group $\mathcal{G}/\mathcal{G}_k$. This is a Lie group and acts on the affine space $J^k(n,p)$. We will abbreviate the term 'complete transversal' as 'CT'.

1.3 Some History

I started writing the `Transversal` package as part of the work towards my Ph.D. thesis in 1991 while working in the area of classification of singularities. This work entailed numerous calculations ranging from tedious and repetitive to computationally demanding. It was clear, both from this work and the work of many others who have struggled through such classifications, that a general computer package which could carry out such calculations and in some sense automate the process would be a useful tool. Indeed, in some problems the use of a computer in some capacity is unavoidable (usually to answer questions regarding the ranks of very large matrices).

During my Ph.D. the main package dealing with standard classifications and classifications with respect to weighted filtrations was written, tested and applied to several substantial problems. Later, several colleagues became interested so I tidied things up a bit so that the package could be easily distributed and used (1994). Again, the package was tested and put to use in several problems (see above for references). More recently the package has been extended to deal with multigerm classifications (1996) and classifications with respect to lowerable diffeomorphisms (1997). That was pretty much it. But after further interest in the work I was persuaded that it should be distributed within the singularities community. I have since updated the package and its documentation and made it more suitable for general use and distribution (1998). This represents the current state of the project. Further improvements and extensions are, of course, possible. However, the package has reached a fairly stable version and, although I would describe it as an on-going project, I doubt whether any major changes will be made in the near future.

1.4 Some Related Computer Packages

We will end this introduction by giving a brief review of several existing computer packages which are aimed at singularity theory; we make no attempt at completeness. The packages related most closely to ours are the `CATFACT` package developed by Cowell and Wright [9]; the `OCRM` program written by Olsen, Carter and Rockwood (published in the book by Poston and Stewart [22], and corrected and enhanced by Millington [19]); and the `TGf` program written by Ratcliffe and referred to in [23]. The first two deal with the case of function-germs under $\mathcal{R}$-equivalence (an area which is often called 'elementary catastrophe theory'). The program developed by Ratcliffe is notable in that it performs similar calculations to `Transversal` and was written, independently, at about the same time that `Transversal` was written. The original version was restricted to $\mathcal{A}$-equivalence of map-germs from surfaces to 3-space and was used successfully in [23]. Both `TGf` and `OCRM` suffered from being written in a non-symbolic language (respectively, `Pascal` and a version of `ALGOL`). The final version of `TGf` (1994) was rewritten in Maple and the restriction to map-germs from surfaces to 3-space was lifted. All three programs are no longer being developed. The major improvements `Transversal` makes on these packages include an extensive broadening of the types of problems considered (for example, a greater variety of equivalence relations; extensions to multigerms and lowerable fields) together with the implementation of the latest classification techniques [5, 7]. Its success in several important and outstanding projects (cited above) is an indication of these claims.

We should add that the `CATFACT` package performs a lot more than determinacy and unfolding calculations. It contains a 'recognition' algorithm which identifies if a given function-germ belongs to one of those on Arnold's list of low modality singularities [1], and a 'reduction' algorithm which solves the 'mapping-problem' for unfoldings [9]. The 'recognition' algorithm calculates the Boardman symbol of the singularity (using Gröbner basis methods) and uses the fact that this identifies the low modality singularities. One needs to know Arnold's classification in advance to exploit such observations, which why it is necessary to obtain similar classifications of map-germs under the other important equivalence relations (in particular, the $\mathcal{A}$ and $\mathcal{K}$ cases). On a similar theme we note the 'recognition' program of Tari [26]. This implements a version of Arnold's 'determinator' algorithm [1] using Maple.

Other packages aimed at singularity theory include `Singular` and `Macaulay` [25, 3], though the latter deals more with applications in algebraic geometry. Both represent extensive ongoing projects. They have their own kernel which is purpose written to exploit techniques from computational commutative algebra, and their own user-interface and programming language. They have numerous applications in singularity theory, algebraic geometry and commutative algebra, but are not suited to the specialist area of classification problems discussed in this article, especially in the case of $\mathcal{A}$-classification.

Chapter 2

Applications to Singularity Theory

We discuss how our package may be used to solve problems in singularity theory and review the mathematical techniques which are required. Chapter 3 will describe how one actually implements these techniques in the package.

2.1 Classification Theory: Complete Transversals and Determinacy

In classification theory we seek to list orbits of finitely determined germs $f \in \mathcal{M}_n\mathcal{E}(n,p)$ under the action of the group $\mathcal{G}$, choosing suitable normal forms as representatives. Classification is done inductively at the jet-level, classifying in turn all $(k+1)$-jets with a given k-jet until determined jets result (or pre-selected upper bounds on moduli or codimension are reached). The method of 'complete transversals' provides an efficient means of carrying out this procedure. We recall some of the main results from [5, 7].

The group $\mathcal{G}$ is said to be *jet-closed* if for each $r \geq 1$, $J^r\mathcal{G}$ is a closed subgroup of $J^r\mathcal{K}$. If $\mathcal{G}$ is jet-closed it follows that $J^s(L\mathcal{G}) \subset L(J^s\mathcal{G})$ for all s. In many cases we have equality. If a jet-closed group $\mathcal{G}$ satisfies $J^s(L\mathcal{G}) = L(J^s\mathcal{G})$ for all s then we call it *fibrant*. We find that $\mathcal{R}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{K}$ are all jet-closed and fibrant. Further examples are given via the following concept. Let $\mathcal{H}$ be a subgroup of $\mathcal{G}$, then $\mathcal{H}$ is said to be *strongly closed* in $\mathcal{G}$ if $\mathcal{H}_s = \mathcal{G}_s$ for some s (equivalently $\mathcal{G}_s \subset \mathcal{H}$), and $J^s\mathcal{H}$ is closed in $J^s\mathcal{G}$. Now, a strongly closed subgroup $\mathcal{H}$ of a jet-closed group $\mathcal{G}$ is itself jet-closed. If, in addition, $\mathcal{G}$ is fibrant then so is $\mathcal{H}$.

The map

$$\begin{aligned} L(J^1\mathcal{K}) \cong gl(n,\mathbf{F}) \oplus gl(p,\mathbf{F}) &\to gl(n+p,\mathbf{F}) \\ (M,N) &\mapsto \begin{pmatrix} M & 0 \\ 0 & N \end{pmatrix}, \end{aligned}$$

where $gl(n, \mathbf{F})$ denotes the Lie algebra of the general linear group $GL(n, \mathbf{F})$, is a faithful representation of the Lie algebra $L(J^1\mathcal{K})$ on $\mathbf{F}^{n+p}$. Suppose that $L(J^1\mathcal{G})$ acts nilpotently on $\mathbf{F}^{n+p}$ under this representation. This happens if the source and target parts of $L(J^1\mathcal{G})$ are spanned by *strictly* upper (or lower) triangular matrices, for example. Generally the requirement is equivalent to $J^1\mathcal{G}$ being unipotent. In this situation the following sum is finite (see [7]) and we may define the *nilpotent filtration* of $\mathcal{M}_n\mathcal{E}(n,p)$,

$$M_{r,s}(\mathcal{G}) = \sum_{i\geq s}(L\mathcal{G})^i \cdot (\mathcal{M}_n^r\mathcal{E}(n,p)) + \mathcal{M}_n^{r+1}\mathcal{E}(n,p),$$

for integers $r \geq 1$ and $s \geq 0$. Observe that this is finer that the standard filtration by degree. For $r = 0$ we define $M_{0,0}(\mathcal{G})$ to be $\mathcal{M}_n\mathcal{E}(n,p)$ for consistency. The associated (r,s)*-jet-space* $J^{r,s}(n,p)$ is then defined to be $\mathcal{M}_n\mathcal{E}(n,p)/M_{r,s}(\mathcal{G})$. This is a refinement of the standard jet-space $J^r(n,p) = \mathcal{M}_n\mathcal{E}(n,p)/\mathcal{M}_n^{r+1}\mathcal{E}(n,p)$. Thus, $J^{r,0}(n,p)$ is $J^{r-1}(n,p)$, and as s increases $J^{r,s}(n,p)$ contains more of the homogeneous terms of degree r, until for some finite $s = k_r$ where we find that $J^{r,k_r}(n,p)$ is the whole of $J^r(n,p)$ (k_r exists due to nilpotency). Provided we work with these refined jet-spaces we have the following complete transversal result.

Theorem 2.1 *Let $\mathcal{G}$ be a fibrant subgroup of $\mathcal{K}$ such that $L(J^1\mathcal{G})$ acts nilpotently on $\mathbf{F}^{n+p}$. Let f be a smooth germ $\mathbf{F}^n, 0 \to \mathbf{F}^p, 0$ and let T be a subspace of $M_{r,s}(\mathcal{G})$ with*

$$M_{r,s}(\mathcal{G}) \subset T + L\mathcal{G}\cdot f + M_{r,s+1}(\mathcal{G}).$$

Then any germ $g : \mathbf{F}^n, 0 \to \mathbf{F}^p, 0$ with $g - f \in M_{r,s}(\mathcal{G})$ is $\mathcal{G}$-equivalent to a germ of the form $f + t + \phi$ with $t \in T$ and $\phi \in M_{r,s+1}(\mathcal{G})$.

This is really just a question in the standard jet-space $J^r(n,p)$ provided we order the homogeneous terms of degree r as dictated by $M_{r,s}(\mathcal{G})$. The latter can be achieved using a system of weights, see Section 3.2. The spaces T and $f + T$ are both referred to as a *complete transversal* (CT). One of the main features of the package is to calculate a basis for T, taking $L = J^r(L\mathcal{G})$. In practice, this is a process which has to be carried out numerous times and, as the classification proceeds, soon becomes computationally infeasible without the help of a computer.

Example 2.2 An example should clarify the above. Consider the classification of map-germs $\mathbf{F}^2, 0 \to \mathbf{F}^2, 0$ under $\mathcal{A}$-equivalence. Let (x, y) denote coordinates in the source and (u, v) those in the target. Recall that $\mathcal{A}_1$ denotes the subgroup of $\mathcal{A}$ consisting of those germs whose 1-jet is the identity and define $\mathcal{G}$ to be the unipotent subgroup of $\mathcal{A}$ having nilpotent Lie algebra

$$L \;=\; L\mathcal{A}_1 \;\oplus\; \mathbf{F}\{x\partial/\partial y\} \;\oplus\; \mathbf{F}\{v\partial/\partial u\}.$$

This acts on a germ $f = (f_1, f_2)$ by

$$L\cdot f = \mathcal{M}_2^2\langle\partial f/\partial x, \partial f/\partial y\rangle + f^*(\mathcal{M}_2^2)\{e_1, e_2\} + \mathbf{F}\{x\partial f/\partial y, f_2e_1\},$$

(r,s)	Homogeneous Part	Weight
$(1,0)$	$\{0\}$	
$(1,1)$	$\{(0,y)\}$	1
$(1,2)$	$\{(y,0),(0,x)\}$	2
$(1,3)$ or $(2,0)$	$\{(x,0)\}$	3
$(2,1)$	$\{(0,y^2)\}$	2
$(2,2)$	$\{(y^2,0),(0,xy)\}$	3
$(2,3)$	$\{(xy,0),(0,x^2)\}$	4
$(2,4)$ or $(3,0)$	$\{(x^2,0)\}$	5

Table 2.1: Generators for the homogeneous part of $J^{r,s}(\mathcal{G})$.

where e_1 and e_2 are the canonical basis vectors in $\mathbf{F}^2$. Each (r,s)-jet-space is just a refinement of the standard r-jet-space and a convenient way to describe these spaces is to list the 'homogeneous' generators for each of the spaces $J^{r,s}(2,2)$; see Table 2.1. The 'weight' column demonstrates the use of weights to partition the monomial vectors into their (r,s)-levels as described in Section 3.2; here $\alpha = (2,1)$ and $\beta = (-1,0)$. A similar example is discussed in the tutorial, Chapter 5, Section 5.2.1.

Example 2.3 The above results incorporate the notion of *strong equivalence*. For example, two germs are defined to be *strongly $\mathcal{A}$-equivalent* if they are $\mathcal{A}_1$-equivalent; that is, the diffeomorphism defining the equivalence has linear part the identity. Here we can take $\mathcal{G}$ to be the unipotent group $\mathcal{A}_1$. Thus, $M_{r,s}(\mathcal{G}) = \mathcal{M}_n^{r+1}\mathcal{E}(n,p)$ for all $s > 0$ and the CT theorem applies to the standard jet-spaces $J^r(n,p)$. Given a germ $f : \mathbf{F}^n, 0 \to \mathbf{F}^p, 0$, suppose T is a vector subspace of the space of homogeneous jets of degree $k+1$ such that

$$\mathcal{M}_n^{k+1}\mathcal{E}(n,p) \subset T + L\mathcal{A}_1 \cdot f + \mathcal{M}_n^{k+2}\mathcal{E}(n,p).$$

Then every germ F with $F - f \in \mathcal{M}_n^{k+1}\mathcal{E}(n,p)$ is $\mathcal{A}_1$-equivalent to a germ of the form $f + t + \phi$ with $t \in T$ and $\phi \in \mathcal{M}_n^{k+2}\mathcal{E}(n,p)$. That is, if $j^kF = j^kf$ then $j^{k+1}F$ is $J^{k+1}\mathcal{A}_1$-equivalent to a jet of the form $j^{k+1}f + t$, for some $t \in T$. This provides an $\mathcal{A}$-classification procedure with respect to familiar polynomial degree. However, in many classifications we need to use larger unipotent subgroups than $\mathcal{A}_1$ to obtain an efficient $\mathcal{A}$-classification procedure, at least during the early stages of the classification. We therefore have to classify in finer steps, using the refined jet-spaces $J^{r,s}(n,p)$, as in Example 2.2.

We now turn to the determinacy question. Algebraic criteria which characterise determinacy were found in [7]. These results also provide excellent determinacy estimates for use in practical situations. A version of the results suited to our needs is the following. We shall restrict to the case where $\mathcal{G}$ is one of the standard Mather groups to avoid the extra technicalities, though the determinacy results do apply to a larger class of groups.

Theorem 2.4 *Let $\mathcal{G}$ be one of $\mathcal{R}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{C}$ or $\mathcal{K}$ and let $\mathcal{H}$ be a strongly closed subgroup of $\mathcal{G}$ such that $L(J^1\mathcal{H})$ acts nilpotently on $\mathbf{F}^{n+p}$. Then a smooth germ $f : \mathbf{F}^n, 0 \to \mathbf{F}^p, 0$ is k-$\mathcal{H}$-determined if and only if*

$$\mathcal{M}_n^{k+1}\mathcal{E}(n,p) \subset L\mathcal{H} \cdot f.$$

Although the results are stated in terms of germs, they may be reduced to questions involving jets. We will show that establishing the degree of determinacy of a germ is a special case of calculating CTs. When the tangent space $L \cdot f$ is a module over $\mathcal{E}_n$ (for example, when $\mathcal{G} = \mathcal{R}$, $\mathcal{C}$ or $\mathcal{K}$) this follows from a simple application of *Nakayama's Lemma*; see [18, p.131] and [27, p.489]. We find that the germ is k-$\mathcal{G}$-determined if, when considered as a k-jet, the CT of degree $k+1$ is empty. For the remaining cases of interest, where $L \cdot f$ is a module over $\mathcal{E}_p$ via f^*, we apply a result of du Plessis [7, Lemma 2.6]. Probably the most important and informative application is where $\mathcal{G} = \mathcal{A}$, so we take this as an example. Applying du Plessis' result to the above determinacy theorem gives the following.

Theorem 2.5 *Using the notation of Theorem 2.4, f is k-$\mathcal{H}$-determined if and only if*

$$\mathcal{M}_n^{k+1}\mathcal{E}(n,p) \subset L\mathcal{H} \cdot f + \mathcal{M}_n^{k+1} f^*(\mathcal{M}_p)\mathcal{E}(n,p) + \mathcal{M}_n^{2k+2}\mathcal{E}(n,p).$$

Thus, f is k-$\mathcal{A}$-determined if the successive transversals from degree $k+1$ to degree $2k+1$ are empty. (Of course, the terms in $\mathcal{M}_n^{k+1} f^*(\mathcal{M}_p)\mathcal{E}(n,p)$ can be used to reduce the upper limit from $2k+1$. This is extremely important in applications, but the revised upper limit one obtains depends on the particular germ f.)

The spaces $L \cdot f$ used in determinacy calculations are precisely those used in CT calculations. We therefore obtain a very efficient classification process: if the determinacy criterion fails due to a non-empty transversal we simply continue the classification, the transversal providing us with a list of (possible) new branches in the classification tree.

Example 2.6 We reconsider Examples 2.2 and 2.3. For the former we take $\mathcal{H}$ in Theorems 2.4 and 2.5 to be the unipotent group $\mathcal{G}$ defined in Example 2.2. For strong determinacy considered in Example 2.3 we take $\mathcal{H}$ to be $\mathcal{A}_1$. As a further example consider $\mathcal{R}$-determinacy of function-germs. The condition for strong determinacy is given by taking $\mathcal{H}$ to be $\mathcal{R}_1$ and can be rewritten in the familiar form found in texts on elementary catastrophe theory, such as Poston and Stewart [22, p.134 and p.159], as follows. The germ f is k-$\mathcal{R}_1$-determined if and only if

$$\mathcal{M}_n^{k+1} \subset L\mathcal{R}_1 \cdot f = \mathcal{M}_n^2\langle \partial f/\partial x_1, \dots, \partial f/\partial x_n \rangle.$$

This provides a practical criterion for $\mathcal{R}$-determinacy.

2.2 Working with Jet-groups: Mather's Lemma and Moduli Detection

The method of CTs gives a complete set of representatives for the $J^{k+1}\mathcal{G}$-orbits over a given k-jet f. This set is given as an affine space in $J^{k+1}(n,p)$ through f and we wish to reduce it further, preferably to a finite set of representatives. This can often be achieved using 'scaling' coordinate changes in the source and target, a simple problem involving linear algebra. However, in cases where moduli are present, scaling is not possible and we need a criterion to detect such moduli. Alternatively, the family given by the affine space may be $\mathcal{G}$-trivial, collapsing to give one normal form, f. The space L used in CT calculations is generally smaller than the tangent space to the whole group $L\mathcal{G}$, so it is not surprising that a CT may contain redundant terms. (In general we cannot take L to be the whole of $L\mathcal{G}$, but there are theoretical reasons why the CT technique can provide the most efficient method of inductive classification.)

In cases where further simplification is necessary we have to work with the whole group $\mathcal{G}$, and a result specifically intended to deal with such questions is *Mather's Lemma* [27, Lemma 1.1]. We state it in our special case of interest, where a jet-group $J^k\mathcal{G}$ acts on $J^k(n,p)$.

Lemma 2.7 *Let X be a connected submanifold of $J^k(n,p)$. Then X is contained in a single orbit of $J^k\mathcal{G}$ if and only if*

(i) for each jet $x \in X$, the tangent space $T_xX \subset T_x(J^k\mathcal{G}\cdot x)$, and

(ii) $\dim T_x(J^k\mathcal{G}\cdot x)$ is constant for all $x \in X$.

The tangent space to the orbit through x is given by the action of the Lie algebra thus, $T_x(J^k\mathcal{G}\cdot x) = L(J^k\mathcal{G})\cdot x$. The two conditions of Mather's Lemma are extremely difficult to check using hand calculations but are easily dealt with by our package, taking $L = L(J^k\mathcal{G})$. Verifying the inclusion condition (i) requires little computational overhead once a basis for the tangent space has been calculated. Note that the jet passed to the package contains arbitrary parameters and represents a whole affine space in $J^k(n,p)$. Our algorithm will provide a set of exceptional values where the dimension of the tangent space may drop or the inclusion condition (i) fails. These exceptional values are stored for examination by the user after the algorithm has terminated; see Section 3.2.

A related issue is the detection of moduli. The CT process may produce an entire family of jets which are all distinct up to $\mathcal{G}$-equivalence. To prove that moduli are indeed present we use the following straightforward criteria.

Lemma 2.8 *Let W be a smooth constructible subset of the jet-space $J^k(n,p)$ and for $w \in W$ define*

$$d(w) = \dim\Big(\big(T_w(J^k\mathcal{G}\cdot w) + T_wW\big)/T_w(J^k\mathcal{G}\cdot w)\Big).$$

Then, given an integer $r \geq 1$, if the set $\{ w \in W : d(w) \leq r-1 \}$ is a constructible subset of W of smaller dimension, then every germ f with $j^k f \in W$ is of $\mathcal{G}$-modality r or greater.

Again, this is an extremely difficult condition to check using hand calculations. It may be verified easily by our package, taking $L = L(J^k\mathcal{G})$.

2.3 Unfolding Theory

Let $F : \mathbf{F}^n \times \mathbf{F}^s, 0 \to \mathbf{F}^p \times \mathbf{F}^s, 0$ defined by $(x, u) \mapsto (f(x, u), u)$ be an unfolding of $f_0 \in \mathcal{M}_n\mathcal{E}(n, p)$. We recall the following fundamental result from unfolding theory. (The case where $\mathcal{G}$ is one of the standard Mather groups is discussed in [18, 27]; for the generalisation to geometric subgroups of $\mathcal{K}$ see [10].)

Theorem 2.9 *F is $\mathcal{G}$-versal if and only if*

$$L\mathcal{G}_e \cdot f_0 + \mathbf{F}\{\dot{F}_1, \ldots, \dot{F}_s\} = \mathcal{E}(n, p),$$

where the initial speeds $\dot{F}_i \in \mathcal{E}(n, p)$ of F are defined by

$$\dot{F}_i(x) = \partial f/\partial u_i(x, 0), \quad \textit{for} \quad i = 1, \ldots, s.$$

Corollary 2.10 *If $g_1, \ldots, g_s \in \mathcal{E}(n, p)$ form an $\mathbf{F}$-spanning set for the normal space to $L\mathcal{G}_e \cdot f_0$ in $\mathcal{E}(n, p)$ then $F(x, u) = (f(x) + \sum_{i=1}^{s} u_i g_i(x), u)$ is a versal unfolding of f_0, where $u = (u_1, \ldots, u_s)$.*

Thus, to calculate a versal unfolding of f_0 we need to determine the g_i. As stated this is a problem at the germ level. However, if f_0 is k-$\mathcal{G}$-determined then by the characterisation of determinacy given in [7] (see Theorem 1.9 for $\mathcal{G}$ a standard Mather group, and Theorems 4.5 and 4.6 for more general subgroups of $\mathcal{K}$) we have $\mathcal{M}_n^{k+1}\mathcal{E}(n, p) \subset L\mathcal{G} \cdot f_0$. But the latter is a subset of $L\mathcal{G}_e \cdot f_0$ and it therefore suffices to calculate the normal space to $L\mathcal{G}_e \cdot f_0$ in $J^k(n, p)$. This is a simple application of the package, taking $L = J^k(L\mathcal{G}_e)$. (Note that in practical situations one usually establishes k-determinacy of f_0 by applying a determinacy result such as Theorem 2.4. In this case the above inclusion $\mathcal{M}_n^{k+1}\mathcal{E}(n, p) \subset L\mathcal{G} \cdot f_0$ follows directly from the determinacy criterion anyway.)

Chapter 3

Package Overview

This chapter provides an introduction to the package. Section 3.1 describes the basic setup of the package and the main scheme of the algorithm used. It is important that the reader understands this section as it is referred to in other parts of the manual. Section 3.2 describes some of the more important computational issues behind the algorithm. It is somewhat more technical in nature, though it is probably worth glancing through it on first reading. The information in this section will provide valuable background for the reader who is interested in some of the more subtle features of the algorithm but does not want a complete breakdown of how it works. Further details on the actual program code and a presentation of parts of the algorithm in the form of pseudo code were given in [16]. In addition we remark that the Maple source code is fully documented.

3.1 The Underlying Algorithm

The main principle behind the algorithm is to treat the spaces $L \cdot f$ as vector subspaces of the jet-space. Once a basis has been found we can answer all of the questions raised by the theory. It is a simple matter to calculate a spanning set for a given tangent space; reducing this to a basis is the major computational problem. Elements of the jet-space correspond to truncated polynomial vectors over the field of real or complex numbers. By extracting monomial coefficients we can treat jets as familiar coordinate vectors and reduce the spanning set to a basis using Gaussian elimination. A major concern with this approach is the size of the matrices involved. However, these matrices are highly sparse and, as numerous examples demonstrate, can be reduced relatively quickly. In addition, there are several features of the problem which we may exploit to reduce the computational overhead at the elimination stage. It is wasteful to extract coefficient vectors (which are generally of a high dimension) and create a matrix. Instead we apply the elimination directly to the polynomial vectors, manipulating them as symbolic expressions. This technique will be called *indexed Gaussian elimination.* The symmetry present in the 'target' tangent spaces (for example, types $\mathcal{L}$ and $\mathcal{C}$) is exploited at the elimination stage also. We will discuss these and other technical issues in Section 3.2.

Our concerns regarding large coefficient matrices, were also noted in [19] and Gröbner basis methods were used in the underlying algorithm in `CATFACT`. Although successful, this approach cannot generalise to $\mathcal{A}$-calculations because the algebraic structure of the $\mathcal{A}$-tangent space is that of a mixed module. That one has to treat the $\mathcal{A}$-tangent space as a vector space and work with the associated large matrices appears to be an unavoidable problem. The utility of our approach is ultimately measured by its success in dealing with important problems.

The stages of the algorithm are summarised below. The initial stage involves defining, via several global variables, the tangent space L we are to work with. We refer to Section 4.2 for a description of the actual variables.

Initialisation Step

Firstly, L is specified as one of five broad 'types' which we will denote $\mathcal{R}$, $\mathcal{L}$, $\mathcal{C}$, $\mathcal{A}$ and $\mathcal{K}$. The required 'type' is set by a global variable which may take the string constant values R, L, C, A and K. For 'type' $\mathcal{R}$, L is defined to act on a given jet f by

$$L \cdot f = \mathcal{M}_n^{t_1} \big\langle \xi_1 \cdot f, \ldots, \xi_s \cdot f \big\rangle,$$

where the exponent t_1 is given by a user-defined integer variable and the ξ_i are user-defined vector fields. The ξ_i are defined via a procedure which takes f as a parameter and returns the vectors $\xi_i \cdot f$; the procedure is pointed to by another global variable and is called at run-time. Several procedures are provided: the standard $\mathcal{R}$ case, where $\xi_i = \partial/\partial x_i$, is covered as are cases where L is the space of vector fields tangent to a discriminant variety. Thus, 'type' $\mathcal{R}$, with $\xi_i = \partial/\partial x_i$ and $t_1 = 0, 1$ and 2 defines the tangent spaces $L\mathcal{R}_e$, $L\mathcal{R}$ and $L\mathcal{R}_1$, respectively. For 'type' $\mathcal{L}$, L is defined to act by

$$L \cdot f = f^*(\mathcal{M}_p^{t_2})\{e_1, \ldots, e_p\},$$

where the e_i are the canonical basis vectors in $\mathbf{F}^p$ and t_2 is a user-defined integer variable. For 'type' $\mathcal{C}$ the action is defined by

$$L \cdot f = \mathcal{M}_n^{t_2} f^*(\mathcal{M}_p)\mathcal{E}(n, p).$$

As one would expect, for 'types' $\mathcal{A}$ and $\mathcal{K}$, $L \cdot f$ is defined as the sum of the spaces defined by 'types' $\mathcal{R}, \mathcal{L}$ and $\mathcal{R}, \mathcal{C}$, respectively.

This approach allows one to define a wide range of tangent spaces L and covers virtually everything which arises in practice. For complete transversal and determinacy techniques we often work with a unipotent subgroup of $\mathcal{K}$ and the corresponding nilpotent tangent space L is given by the sum of a 'standard' tangent space and a linear space spanned by a set of 'extra' vectors. For example, in the $\mathcal{A}$ case the space L is given by the sum of $L\mathcal{A}_1$ and a space spanned by 'extra' vectors belonging to $L\mathcal{A} \setminus L\mathcal{A}_1$; see [5, 7]. Further global variables specify these 'extra' vectors and the package can be used in such situations.

Having initialised the calculation we now call the appropriate functions in the package. The first three stages of the algorithm form the major part of the calculation and are performed by one function which takes a jet f and jet-space degree k as parameters.

Step 1

For the given jet f, jet-space degree k and tangent space L, calculate $L \cdot f$ in $J^k(n,p)$. Specifically, calculate a spanning set for $L \cdot f$ as a vector subspace of $J^k(n,p)$. The algorithm constructs this set using the definition of $L \cdot f$ given above for each 'type' and essentially follows the same procedure as that used if one were doing the calculation by hand. For example, in a standard $\mathcal{R}$ classification, using the complete transversal method with $L = L\mathcal{R}_1$ say, we calculate $L \cdot f = \mathcal{M}_n^2\langle \partial f/\partial x_1, \ldots, \partial f/\partial x_n\rangle$ by first obtaining the vectors which generate $L \cdot f$ as an $\mathcal{E}_n$-module, $\{\partial f/\partial x_i\}$. These are multiplied by all monomials of degree 2 and higher in the source variables until we obtain jets whose initial degree is greater than the jet-space degree k. The space $J^k(n,p)$ is identified with the space of p-tuples of polynomials in n indeterminates over $\mathbf{F}$, truncated to degree k. The spanning set in therefore given as a set of such polynomial vectors.

Step 2

The spanning set calculated in Step 1 is reduced to echelon form using Gaussian elimination. By ordering the monomial vectors $x_1^{i_1} \ldots x_n^{i_n} e_j \in J^k(n,p)$, each jet in $J^k(n,p)$ corresponds to a coordinate vector over $\mathbf{F}$ via extraction of coefficients. The spanning set obtained in Step 1 then corresponds to the matrix whose rows consists of these coefficient vectors. Reducing this matrix using Gaussian elimination gives a canonical basis for the tangent space. We actually use the technique of indexed Gaussian elimination, mentioned above and discussed in Section 3.2.

Step 3

A basis C for the complementary (normal) space to the tangent space is calculated. That is, the independent set obtained in Step 2 is extended to one of full rank in $J^k(n,p)$ by the addition of monomial vectors. In the $\mathcal{A}_e$ and $\mathcal{A}$ cases (for example) this gives the terms required in a versal unfolding and the corresponding codimension (for determined jets). In the $\mathcal{A}_1$ complete transversal case (for example) the monomial jets in $J^k(n,p)$ are ordered so that those of degree k correspond to the latter columns of the matrix. The monomial vectors in C of degree k will then form a basis for a complete transversal. This process can be generalised to deal with complete transversal calculations using a unipotent subgroup $\mathcal{G}$ and corresponding nilpotent filtration; see Section 3.2.

Step 4

Calculating a basis for the tangent space is the main computational overhead in the algorithm. During this procedure all by-products of the reduction process which may be of further use (such as the bases for the tangent and normal spaces, invariants such as the dimension and codimension of these spaces) are stored as global variables for access by other routines. Step 4 deals with output and manipulation of these results. A number of procedures are associated with Step 4 and perform such functions as displaying the bases, displaying a basis for a complete transversal, and testing whether a given set of vectors is independent from the tangent space (such questions arise in checking the hypotheses of Mather's Lemma and in moduli detection). The computational overhead of such procedures is negligible compared to that involved in Steps 1 – 3.

3.2 Technical and Computational Considerations

We will now describe some of the more important computational issues behind the algorithm.

3.2.1 Symbolic Pivots: Fraction-free Gaussian Elimination

Writing the package in a symbolic language such as Maple allows great flexibility. One notable advantage is that parameters (such as moduli) may be present in the jets we work with, thus allowing us to perform calculations for whole families. The matrix created in Step 2 will contain polynomial entries and we must take this into account during the Gaussian elimination routine. We choose numeric pivotal elements (in this context meaning 'constant polynomials') where possible, but when we are forced to choose a non-constant polynomial pivotal element *no* division is performed on the chosen row to reduce the pivot to unity. Division is still performed (working in the field of rational functions) when using the pivot to reduce the rest of the column to zero. This is in contrast to standard 'fraction-free' Gaussian elimination [12, p.82] where the pivot and the term it is to eliminate are multiplied up and no division occurs at all. Our method provides a valid elimination algorithm for jets involving parameters without the inconvenience of standard fraction-free elimination where the matrix entries rapidly blow-up into large expressions. The elimination only breaks down for certain values of the parameters for which a pivot vanishes, but the conditions determining this are retained. The list of all non-numeric pivots is stored for global access after the algorithm terminates, and may be examined by a procedure associated with Step 4.

The non-numeric pivots will, in general, be rational functions in the parameters, the vanishing of their numerators defining a finite set of proper algebraic varieties within the parameter space. The elimination applies to members of the family corresponding to values of the parameters not lying on these varieties, and the algorithm therefore determines the generic behaviour by default. To investigate the exceptional behaviour we must inspect each of the non-numeric pivots in turn, obtaining conditions on the parameters for which the elimination breaks down. In many cases (at least those with one or two parameters) the solutions can be determined explicitly using one of the Maple factor or solver procedures, the solutions being substituted back into the family and the calculation repeated. This process detects such phenomena as exceptional values in modular families, or cases where applications of Mather's Lemma break down thus obstructing triviality within the family but providing a finite list of normal forms.

3.2.2 Exploiting Sparsity: Indexed Gaussian Elimination

Working with a matrix of coefficient vectors in Step 2 is wasteful on memory and CPU time. By the very nature of the algorithm the data is created as a set of polynomial vectors (truncated to the prescribed degree k). This is a very efficient data structure to work with. Storage of the sparse data (the non-zero coefficients) is optimised, as is its manipulation. The idea is to work with the set of polynomial vectors and manipulate these directly using symbolic techniques, a coefficient matrix is never created. We use a set of indexing tables which, for a given row and column (i, j) of the would-be coefficient matrix, index the appropriate coefficient of the ith polynomial vector in our spanning set. The column j therefore indexes two pieces of information: the component of the vector and a monomial term in the resulting polynomial. During elimination, coefficients are looked-up from this set of polynomial vectors using the indexing tables and, for all intensive purposes, could be thought of as matrix entries. However, the row reduction operations performed in Gaussian elimination are now achieved by direct polynomial addition — a very efficient process in Maple which uses the internal kernel functions.

We had to completely rewrite the Gaussian elimination routine found in Maple in order to incorporate both the above method, and the type of fraction-free elimination described in the previous section. The resulting elimination algorithm proved, on average, to be two to three times faster, using three to four times less memory than methods which extract an explicit matrix of coefficients and apply the standard Maple library routines.

3.2.3 Exploiting Symmetry in the Target

The $\mathcal{L}$ and $\mathcal{C}$ 'type' tangent spaces exhibit a large degree of symmetry. Their respective action on a given germ f is given by,

$$f^*(\mathcal{M}_p^{t_2})\{e_1, \ldots, e_p\} \qquad \text{and} \qquad \mathcal{M}_n^{t_2} f^*(\mathcal{M}_p)\mathcal{E}(n,p).$$

In the $\mathcal{L}$ case we create a spanning set for the ideal $f^*(\mathcal{M}_p^{t_2})$ as a vector subspace of $J^k(n,1)$ and reduce this to echelon form using (indexed) Gaussian elimination. In the $\mathcal{C}$ case we do the same for the ideal $\mathcal{M}_n^{t_2} f^*(\mathcal{M}_p)$. We then produce a spanning set for the full $\mathcal{L}$ or $\mathcal{C}$ tangent space by stacking together p copies of the resulting 'matrix'. The important point is that we can do this in such a manner as to create a spanning set for the full tangent space which is *already* in echelon form and therefore requires no further elimination. This is clear if the matrices were stacked together to form a diagonal block matrix, but this corresponds to a specific ordering of the monomial vectors in $J^k(n,p)$. The monomial orderings required in certain problems, such as complete transversal calculations, do not give rise to such a simple diagonal block matrix, but the principle still applies and we indeed find that the matrix formed by stacking is automatically in echelon form. The reduction in computational overhead is clear.

Finally, in a problem dealing with $\mathcal{A}$-equivalence or $\mathcal{K}$-equivalence, this basis for the target tangent space is adjoined with a spanning set for the source tangent space and the resulting 'matrix' reduced to echelon form. Represent these as matrices of coefficient vectors, M_1 and M_2, respectively. The full tangent space matrix, formed by adjoining these,

$$\begin{pmatrix} M_1 \\ M_2 \end{pmatrix}$$

is reduced to echelon form. However, M_1 is already in echelon form and a full blown Gaussian elimination is replaced by the following algorithm. Keep the current row and column pointer in the matrix M_1. If the corresponding entry is a pivot then reduce as usual; only the column in M_2 needs to be reduced to zero as the column in M_1 will already be zero. Otherwise, (if the entry in M_1 is zero) try and find a pivot in M_2. If this is possible, again only the column in M_2 needs to be reduced. However, if we need to use M_2 to obtain a pivot then we do not *swap* the rows of M_1 and M_2 as in standard Gaussian elimination, but rather *insert* the row of M_2 into M_1 thus preserving the fact that M_1 is echelon. This is the basic idea at least. In the code it is more efficient to create a separate matrix which stores the final result: when a pivot is found the corresponding row is added to this 'result matrix' thus eliminating the need to insert a row of M_2 into M_1 (moving all the remaining rows of M_1 down). In addition, the process is carried out using the indexing tables referred to above, not coefficient matrices.

We remark that the presence of target tangent spaces and a target dimension greater than 1 make the computational overhead at the elimination stage *considerably* greater in $\mathcal{A}$ and $\mathcal{K}$ 'type' calculations, compared to $\mathcal{R}$ 'type' calculations. This exploitation of symmetry means that many significant calculations remain feasible.

3.2.4 Normal Spaces, Complete Transversals and Nilpotent Filtrations

It is an easy matter to extend the basis for $L \cdot f$ to one of full rank in $J^k(n,p)$, thus providing a basis C for the normal space. If the basis for $L \cdot f$ is given in coordinate form by the rows of the echelon matrix (a_{ij}) with pivotal elements a_{1j_1}, a_{2j_2}, $\ldots$, a_{rj_r} (so these are non-zero elements and $1 \le j_1 < j_2 < \cdots < j_r \le q = \dim J^k(n,p)$, where $r = \dim L \cdot f$), then the canonical vectors

$$\{e_1, \ldots, \hat{e_{j_1}}, \ldots, \hat{e_{j_2}}, \ldots, \hat{e_{j_r}}, \ldots, e_q\}$$

(where $\hat{e_j}$ denotes the exclusion of e_j from this set of vectors) form the basis C. Of course, we calculate C as the set of monomial vectors which correspond to these coordinate vectors e_i. The algorithm to derive C from (a_{ij}) also takes the opportunity to pick off all of the non-numeric pivots (discussed above) and store them for global access.

Calculating complete transversals requires a little more subtlety. Provided the columns corresponding to the monomial jets of degree k appear as a block at the end of the column, the above procedure will provide a basis for a degree k complete transversal associated with the standard filtration by degree (see Example 2.3). This basis simply consists of those elements in C of degree k. For this to work for the general complete transversal theorem 2.1 we must order the degree k monomial jets according to the nilpotent filtration, starting with those of degree $(k, 1)$, then those of degree $(k, 2)$, and so on. In most situations which arise in practice, this can be achieved via a system of weights. In what follows $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_p)$ will denote the source and target weights respectively. We recall the following; see [5, Section 2.3] for a full discussion on weighted filtrations. The monomial vector $x_1^{k_1} \ldots x_n^{k_n} e_i$ is assigned a weight $k_1\alpha_1 + \cdots + k_n\alpha_n - \beta_i$. The $\mathcal{E}_n$-submodule of $\mathcal{M}_n\mathcal{E}(n, p)$ generated by such monomial vectors of weight $\geq k$ is denoted $F^k_{\alpha,\beta}\mathcal{E}(n, p)$.

We consider the case of $\mathcal{A}$-classification, though the method extends to other subgroups of $\mathcal{K}$. Let $(x_1, \ldots, x_n)$ denote coordinates on $(\mathbf{F}^n, 0)$ and $(y_1, \ldots, y_p)$ those on $(\mathbf{F}^p, 0)$. Let $\mathcal{G}$ be a subgroup of $\mathcal{A}$ such that $L(J^1\mathcal{G})$ acts nilpotently on $\mathbf{F}^{n+p}$. For 'large enough $\mathcal{G}$' (we make this precise below) we can assign source and target weights such that the partition of the monomial vectors of (standard) degree k via their weight corresponds to their partition into the (k, s)-jet-levels using the nilpotent filtration. The following was proved in [5].

Proposition 3.1 *Suppose $L\mathcal{G}$ contains the following vectors and assign source and target weights according to the case in question.*

Vectors		*Weight*
$x_i\partial/\partial x_{i+1} \in L\mathcal{R}$	*or*	$\alpha = (n, \ldots, 2, 1)$
$x_{i+1}\partial/\partial x_i \in L\mathcal{R}$		$\alpha = (1, 2, \ldots, n)$
$y_j\partial/\partial y_{j+1} \in L\mathcal{L}$	*or*	$\beta = (0, -1, \ldots, -p+1)$
$y_{j+1}\partial/\partial y_j \in L\mathcal{L}$		$\beta = (-p+1, \ldots, -1, 0)$

for $i = 1, \ldots, n-1$ and $j = 1, \ldots, p-1$. Then

$$\sum_{i \geq s} (L\mathcal{G})^i \cdot (\mathcal{M}_n^k\mathcal{E}(n, p)) + \mathcal{M}_n^{k+1}\mathcal{E}(n, p) =$$
$$\big(F^{k+s}_{\alpha,\beta}\mathcal{E}(n, p) \cap \mathcal{M}_n^k\mathcal{E}(n, p)\big) + \mathcal{M}_n^{k+1}\mathcal{E}(n, p).$$

So for fixed k, the $M_{k,s}(\mathcal{G})$ filtration can be replaced by the weighted filtration modulo $\mathcal{M}_n^{k+1}\mathcal{E}(n,p)$, that is the filtration on the right-hand side of the above expression. In particular, the homogeneous monomial vectors of degree (k,s) (to be precise, those that span the space given by the image of $M_{k,s-1}(\mathcal{G})$ in the jet-space $J^{k,s}(n,p)$) are just the homogeneous monomial vectors of (standard) degree k with weight $k+s-1$.

The vectors referred to in Proposition 3.1 are the 'extra' vectors present in $L\mathcal{A} \setminus L\mathcal{A}_1$. For classification purposes one would prefer to use some unipotent group $\mathcal{G}$ such that the nilpotent Lie algebra $L\mathcal{G} \subset L\mathcal{A}$ contains as many of these 'extra' vectors as possible. There are four natural cases to consider:

$$L\mathcal{G} \;=\; L\mathcal{A}_1 \,\oplus\, \mathbf{F}\{x_i\partial/\partial x_j\} \,\oplus\, \mathbf{F}\{y_k\partial/\partial y_l\}$$

for all $i<j$ (or alternatively all $i>j$) and similarly for k and l. Such cases are used in practical applications (such as the examples in Chapter 5) and Proposition 3.1 applies.

Chapter 4

Reference Guide

We assume the reader is familiar with the background singularity theory as reviewed in Chapter 2. The basic setup of the package was described in Section 3.1 and we will make references to some of the features introduced there.

4.1 Getting Started

The `Transversal` package runs under Maple V, it has been tested successfully using the Release 3 and Release 4 systems. To date, it has not been tested under the latest Release 5, but compatibility problems are not expected. The package should run on most platforms, though the majority of development has been carried out using the Unix version of Maple (in particular, on Sun and SGI workstations).

For installation instructions see the README file which comes with the distribution. A simple Unix shell script which installs `Transversal` as a Maple library is provided. This script essentially builds the '.m' files and table which define the library. Having installed the package and defined the library paths appropriately (see the README file) the package may be loaded into any Maple session in the usual way via the command

```
with(transversal);
```

I have not written installation scripts for other platforms such as PCs or Macintoshs but this should be a simple task for anyone with the appropriate background (just translate the simple Unix scripts!). Alternatively, since the Maple source code for the library routines is provided as ASCII files the library may be loaded on any platform by performing a straightforward 'brute force' read of all the files in the package — an appropriate Maple file is provided to do this. Thus, the package should run under all platforms which support Maple, though the neat approach of installing it as a self loading library is only catered for under Unix at present.

4.2 Global Variables: The Initialisation Stage

An essential part of any problem tackled by `Transversal` is to specify the tangent space L. This requires several different items of data which, within a typical class of problems, remain fixed. We therefore specify L using a set of global variables which are accessed internally by the package's routines. The remaining items, which typically vary from calculation to calculation (for example, the germ under consideration or the jet-space degree) are passed to the routines via the usual parameter interfaces.

Thus, the first stage of any problem requires the user to assign these global variables accordingly. We begin by describing this process, in conjunction with the discussion in Section 3.1, and then give a more precise synopsis for each variable.

4.2.1 Brief Description

The global variable `equiv` may take the string constant values R, L, A, C or K to specify the 'type' of equivalence, as described in Section 3.1. The actual space L which `jetcalc` uses in any calculation depends on the other global variables.

The global variables `source_power` and `target_power` specify the powers to which the maximal ideals are to be raised in the defining equation for L. We refer to Section 3.1 where t_1 and t_2 represent the values held by `source_power` and `target_power`, respectively.

The space L can be decomposed into the direct sum of two components consisting of the source and target vector fields. In most applications (at least those to date) it is the source component which departs from the standard space associated with $\mathcal{R}$-equivalence; the module of vector fields tangent to a variety in the source space is a typical example. We therefore allow the source component of L to be user-specified as an $\mathcal{E}_n$-module of vector fields. (This may be extended to the target component in future releases but, at present, there is little call for such requirements, especially as the programming issues are non-trivial.) The user specifies a set of vector fields ξ_i which generate the source component of L as an $\mathcal{E}_n$-module. These fields take the form

$$\xi_i = g_1 \frac{\partial}{\partial x_1} + \cdots + g_n \frac{\partial}{\partial x_n} \qquad \text{where} \qquad g_j \in \mathcal{E}_n,$$

and are defined for the package via a procedure which takes a jet f as a parameter and returns $\xi_i \cdot f$. The global variable `liealg` holds the name of the procedure to be called. The exact Maple syntax for these vector fields is discussed below, together with the other functions of the `liealg` procedures. Several `liealg` procedures come shipped with the package. The standard one which specifies the 'pseudo right group', $\mathcal{R}_e$, is called `stdjacobian` and defines the generating set $\xi_i = \partial/\partial x_i$ where $i = 1, \ldots, n$. Others include `cusp`, `swallowtail` and `d4discrim` for the module of vector fields tangent to the respective discriminant varieties. Should any user need to write their own `liealg` procedure we recommend they use those provided as a template.

For complete transversal and determinacy techniques we often employ a unipotent subgroup of $\mathcal{K}$ and its nilpotent Lie algebra L may be specified as the sum of a 'standard' tangent space and a linear space spanned by a set of 'extra' nilpotent vectors. For example, in the $\mathcal{A}$ case the space L is given by the sum of $L\mathcal{A}_1$ and a space spanned by extra vectors belonging to $L\mathcal{A}\backslash L\mathcal{A}_1$. The global variables `R_nilp` and `L_nilp` are used to list these extra vectors, specifying those from the source and target, respectively. In addition, the 'pseudo' Boolean variable `nilp` indicates whether to include the extra vectors specified by `R_nilp` and `L_nilp` in the calculation, whether to ignore them, or whether to include them and additionally order the polynomial jets as dictated by the underlying nilpotent filtration $M_{k,s}(\mathcal{G})$. The latter case may be achieved for the nilpotent spaces L which are typically used in applications via a system of weights assigned to the source and target variables; see Section 3.2. That we can construct complete transversals from the basis for the complementary space relies on how we order the monomial jets, the main point being that `jetcalc` always orders the jets so that a complete transversal is given by taking the vectors of degree k from the basis for the complementary space. By ordering the homogeneous jets of degree k using the order induced by $M_{k,s}(\mathcal{G})$ this method extends to the nilpotent complete transversal methods. Even though `jetcalc` produces a complete transversal of degree k, we can obtain the appropriate $M_{k,s}(\mathcal{G})$ transversal by truncating the jets at that level — explicitly we just take the degree k terms in the complete transversal which belong to the required $M_{k,s}(\mathcal{G})$ jet-space; all the other degree k terms being ignored. The jet resulting from the complete transversal calculation must then be fed back into `jetcalc` using the *same* value of k, but now truncating the resulting transversal at a higher $M_{k,s}(\mathcal{G})$ level than before. The examples in Chapter 5 should make the process clear.

Remark. The above scheme is limited and does not allow one to specify the tangent space L for an arbitrary subgroup of $\mathcal{K}$. However, a reasonable compromise is reached in that virtually every case which comes up in applications may be defined using a straightforward method which is easy for the user to implement. One area where the above scheme could be improved is that of specifying a user-defined generating set of vector fields for the target component of L. At present only the source component is user-specified but one would need to specify the target component in certain applications, for example those which use Damon's $\mathcal{K}_V$-equivalence; see [11]. However, the use of the package in such areas is not a pressing requirement and is only likely to be addressed at some point in the future.

4.2.2 The Individual Global Variables

We now describe the individual global variables, it is the user's responsibility to assign these appropriately.

liealg

This holds the name of a Maple procedure called from within the main routine `jetcalc`. The main purpose of this procedure is to define a set of vector fields which generate the source component of L as an $\mathcal{E}_n$-module. It also specifies the names to be used for the source coordinates, this enables such coordinates to be distinguished from any unfolding parameters or moduli which may be present in the jet f. It may be convenient to assign, within the `liealg` procedure, any other global variables particular to the problem at hand. For example, the `liealg` procedures associated with classifications on discriminant varieties also assign the variables which specify the nilpotent setup. We remark on the following.

- The source coordinate names assigned within a `liealg` procedure must always be used in the defining equation for any jet f passed to `jetcalc`. For instance, `stdjacobian` uses $x_1, \ldots, x_n$ as source coordinates and their actual Maple names are defined to be the string constant values $x1, \ldots, xn$, that is juxtaposition of x with a number (technically speaking we are using the Maple concept of concatenation of names). When called, `jetcalc` prints out the coordinate names it is using to clarify this.
- The source coordinate names are required by several routines and are stored in the global variable `coords` for convenience.
- In the case `liealg` = `stdjacobian`, the source dimension n must also be specified. This is done by the global variable **source_dim**, a positive integer.

The rest of this section describes the technical details behind the `liealg` procedures and need only be read should the user want to write their own procedures.

A `liealg` procedure is defined with three formal parameters thus

`liealg_example` := proc(f,p,`tgtspace`)

where f, a list, stores the given jet passed to `jetcalc` and p, a positive integer, the target dimension deduced from the number of components of f. These are pre-determined in `jetcalc` before it calls the `liealg` procedure. The parameter `tgtspace` is of Maple type ‘table’ and is assigned within the procedure.

The source coordinates must be specified by assigning a Maple name to each entry in the global list `coords`. This is another function which must be carried out by the `liealg` procedure. It is a good idea at this stage to check the required names are unassigned as Maple expressions and return an error otherwise; see the routines which come with the package.

Next a generating set for the source Lie algebra must be specified using the table `tgtspace`. Each entry of `tgtspace` is *itself* of type ‘table’ with p components and corresponds a generator ξ_i, specifying how ξ_i acts on the jet f. The precise syntax is as follows. Suppose the i-th vector in the generating set is of the form

$$\xi_i = g_1 \frac{\partial}{\partial x_1} + \cdots + g_n \frac{\partial}{\partial x_n} \qquad \text{where} \qquad g_j \in \mathcal{E}_n,$$

then the i-th entry of `tgtspace` specifies the p components of $\xi_i \cdot f$ and is defined in Maple by

$$\begin{gathered}\texttt{tgtspace}[i][1] := g_1{*}\mathrm{diff}(f[1],\texttt{coords}[1]) + \cdots + g_n{*}\mathrm{diff}(f[1],\texttt{coords}[n]);\\ \vdots \\ \texttt{tgtspace}[i][p] := g_1{*}\mathrm{diff}(f[p],\texttt{coords}[1]) + \cdots + g_n{*}\mathrm{diff}(f[p],\texttt{coords}[n]);\end{gathered}$$

where f is given in Maple by a list of p entries, $f := [f_1, \ldots, f_p]$.

A warning is needed on the special case when the target dimension p is one. Since each entry `tgtspace`$[i]$ must itself be of type 'table', we must use expressions of the form

`tgtspace`$[i][1]$:= ... ;   and not
`tgtspace`$[i]$:= ... ;

Also, the use of Maple type 'list' on the right hand side does *not* work and the following should be avoided

`tgtspace`$[i]$:= [...];

Finally, the global variables which define the nilpotent terms may be assigned, if required. We refer to the sections below for descriptions of the various nilpotent variables and the required syntax.

equiv

This takes a value from one of the string constant values R, L, A, C or K and specifies which of the five 'types' the space L falls into. (Note that *capitals* must be used, and that if any of these letters are assigned Maple expressions then they must be evaluated to actual Maple names using single right-quotes, that is, using `equiv` := 'R' instead of `equiv` := R, in the $\mathcal{R}$ case, say.) The three types which specify a source component to L, namely R, A and K, will use the `liealg` procedure to define this component. However, the target component is always based on standard ones arising from $\mathcal{L}$ or $\mathcal{C}$ equivalence and can only be altered through the use of the variable `target_power`.

source_power/target_power

These are non-negative integers which specify the power by which the appropriate maximal ideal, $\mathcal{M}_n$ or $\mathcal{M}_p$, is to be raised. For example, consider the $\mathcal{A}$ case. Setting `equiv` := A; `liealg` := `stdjacobian`; gives the tangent space $L \cdot f$ as

$$\mathcal{M}_n^{\text{source\_power}}\langle \partial f/\partial x_1, \ldots, \partial f/\partial x_n \rangle + f^*(\mathcal{M}_p^{\text{target\_power}})\{e_1, \ldots, e_p\}.$$

The standard examples are therefore given by the following settings.

equivalence	source_power	target_power
$\mathcal{A}_e$	0	0
$\mathcal{A}$	1	1
$\mathcal{A}_1$	2	2
$\mathcal{A}_2$	3	3

Now consider the $\mathcal{K}$ case. Setting `equiv` := K; `liealg` := `stdjacobian`; gives the tangent space $L \cdot f$ as

$$\mathcal{M}_n^{\text{source\_power}}\langle \partial f/\partial x_1, \ldots, \partial f/\partial x_n\rangle + \mathcal{M}_n^{\text{target\_power}} f^*(\mathcal{M}_p)\mathcal{E}(n,p).$$

We can therefore obtain the following.

equivalence	source_power	target_power
$\mathcal{K}_e$	0	0
$\mathcal{K}$	1	0
$\mathcal{K}_1$	2	1
$\mathcal{K}_2$	3	2

Note that the phrases 'source' and 'target' refer to the component of L in which these exponents feature (that is, source: $\mathcal{R}$, target: $\mathcal{L}$ or $\mathcal{C}$) and *not* which ideal ($\mathcal{M}_n$ or $\mathcal{M}_p$) they apply to. Specifically, the source component of L is always multiplied by a power of the ideal $\mathcal{M}_n$, whereas the target component of L is multiplied by a power of the ideal $\mathcal{M}_p$ in the $\mathcal{L}$ and $\mathcal{A}$ cases, and by a power of $\mathcal{M}_n$ in the $\mathcal{C}$ and $\mathcal{K}$ cases.

compltrans

This is a Boolean variable. The main function `jetcalc` calculates a basis for the complementary space and stores it globally for access by the function `pcomp` which is used to display ('print') the basis; see Section 4.4. A complete transversal can be obtained from this basis simply by extracting the terms of degree k; see Section 3.2. Setting `compltrans` := `true` causes `pcomp` to extract the complete transversal from the basis and then print it out. To output the full complementary basis, required in unfolding theory, we set `compltrans` := `false` before calling `pcomp`. The value of `compltrans` has no effect on the actual calculation carried out within `jetcalc`.

nilp

This is a 'pseudo' Boolean variable. When set to `true` this tells `jetcalc` to include the nilpotent vectors specified by `R_nilp` and `L_nilp` in the calculation; this would be the case, say, for determinacy calculations which use a nilpotent space L. However, when set to `false` this tells `jetcalc` to ignore the variables `R_nilp` and `L_nilp`; this would be the case for standard unfolding calculations, say. For complete transversal calculations which use a nilpotent space L the homogeneous jets of degree k must be ordered as dictated by the nilpotent filtration $M_{k,s}(\mathcal{G})$. To achieve this we set `nilp` equal to `true_order` and assign the variables `nilp_source_wt` and `nilp_target_wt` with the appropriate weights, these being used to define the ordering. (For the other settings of `nilp` a default lexicographical ordering which ensures the homogeneous jets of degree k appear last in the order is employed. This ordering suffices for $\mathcal{A}_1$ complete transversal calculations, for example.) It is up to the user to make sure these weights are the correct weights to use with the nilpotent vectors given by `R_nilp` and `L_nilp`. However, the function `setup_Aclassn` which provides a setup for $\mathcal{A}$-classifications is helpful in this regard. It assigns all the global variables, in particular all the nilpotent variables are assigned appropriately; see Section 4.4.

In most situations, complete transversal and determinacy calculations are carried out back-to-back so the setting `true` is unlikely to be used.

As a safety device, `jetcalc` performs type checks on the global variables it uses and returns errors if necessary. However, note that it only checks those required for a particular setting of `nilp`. This is a convenience measure for the user — only the required variables need to be assigned correctly.

R_nilp/L_nilp

These are two variables of Maple type 'list'; each entry in the lists being a list with two entries. These variables specify the extra nilpotent vectors which are to be included in L. Nilpotent vectors in the source component of L will take the form $g\xi_i$, where $g \in \mathcal{E}_n$ and ξ_i is a generator for the source component of L. For example, in the $\mathcal{A}$ case we will require (some of the) vectors of the form $x_i\partial/\partial x_j$ to be included, but in general ξ_i will be some linear combination of the standard vectors $\partial/\partial x_j$. Nilpotent vectors in the target component of L will always be of the form $y_i\partial/\partial y_j$, where $(y_1, \ldots, y_p)$ are coordinates on $\mathbf{F}^p$. (In coordinate form this vector operates on a germ f via $(y_i\partial/\partial y_j) \cdot f = f^*(y_i)e_j$.)

The precise syntax is as follows. For $g \in \mathcal{E}_n$ and ξ_i the i-th vector specified by the table `tgtspace` (see `liealg` above), the entry $[g, i]$ in the list `R_nilp` indicates that the vector $g\xi_i$ is to be included in the source component of L. For example, with `liealg` = `stdjacobian`, to include $x_i\partial/\partial x_j$ we include the entry $[xi, j]$ in `R_nilp`. Similarly, the entry $[i, j]$ in the list `L_nilp` indicates that the vector $y_i\partial/\partial y_j$ is to be included in the target component of L. Note `jetcalc` extends the tangent space by including the $\mathbf{F}$-span of all of the nilpotent vectors.

As an example consider the $\mathcal{A}$ classification of map-germs $\mathbf{F}^2, 0 \to \mathbf{F}^3, 0$. We will use the unipotent subgroup of $\mathcal{A}$ with Lie algebra

$$L \;=\; L\mathcal{A}_1 \;\oplus\; \mathbf{F}\{x_1\partial/\partial x_2\} \;\oplus\; \mathbf{F}\{y_2\partial/\partial y_1, y_3\partial/\partial y_1, y_3\partial/\partial y_2\}$$

for the calculations. Only the scalar multiples of the nilpotent vectors

$$(x_1\partial/\partial x_2) \cdot f,\; (y_2\partial/\partial y_1) \cdot f,\; (y_3\partial/\partial y_1) \cdot f,\; (y_3\partial/\partial y_2) \cdot f,$$

are not already present in $L\mathcal{A}_1 \cdot f$. We must therefore specify $\mathcal{A}_1$ and then add the nilpotent vectors. This may be done as follows (recall that we must also specify the source dimension when we are using `liealg` = `stdjacobian`).

```
equiv := A;
liealg := stdjacobian;
source_dim := 2;
source_power := 2;
target_power := 2;
nilp := true;
R_nilp := [ [x1, 2] ];
L_nilp := [ [2, 1], [3, 1], [3, 2] ];
```

nilp_source_wt/nilp_target_wt

These are two variables of Maple type 'list'; each entry in the lists being an integer. When using complete transversal methods based on a nilpotent filtration these integers provide the weights required to order the homogeneous jets of degree k. The values of the weights depend on the nilpotent vectors specified by `R_nilp` and `L_nilp`; see Section 3.2 and above. It is up to the user to make sure these weights are compatible with `R_nilp` and `L_nilp`. However, the function `setup_Aclassn` discussed in Section 4.4 is helpful in this regard.

For the example $\mathbf{F}^2, 0 \to \mathbf{F}^3, 0$ above, the choice of `R_nilp` and `L_nilp` requires the following weights.

```
nilp := true_order;
nilp_source_wt := [2, 1];
nilp_target_wt := [−2, −1, 0];
```

jetcalc_verbosity

In addition to the global variables which define the space L we take this opportunity to mention that the global variable `jetcalc_verbosity` must also be set by the user before calling the function `jetcalc`. During a calculation `jetcalc` reports information relating to how the calculation is progressing and this variable controls the amount of information output; see Section 4.4.

4.3 Extensions to the Package

Extensions of the standard `Transversal` package have been developed as follows.

`Transversal_W` a version of the package which implements weighted filtrations and the corresponding weighted jet-spaces.

`Transversal_M` a version dealing with multigerms.

`Transversal_L` a version dealing with equivalence relations defined using lowerable diffeomorphisms in the source.

Remark. The packages `Transversal` and `Transversal_W` represent comprehensive projects developed over a number of years. The packages `Transversal_M` and `Transversal_L` are relatively new and apply to more specialist applications. Although these newer packages are essentially complete (and have been used by the author in several applications) they are not as comprehensive as the standard package and are not, at present, included in the standard distribution. I hope, time permitting, to make them available in the next release of the package.

These packages are used in much the same way as `Transversal`, indeed they share many of the functions present in the standard package. Several functions (notably `jetcalc`) have been extended accordingly, but retain the same name as in `Transversal` and are called and used in an analogous way. The packages are loaded in the usual way using the `with` command, the selection of the appropriate package functions (whether the standard or extended versions) are chosen automatically.

For the most part the details for the standard `Transversal` package apply to these extended packages. There are slight differences in the setup of the packages (for example, with the global variables) and these are discussed in this section. Unless otherwise stated, the actual routines present in each package are used and function in exactly the same way as the routines in the standard `Transversal` package (indeed, in many cases the same routines are shared). When differences do occur they are described in Section 4.4 under the heading *Package Extensions.*

4.3.1 Weighted Filtrations

`Transversal` uses the standard filtration of $\mathcal{M}_n.\mathcal{E}(n,p)$ by degree, that is by the submodules $\{\mathcal{M}_n^{k+1}.\mathcal{E}(n,p)\}$ for $k \geq 0$, and performs calculations in the standard jet-space $J^k(n,p)$. Methods which use nilpotent filtrations $M_{k,s}(\mathcal{G})$ may also be implemented, but many problems naturally lend themselves to the use of weighted filtrations. The package `Transversal_W` implements filtrations given by weights and allows us to work with weighted jet-spaces. (For a description of classification techniques using weighted filtrations, in particular the complete transversal method, we refer to [5, Section 2.3].)

Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ be a sequence of positive integers, $\beta = (\beta_1, \ldots, \beta_p)$ be a sequence of non-negative integers, and $\{F^r_{\alpha,\beta}.\mathcal{E}(n,p)\}$ be the corresponding filtration of $\mathcal{M}_n.\mathcal{E}(n,p)$ by weight, which we will denote $\{F^r\}$; see [5]. The global variables which define L are used in *exactly* the same way as described in Section 4.2 except the global variables `nilp_source_wt` and `nilp_target_wt` are not used and `nilp` is a *real* Boolean variable, taking only the values `true` or `false`. In addition the source and target weights, α and β, are specified using the global variables `source_wt` and `target_wt`, both of Maple data type 'list'.

Given a germ f and weighted degree k, the function `jetcalc` calculates $L{\cdot}f$ and its normal space in the weighted jet-space using Gaussian elimination. This is the same algorithm as described in Section 3.1, except now the filtration $\{F^r\}$ is used and all polynomials are truncated by weight and not by standard degree. (A note on the algorithm: the version of `jetcalc` currently distributed in `Transversal_W` implements indexed Gaussian elimination but does not use the techniques which exploit symmetry in the target; see Section 3.2. This may be addressed in the future but has not caused any problems so far, possibly because the weighted jet-spaces are usually generated by considerably fewer monomial jets than the standard jet-space, that is, the weighted filtration $\{F^r\}$ is generally a lot finer than the standard filtration by degree.)

4.4 Package Functions

Here we describe all the user functions available in the package. The synopsis for each function is intended for reference, so there is inevitably some duplication of the notes for different functions.

Remark. In what follows the jet-space $J^k(n,p)$ is identified with the space of p-tuples of polynomials in n indeterminates over $\mathbf{F}$, truncated to degree k. The term 'jet' will refer to a member of $J^k(n,p)$ and will be represented by a Maple list of polynomials, that is formally as the Maple expression $[f_1,\ldots,f_p]$ where f_i are polynomial expressions in the n source coordinates (as specified by the global variable `coords`).

Function: classify — apply the method of complete transversals to verify determinacy criterion or find the first non-empty complete transversal for a given jet

Calling Sequence:

`classify`(f, k_1, k_2)

Parameters:

f	— a jet
k_1, k_2	— positive integers, $k_1 \leq k_2$

Synopsis:

- The parameter f represents a jet, as described at the start of this section.
- The function applies the inductive complete transversal classification method. It calculates successive transversals for the jet f from degree k_1 up to degree k_2, checking that they are empty. If this fails then the *first* non-empty transversal is output (and the function terminates) thus providing the first non-trivial family of jets lying over f (viewed as a $(k_1 - 1)$-jet).
- The tangent space to the orbit through f in the jet-space is $L \cdot f$ where L is a space of vector fields specified via the appropriate global variables. This is discussed fully in Section 4.2. Note that it is the user's responsibility to specify a space L which is suitable for complete transversal calculations.
- The function essentially makes repeated calls to the routine `jetcalc` and checks the resulting transversals. All the usual functions which access the results stored by `jetcalc` may be used. The results only apply to the jet-space being used when `classify` terminated, the degree of this is stored as the variable `jetspace_deg`.
- The function is particularly useful for checking determinacy criteria. In the case of $\mathcal{R}$-equivalence, for example, it is enough to verify the $(k+1)$-transversal is empty to prove k-$\mathcal{R}$-determinacy (taking $L = L\mathcal{R}_1$ or, more generally, L some nilpotent subspace of $L\mathcal{R}$) and the use of `jetcalc` suffices. However, in applications where $L\mathcal{G}$ is not a module over $\mathcal{E}_n$ it is necessary to check several levels of transversals. As an example we consider the important case of $\mathcal{A}$-determinacy and refer to the notes in Chapter 2, in particular to Theorem 2.5. Recall, in the notation introduced there, if

$$\mathcal{M}_n^{k+1}\mathcal{E}(n,p) \subset L\mathcal{H} \cdot f + \mathcal{M}_n^{k+1} f^*(\mathcal{M}_p)\mathcal{E}(n,p) + \mathcal{M}_n^{2k+2}\mathcal{E}(n,p)$$

then f is k-$\mathcal{A}$-determined. Thus, in general, it is enough to take $k_1 = k+1$ and $k_2 = 2k+1$: it all transversals are empty then the determinacy criterion holds, otherwise the *first* non-trivial family of jets lying over f is obtained. Of course, in practice it is usually possible (and very desirable) to work with a much smaller upper bound k_2. The term $\mathcal{M}_n^{k+1} f^*(\mathcal{M}_p)\mathcal{E}(n,p)$ allows this: if $\mathcal{M}_n^{k+1} f^*(\mathcal{M}_p) \supset \mathcal{M}_n^{l+1}$ for some l, then we may take $k_2 = l$. For example, if $f : \mathbf{F}^2, 0 \to \mathbf{F}^p, 0$ has as two of its coordinate functions x and $y^2 + xg(x,y)$ where x, y denote coordinates in the source and g is some function-germ in x, y, then $\mathcal{M}_2^{k+1} f^*(\mathcal{M}_p) \supset \mathcal{M}_2^{k+3}$ and only the complete transversals of degree $k+1$ and $k+2$ need to be calculated.

- It may happen that `classify` terminates with a non-empty transversal even though the jet is determined. For example, the terms in this transversal may belong to $\mathcal{M}_n^{k+1} f^*(\mathcal{M}_p)\mathcal{E}(n,p)$. (Experience indicates this particular scenario is quite rare, even so it is relatively easy to spot.) If determinacy indeed holds, but one accepts a non-empty transversal and continues with the classification, then further investigations (using techniques such as Mather's Lemma which incorporate the *whole* of the group $\mathcal{A}$) should reveal triviality for the family and (eventually) lead to determinacy. However, such considerations are rarely needed, even when they are continuing the classification will reveal the answers fairly quickly. In short, the complete transversal and determinacy techniques which `Transversal` implements are both efficient and effective, but not entirely foolproof — in a few cases one may need to work a little harder to obtain the *sharpest* bound for the determinacy degree.

- The function `classify` merely provides a convenient way of checking several successive transversals, it is usually the preferred method for checking determinacy criteria. Generally one does not know, a priori, that a jet is determined so it is more efficient to check the criteria using the method of complete transversals (which provide the next stage in the classification should determinacy fail) rather than directly. However, as noted in the previous point, failure of the determinacy condition does not necessarily imply the jet is not finitely determined. In some cases the function `determined` which checks the determinacy criterion directly may be successful and more appropriate. In addition, if the jet contains moduli then it is necessary to examine the 'check list' for degenerate behaviour at each jet-level (see `jetcalc`) and here it is more appropriate to make each call to `jetcalc` individually rather than use `classify` to perform a succession of calls.

See also: `determined`, `jetcalc` (and related functions).

Function: determined — test determinacy condition for a given jet

Calling Sequence:

`determined`(f, r, k)

Parameters:

f — a jet

r, k — positive integers, $r < k$

Synopsis:

- Determinacy conditions are typically of the form: if

  $$\mathcal{M}_n^{r+1}\mathcal{E}(n,p) \subset L \cdot f + \mathcal{M}_n^{k+1}\mathcal{E}(n,p)$$

  then f is r-$\mathcal{G}$-determined. Here L is an (appropriate) subspace of $L\mathcal{G}$ and r, k are integers suitable for the determinacy problem in question. The function `determined` tests such conditions.

- The parameter f represents a jet, as described at the start of this section.

- The integers r and k in the above determinacy condition coincide with the parameters r and k in the function call.

- The tangent space to the orbit through f in the jet-space is $L \cdot f$ where L is a space of vector fields specified via the appropriate global variables. This is discussed fully in Section 4.2. Note that it is the user's responsibility to specify a space L which is suitable for complete transversal calculations.

- The above determinacy condition may be reduced to a finite-dimensional problem within the jet-space $J^k(n,p)$. The function `determined` makes use of this fact, firstly calling `jetcalc` to calculate $L \cdot f$ in $J^k(n,p)$, and then checking the determinacy condition (via the function `intangent`). Thus, r indicates the determinacy degree to check and k the jet-space to work in. If the condition holds then a message is output to confirm this, otherwise the terms which fail the condition are output (these will be monomial vectors in $\mathcal{M}_n^{r+1}\mathcal{E}(n,p)$). All the usual functions which access the results of `jetcalc` may be used. In addition, any terms which fail the determinacy condition may be recalled with the function `pdetterms` (the data is stored as the global variable `det_store`).

- The choice of a suitable jet-space $J^k(n,p)$ depends on the determinacy condition in question. In the case of $\mathcal{R}$-equivalence, for example, it is enough to take $k = r + 1$ to prove k-$\mathcal{R}$-determinacy (taking $L = L\mathcal{R}_1$ or, more generally, L some nilpotent subspace of $L\mathcal{R}$). However, in applications where $L\mathcal{G}$ is not a module over $\mathcal{E}_n$ it is necessary to work in a jet-space of higher degree. As an example we consider the important case of $\mathcal{A}$-determinacy and refer to the notes in Chapter 2, in particular to Theorem 2.5. Recall, in the notation introduced there, if

$$\mathcal{M}_n^{r+1}\mathcal{E}(n,p) \subset L\mathcal{H}\cdot f + \mathcal{M}_n^{r+1}f^*(\mathcal{M}_p)\mathcal{E}(n,p) + \mathcal{M}_n^{2r+2}\mathcal{E}(n,p)$$

then f is r-$\mathcal{A}$-determined. Thus, in general, it is enough to take $k = 2r+1$ and show that all the monomial jets in $J^k(n,p)$ of degree $r+1$ and greater belong to $L\mathcal{H}\cdot f$. Of course, in practice it is usually possible (and very desirable) to work with a much smaller degree than $k = 2r+1$. The term $\mathcal{M}_n^{r+1}f^*(\mathcal{M}_p)\mathcal{E}(n,p)$ allows this: if $\mathcal{M}_n^{r+1}f^*(\mathcal{M}_p) \supset \mathcal{M}_n^{l+1}$ for some l, then we may take $k = l$. For example, if $f : \mathbf{F}^2, 0 \to \mathbf{F}^p, 0$ has as two of its coordinate functions x and $y^2 + xg(x,y)$ where x, y denote coordinates in the source and g is some function-germ in x, y, then $\mathcal{M}_2^{r+1}f^*(\mathcal{M}_p) \supset \mathcal{M}_2^{r+3}$ and we may work in $J^{r+2}(n,p)$.

- It may happen that the determinacy criterion checked by `determined` fails even though the jet is determined. A simple example occurs when the terms reported by `determined` (as failing to lie in $L\cdot f$) belong to $\mathcal{M}_n^{r+1}f^*(\mathcal{M}_p)\mathcal{E}(n,p)$; here we may still conclude that f is r-determined. Generally such considerations are rare, but should they arise it is usually easy to spot whether the terms belong to $\mathcal{M}_n^{r+1}f^*(\mathcal{M}_p)\mathcal{E}(n,p)$ or not. In general, the complete transversal and determinacy techniques which `Transversal` implements are both efficient and effective, but not entirely foolproof — in some cases one may need to work a little harder to obtain the *sharpest* bound for the determinacy degree. If one continues with the classification for a determined jet then further investigations (using techniques such as Mather's Lemma which incorporate the *whole* of the group $\mathcal{A}$) should (eventually) lead to the required determinacy result. However, such considerations are rarely needed and even when they are continuing the classification will reveal the answers fairly quickly.

- Generally one does not know, a priori, that a jet is determined so it is more efficient to check determinacy criteria via the method of complete transversals (which provide the next stage in the classification should determinacy fail) using the function `classify` rather than directly with the function `determined`. However, as noted in the previous point, failure of the determinacy condition does not necessarily imply the jet is not finitely determined, and in some cases the use of `determined` may be successful and more appropriate.

See also: `pdetterms`, `classify`, `jetcalc` (and related functions).

Function: **intangent — test if a set of vectors is in the tangent space calculated by jetcalc**

Calling Sequence:

`intangent`$(v_1, v_2, \dots)$

Parameters:

$v_1, v_2, \dots$ — jets

Synopsis:

- Each parameter v_i specifies a jet in $J^k(n, p)$. The function tests to see if the set of jets $\{v_1, v_2, \dots\}$ together with the basis for the tangent space to the orbit of a jet (already calculated by `jetcalc`) form a dependent set of vectors. It returns `true` when a dependent set results and `false` when an independent set results. The function can be used for testing the hypotheses to Mather's Lemma and for moduli detection; see Chapter 2.

- For a single parameter v the function therefore returns `true` if v is in the tangent space, and `false` if not. This is useful for remembering which way around `intangent` works — that is `true` for 'in tangent space' (or 'dependent'). Also, in the case of a single v a simple method is sometimes worth exploiting. If v is in the basis for the complementary space, as given by `pcomp`, then v cannot be in the tangent space. (However, if v is not in the basis this does not necessarily mean that v is in the tangent space, in this case `intangent` must be used.)

- The function `jetcalc` must always have been called before using this function in order to actually calculate the tangent space. That is the stage at which the jet and required jet-space $J^k(n, p)$ are determined. Only the most recently calculated tangent space is used.

- The function calculates a set of vectors (jets) by which the tangent space basis must be extended to give a basis for the direct sum of the tangent space and the space spanned by the v_i. To be precise, a matrix is calculated whose rows give these 'extension' vectors in coordinate form; this matrix is stored as the global variable `ext_tangent`. (Thus, if the rank of `ext_tangent` is less than the number of parameters v_i then `true` is returned; if these numbers are equal then `false` is returned.) This matrix is obtained by first forming the matrix whose rows are the coordinate vectors representing the v_i. Each row is then individually reduced using the basis vectors for the tangent space (calculated by `jetcalc` and in echelon form). Gaussian elimination is then performed on the resulting matrix to reduced it to echelon form. It is a simple matter to observe that these (row) vectors extend the basis of the tangent space to one of the direct sum.

- The matrix `ext_tangent` may contain non-numeric elements (say, if the original jet passed to `jetcalc` contained unfolding parameters or moduli) and the rank may drop for certain values. This is a similar situation to that which arises for the main reduction algorithm; see Section 3.2. In such cases, `intangent` prints a warning and outputs the matrix `ext_tangent` to allow the user to determine the degenerate situations by examining the pivotal elements. Since `ext_tangent` is a global variable it may be inspected at any later stage with the standard Maple commands for printing matrices. Note that printing `ext_tangent` will only give the vectors in coordinate form (and this depends on how `jetcalc` decided to order the monomial jets in $J^k(n,p)$). However, this is all the information that is needed and suffices for the above considerations.

Function: jetcalc — calculate the tangent space and complementary (normal) space to the orbit of a jet in a given jet-space

Calling Sequence:

`jetcalc`(f, k)

Parameters:

f — a jet

k — a positive integer

Synopsis:

- The parameter f represents a jet, as described at the start of this section. The integer k specifies the jet-space degree (thus f belongs to $J^k(n,p)$, truncating if necessary).

- The tangent space to the orbit through f in $J^k(n,p)$ is $L \cdot f$ where L is a space of vector fields specified via the appropriate global variables. This is discussed fully in Section 4.2.

- The tangent space $L{\cdot}f$ is calculated, together with the normal space (also referred to as complementary space) to $L \cdot f$ in $J^k(n,p)$. Specifically, a spanning set for $L \cdot f$ is calculated and then reduced to echelon form using Gaussian elimination to produce a basis. (Here we regard the vectors in the spanning set as coordinate vectors forming the rows of a matrix which is then reduced.) This basis is then extended to a basis for $J^k(n,p)$, thus providing a basis for the normal space. These bases are stored for inspection by the various 'print' routines described elsewhere in this section. (The actual data is stored in a specific format as the global variables `tgtspace` and `compbasis` respectively.) The note found at the end of this synopsis details all the 'print' functions which are available.

- During the reduction process it may not always be possible to choose pivotal elements which are numeric, instead polynomial expressions (or more precisely rational functions) involving unfolding parameters or moduli present in the jet f may be forced upon us. For specific values of the parameters the pivotal elements may vanish and the tangent space degenerates. The calculation fails for such values and `jetcalc` must be called again, substituting such values for the unfolding parameters. A 'check list' of all non-numeric pivotal elements is created by `jetcalc` and may be accessed by the routine `plist` after `jetcalc` has returned. (The check list data is stored as the global variable `checklist`.) A warning is printed by `jetcalc` when the check list is non-empty to remind the user to examine the degenerate behaviour.

- Technical note relating to the previous point: the Gaussian elimination routines in `jetcalc` use the Maple function `normal` when reducing each row. This is a basic form of simplification (chosen for efficiency purposes) which recognises those expressions equal to zero which lie in the domain of "rational functions". For expressions containing subexpressions such as square roots, powers, and functions, `normal` may not recognise when an expression is equal to zero. Thus, `jetcalc` should not be called with the parameters (see the previous point) substituted for exceptional values such as square roots and other (non-integer) powers. This restriction could be overcome by re-writing the elimination routines to use the Maple function `simplify`. However, when the entries in the checklist become complicated (this is especially the case when more than one modulus is present) investigation of the exceptional values is generally quite limited.

- The dimension of the complementary space is stored as the global variable `codim`, while the dimension of the tangent space is stored as `basis_dim`. Note that the complementary space is calculated as the normal space to $L \cdot f$ in $J^k(n,p)$. The calculation automatically includes the constant jets which may therefore need to be discarded in certain applications. The jet-space degree used in the last call to `jetcalc` is stored as `jetspace_deg`, mainly as a convenience and for access by a couple of the other routines.

- The source and target dimensions are both calculated by `jetcalc` from the number of components of `coords` and of f respectively. They do not need to be user-defined. However, in the case `liealg` = `stdjacobian` the source dimension must be specified by the user via the global variable `source_dim`, this enables the `liealg` routine to construct the list `coords` accordingly; see Section 4.2.

- The routine displays information relating to the calculation it is performing. The amount of information relayed may be controlled by the global variable `jetcalc_verbosity` which takes integer values. The setting 0 forces `jetcalc` to operate in silence; *note that with this setting the warning for a non-empty 'check list' is suppressed also.* The setting 1 causes `jetcalc` to output a small amount of information relating to the jet it is working with, the jet-space degree, the source coordinates defined by `coords`, and so forth. Although this information is readily available to the user it is worth while repeating it and the user double checking that `jetcalc` is performing precisely the intended calculation as careless errors can arise from calling `jetcalc` with incorrect arguments. (Of course an incorrect global setup can be just as disastrous but these settings rarely get modified within a typical calculation so it is left to the user to check them at the outset.) The setting 1 is recommended for standard usage; note that the warning for a non-empty 'check list' is displayed in this case. The settings 2 and 3 cause additional information relating to the progress of the calculation to be output. With the setting 2 each separate stage of the algorithm ($\mathcal{R}$ space, $\mathcal{L}$ space, Gaussian elimination, etc.) is reported. In particular, the elimination stage is by far the most computationally demanding and reporting the size of the matrices involved gives an idea of how long the calculation may take. (However, do not be too concerned about the sizes of some of these matrices, they are exceptionally sparse and the elimination techniques can often reduce seemingly large matrices efficiently. The matrix dimensions are given merely as a relative guide to calculation time.) For the more demanding calculations the setting 3 may be appropriate, here the row being dealt with by the Gaussian elimination process is reported, giving some idea about progress at the main computational stage.

Package Extensions: `Transversal_W`

- The function `jetcalc` in `Transversal_W` calculates the tangent space $L \cdot f$ and complementary space to $L \cdot f$ in a given *weighted* jet-space.
- The parameter f represents a jet (with respect to weighted degree) and the parameter k specifies the weighted degree of the jet-space.
- The space L is specified by the appropriate global variables as described in Section 4.2. The source and target weights which define the weighted filtration $\{F^r_{\alpha,\beta}\}$ of $\mathcal{M}_n.\mathcal{E}(n,p)$ are given by the global lists `source_wt` and `target_wt`.
- Otherwise, the same details for the standard version of `jetcalc` apply. (Note that the standard and weighted versions of `jetcalc` store all the data required by the other routines in common global variables so that the same routines can be used by both.)

Package Extensions: `Transversal_M`, `Transversal_L`

- Under development!

See also: `intangent`, `pcomp`, `plist`, `pmons`, `ptangent`.

Function: pcomp — print basis for complementary (normal) space calculated by jetcalc

Calling Sequence:

`pcomp()`

Synopsis:

- Outputs a basis for the complementary space to the tangent space to the orbit of a jet.
- The function `jetcalc` must always have been called before using this function in order to actually calculate the tangent and complementary spaces. That is the stage at which the jet and required jet-space $J^k(n,p)$ are determined. Only the most recently calculated basis is stored.
- Note that the complementary space is calculated as the normal space in $J^k(n,p)$ and, if necessary, therefore contains the constant jets. These may need to be discarded in certain applications.
- The dimension of the complementary space is stored as the global variable `codim` (but again note the previous comment regarding constant jets).
- If the global variable `compltrans` is set to `false` the whole basis is output. If `compltrans` is set to `true` only the degree k terms, where k was the degree passed to `jetcalc` and stored as `jetspace_deg`, are output. If no terms are output then a message indicating the normal space is empty is printed instead. In the weighted version the same applies, only this time to the terms of weighted degree k.
- For complete transversal calculations which use a nilpotent space L one calculates the transversal (working in $J^k(n,p)$, say) using `jetcalc` and displays it using `pcomp`, as described above. For simplicity `pcomp` outputs all the degree k terms; the user needs to identify those belonging to the required (k,s)-sublevel in order to determine the (k,s)-transversal. The function `pmons` may be helpful in this regard.

Function: pdetterms — print terms which failed the determinacy criterion as calculated by the function determined

Calling Sequence:

```
pdetterms()
```

Synopsis:

- Outputs the terms which failed the determinacy criterion checked using the function `determined`. For a full description see `determined`.
- The function `determined` must always have been called before using this function.

Function: plist — print check list calculated by jetcalc

Calling Sequence:

`plist`(flag$_1$, flag$_2$, flag$_3$)

Parameters:

flag$_1$, flag$_2$, flag$_3$ — (optional) flags which may take the values 'A', 'N' or 'P'.

Synopsis:

- Outputs the 'check list' calculated by `jetcalc`. The check list contains all the non-numeric pivotal elements which were formed when `jetcalc` performed Gaussian elimination; see `jetcalc`.
- The function `jetcalc` must always have been called before using this function in order to actually calculate the check list associated with the particular calculation. Only the most recently calculated check list is stored.
- The data is stored globally as the table `checklist`. Each element of `checklist` represents a monomial jet in $J^k(n,p)$, multiplied by some coefficient which is a rational function in the unfolding parameters or moduli present in the original jet passed to `jetcalc` (specifically, these external parameters differ from the source coordinates). In this context, by monomial jet we mean a list with one entry a monomial in the source variables (as specified by `coords`) and the remaining entries zero. The precise storage format is as follows. Each element of `checklist` is itself a table with two entries. The first, and most important, is the coefficient which, by the definition of the check list, will be a non-constant rational function. The second entry gives the corresponding monomial jet.
- Each entry in `checklist` is output by `plist` as follows. Firstly the index number of the entry (as an element of the table) is output, preceeded by a '#' symbol. The coefficient and monomial components of the entry are then output alongside the index, the precise format depending on the flags present in the calling sequence. The index is output so that the corresponding entry in `checklist` may be obtained by the user from within the Maple session once the function has returned. Thus, to obtain the coefficient component of the entry in `checklist` output as $\#i$ we refer to `checklist`[i][1]. The entry `checklist`[i][2] refers to the corresponding monomial component. (Note that the coefficients `checklist`[i][1] are just the non-numeric pivotal elements in the echelon matrix produced by `jetcalc`.)
- Flags may be passed as parameters to `plist` to modify the default output behaviour. These flags are optional and any number from none to three may be present in the function call. The flags take the string constant values A, N or P. It may be necessary to use single right-quotes (thus: 'A', 'N' or 'P') to force these to evaluate to a name if the required name has already been assigned. Each flag is described individually below.

- The most important and useful component of a checklist entry is the coefficient. By default `plist` only outputs this component. In most applications the corresponding monomial component is of no interest to the user, but should it be required then the ‘all’ flag A forces `plist` to output all components.

- In most applications where the user inspects the check list they are looking to determine the exceptional values, that is where a pivotal element vanishes rendering reduction invalid (such situations indicate that the tangent space may degenerate). The default behaviour is for `plist` to apply the Maple ‘factor’ routine to each of the coefficients in `checklist`. For completeness the ‘no factor’ flag N is included, this causes `plist` to output the coefficients ‘as is’ without trying to factor them. Should the user want to inspect the coefficients and try to simplify them manually from within the Maple session they can always be accessed directly as `checklist`[i][1], as described above.

- If the ‘pause’ flag P is passed to `plist` then the entries in `checklist` are printed out in turn with `plist` pausing, waiting for the user to type C;[RETURN] (that is, C followed by a semi-colon followed by the [RETURN] key — for Continue) before proceeding with the next entry. Alternatively typing E;[RETURN] (for Exit) terminates the function. (Again single right-quotes may be required thus: ’C’ or ’E’.)

- Historical notes. The ‘pause’ flag P is an older feature which is probably no longer required. It dates back to when `Transversal` was being developed using the standard Maple session which provides a text interface. A Maple session which uses a window manager interface will not require this flag. On older versions the default behaviour was not to factor the coefficient and to output both components of a `checklist` entry. This has been changed to reflect typical usage.

Function: pmons — print monomial jets

Calling Sequence:

`pmons`(k)

`pmons`(k_1, k_2)

Parameters:

k, k_1, k_2 — non-negative integers

Synopsis:

- In the first call the function outputs the monomial jets of degree k. If two parameters, k_1 and k_2, are present, as in the second call, then all monomial jets of degree k_1 to degree k_2 are output. By monomial jet we mean a list with one entry a monomial in the source variables (as specified by `coords`) and the remaining entries zero.

- The jets are output in order of increasing degree, using the same order as that implemented by `jetcalc` when creating the matrix of coefficient vectors. Thus, if the global variable `nilp` is set to `true_order` then the order induced by the underlying nilpotent filtration and determined by the global lists `nilp_source_wt` and `nilp_target_wt` is used. The monomial jets of each degree k (where k is given or $k_1 \le k \le k_2$, as appropriate) are partitioned into their appropriate (k, s)-level and output along with an indication of this level. (Note that the order for monomials belonging to the same level is the same as that used by `jetcalc` but is otherwise not important.) If the global variable `nilp` is set to `false` (or `true`) then the default lexicographical order implemented by `jetcalc` is used.

- This function is particularly useful for complete transversal calculations which use a nilpotent space L. One calculates the transversal (working in $J^k(n, p)$, say) using `jetcalc` and displays it using `pcomp`. For simplicity `pcomp` outputs all the degree k terms, the user needs to identify those belonging to the required (k, s)-sublevel in order to determine the (k, s)-transversal. The function `pmons` removes the tedium of calculating the nilpotent filtration explicitly. Note that the possible 'nilpotent orders' are restricted to the standard ones obtained by weights, as mentioned in Section 3.2. However, this should be more than adequate in applications.

- This function requires no preliminary function calls or assigned global variables except those mentioned above and the use of `coords` to specify the coordinates. The global list `coords` may be assigned by the user, but note that any `liealg` procedure will (re-)set `coords` when it is called. (These procedures are usually called by `jetcalc` but this requires all the global variables to be assigned beforehand.)

Package Extensions: `Transversal_W`

- The version of `pmons` in the package `Transversal_W` takes the same calling sequence but instead outputs weighted monomial jets. In the first call the function outputs the monomial jets of weighted degree k. If two parameters, k_1 and k_2, are present, as in the second call, then all monomial jets of weight k_1 to weight k_2 are output.
- The jets are output in order of increasing weight, using the same order as that implemented by `jetcalc` (the version in `Transversal_W` of course!) when creating the matrix of coefficient vectors.
- The weighted degree of a monomial is defined in the usual way. The weights for the source and target variables must be specified via the global lists `source_wt` and `target_wt`. In the case of weighted homogeneous functions (that is, where the target has dimension one) the standard weighted degree involving only source weights may be specified by `source_wt` with `target_wt` equal to the list `[0]`.
- This function is particularly useful for complete transversal calculations which use a weighted filtration. Although the weighted version of `jetcalc` will perform the calculations using this filtration it is often helpful to be able to list all the monomial jets of a given weight. The function removes the tedium of calculating the weighted filtration explicitly.
- Otherwise, the same details for the standard version of `pmons` apply.

Function: ptangent — print basis for tangent space calculated by jetcalc

Calling Sequence:

`ptangent`(v)

Parameters:

v — a (monomial) jet, optional

Synopsis:

- Outputs a basis for the tangent space to the orbit of a jet. The basis is canonical in the sense that the coordinate form of each vector is just a row from the echelon matrix produced by `jetcalc` using Gaussian elimination.
- The function `jetcalc` must always have been called before using this function in order to actually calculate the tangent space. That is the stage at which the jet and required jet-space $J^k(n,p)$ are determined. Only the most recently calculated basis is stored.
- The dimension of this vector space is stored as the global variable `basis_dim`.
- The basis is output as follows. By monomial jet we mean a list with one entry a monomial in the source variables (as specified by `coords`) and the remaining entries zero. For each vector of the basis, a collection of monomial jets is output, each with a coefficient (numeric or possibly a rational function involving any unfolding parameters or moduli present in the jet passed to `jetcalc`). The actual basis vector is given by the corresponding linear sum formed by these coefficients and vectors.
- If the optional parameter v is present then only vectors in the basis which contain v as a term (as described in the previous point) are output. This is useful for a closer inspection of the tangent space. Note that v must be a *monomial* jet, otherwise nothing will be output.

Package Extensions

- This function is not presently available in the package `Transversal_M`. However, we remark that the standard version of `ptangent` works for the other extensions to the package, is automatically loaded into each package, and may be used as normal.

Function: pvars — print global variables

Calling Sequence:

`pvars()`

Synopsis:

- Outputs the current values of all the user-defined global variables which define the space L. These are discussed fully in Section 4.2. Specifically, the values of the following variables are output.

  `liealg`, `equiv`, `compltrans`, `source_dim`, `source_power`, `target_power`, `nilp`, `R_nilp`, `L_nilp`, `nilp_source_wt`, `nilp_target_wt`.

Package Extensions: `Transversal_W`

- The version of `pvars` in the package `Transversal_W` outputs the global variables `source_wt` and `target_wt` in addition to those mentioned above. Also, the nilpotent weights `nilp_source_wt` and `nilp_target_wt` are not used by `Transversal_W` (because it implements weighted filtrations) and these are not output.

Function: setup_Aclassn — set up global variables for $\mathcal{A}$ classification

Calling Sequence:

`setup_Aclassn`(n)

`setup_Aclassn`(n, p, l)

Parameters:

n, p	— positive integers
l	— a flag consisting of a list with two entries

Synopsis:

- This function provides a simple mechanism for assigning the global variables which define L. The actual settings specify a space L suitable for $\mathcal{A}$ classification problems (determinacy and complete transversals) and can of course be inspected with the `pvars` function.
- The parameter n specifies the source dimension to be used.
- In the first variant of the call, where the optional parameters p and l are omitted, the global variables are assigned to give $L = L\mathcal{A}_1$. The nilpotent variables `R_nilp`, `L_nilp`, `nilp_source_wt` and `nilp_target_wt` are not assigned.
- In the second variant of the call, where the optional parameters p and l are present, L is defined to be some nilpotent Lie algebra in $L\mathcal{A}$. Both parameters p and l are required. The parameter p specifies the target dimension and the parameter l is a flag which specifies which type of nilpotent Lie algebra is used. There are four natural types where $L \subset L\mathcal{A}$ contains as many of the 'extra' vectors as possible; see Section 3.2. The list l may take the values $[x1, 0]$, $[0, x1]$, $[xn, 0]$ or $[0, xn]$ and specifies which of these types as follows. (Note that l takes on values (symbols) which are Maple lists. Recall also that the source coordinates are defined to be $x1, \ldots, xn$ because `setup_Aclassn` assigns `liealg` := `stdjacobian`. These must be evaluated to Maple names using single right-quotes if they have already been assigned, for example 'x1'.)

$$L \;=\; L\mathcal{A}_1 \;\oplus\; \mathbf{F}\{x_i\partial/\partial x_j\} \;\oplus\; \mathbf{F}\{y_k\partial/\partial y_l\}$$

where the indices i, j, k and l satisfy one of the following conditions.

l	Indices	
$[x1, 0]$	$i > j$	$k < l$
$[0, x1]$	$i > j$	$k > l$
$[xn, 0]$	$i < j$	$k < l$
$[0, xn]$	$i < j$	$k > l$

The symbols which l may assume are purely notational; they were chosen because they indicate the *first* element in the resulting nilpotent ordering. Thus, the orderings start as

$$[x1^k, 0, \ldots, 0], \ldots; \;\; [0, \ldots, 0, x1^k], \ldots; \;\; [xn^k, 0, \ldots, 0], \ldots; \text{ or } [0, \ldots, 0, xn^k], \ldots;$$

respectively. The global variables `R_nilp` and `L_nilp` are assigned accordingly and the weights `nilp_source_wt` and `nilp_target_wt` are assigned as required in Proposition 3.1. For example, $l := [x1, 0]$ requires the Lie algebra

$$L \;=\; L\mathcal{A}_1 \;\oplus\; \mathbf{F}\{\, x_i\partial/\partial x_j : i > j \,\} \;\oplus\; \mathbf{F}\{\, y_k\partial/\partial y_l : k < l \,\}$$

and the nilpotent variables are defined as follows.

```
nilp := true_order;
R_nilp := [ [x2, 1], ..., [xn, 1],
            [x3, 2], ..., [xn, 2],
            ...,
            [xn, n − 1] ];
L_nilp := [ [1, 2],
            [1, 3], [2, 3],
            [1, 4], [2, 4], [3, 4],
            ...,
            [1, p], ..., [p − 1, p] ];
nilp_source_wt := [1, 2, ..., n];
nilp_target_wt := [0, −1, ..., −p + 1];
```

- In addition, the routine sets the variable `jetcalc_verbosity` to the recommended setting of 1; see `jetcalc`.

Function: setup_Agroup — set up global variables specifying the $\mathcal{A}$ group

Calling Sequence:

`setup_Agroup`(n)

Parameters:

n — a positive integer

Synopsis:

- This function provides a simple mechanism for assigning the global variables which define the space L. The actual settings define $L = L\mathcal{A}$, they can of course be inspected with the `pvars` function.
- The parameter n specifies the source dimension to be used.
- In addition, the routine sets the variable `jetcalc_verbosity` to the recommended setting of 1; see `jetcalc`.

Function: setup_Aunf — set up global variables for $\mathcal{A}$ unfolding

Calling Sequence:

`setup_Aunf`(n)

Parameters:

n — a positive integer

Synopsis:

- This function provides a simple mechanism for assigning the global variables which define the space L. The actual settings define $L = L\mathcal{A}_e$ and are suitable for $\mathcal{A}$ unfolding problems. They can of course be inspected with the `pvars` function.
- The parameter n specifies the source dimension to be used.
- In addition, the routine sets the variable `jetcalc_verbosity` to the recommended setting of 1; see `jetcalc`.

Chapter 5

A Tutorial

The following tutorial describes how some standard calculations in singularity theory may be carried out using the `Transversal` package. We will mainly concentrate on the case of $\mathcal{A}$-equivalence, this being of practical significance yet giving rise to calculations which are typically computationally demanding (that is, the sort of problems which motivated the development of `Transversal`). It should be an easy matter to apply `Transversal` to problems involving other equivalence relations once the material below has been understood.

In all of the following examples it is assumed that the user has already initiated a Maple session and loaded the required package, say `Transversal` or `Transversal_W`, by either issuing the appropriate command

```
> with(transversal);    or
> with(transversal_W);
```

(here `>` denotes the Maple prompt) or reading the source code files directly. (The file README which comes with the package gives installation instructions.)

5.1 Important Remarks

A certain amount of care most be exercised when using the package. There are several ways where miss-use of the package can easily produce incorrect results. *Care must be taken when defining the global setup variables and when redefining them to specify different calculations. If in doubt it is advisable to inspect the global variable settings before a given calculation is performed.* Similarly it is easy to miss-type equations defining jets or supply the wrong arguments in function calls, so it is worth double-checking. This is highlighted on several occasions throughout this tutorial. Although the package performs a certain amount of type-checking, in the end the onus is on the user to ask it to perform the correct calculations!

As an example consider the following scenario. `Transversal` will run under any Maple session, for example a standard Maple "text" session, but it is more convenient from an "ease of use" point of view to use a session which provides a graphical interface, for example “xmaple”. Under Maple V Release 4, xmaple allows the user to open several “servers” (separate Maple windows) within the session and an easy mistake to make is to modify the global setup variables within a new server window without realising these global variables apply to the whole session, that is to *every* Maple window within the session. For example, it would be a mistake to define two windows within the same session with one used for “unfolding” calculations and the other used for “classification” calculations: one would need to remember to redefine the global variables every time one changed windows. A far less error-prone solution would be to use entirely different Maple sessions for each type of calculation.

As to the correctness of the actual code: the packages have been checked very extensively, via numerous testing of internal routines to verify that they do what they should, and also by reproducing numerous calculations from classifications within the literature. However, one can never be one hundred percent certain that the code is correct so please report any questionable results you may obtain. In addition, the organisation of the code (for example, the separation of routines into directories, as opposed to the actual algorithms) has undergone some major changes in the latest version to make it more suitable for public release. It is quite possible that minor errors have crept in (such as directory locations of code not being updated). I spent a lot of time checking this release and think I have covered such problems, but if not these errors should manifest themselves quickly and they should be obvious. Please report any problems.

Performance times. Regarding the performance of the package we will be deliberately vague, after all a typical user is interested in whether their problem will complete in an acceptable time rather than comparing times for different problems on different machines. As an indication of expected times for the calculations described in this tutorial we remark that most should complete within a few seconds, or possibly minutes in the more demanding cases, on ‘reasonably modern’ computers. For example, such times apply on a Sparc Ultra workstation (168MHz) and a Pentium PC (133MHz) (which some would regard as ‘dated’, even at time of writing!). Of course, the complexity of the problem depends on the source and target dimensions (n and p) and the jet-space degree (k) in an ‘exponential’ way and as these increase one should expect calculation times nearing hours, or possibly even failure due to the size of the problem.

Typically, the calculations we have considered have not caused serious problems. If there is a concern about performance time for a certain calculation and/or whether it will actually complete we suggest setting `jetcalc_verbosity` to the value 2 or 3 to obtain more diagnostics from `jetcalc`. This will cause `jetcalc` to output the dimensions of the matrices it is reducing, also which row it is currently working on. These dimensions give an *indication* of the size of the problem, mainly through comparison with similar calculations which have already succeeded (for example with the same n and p but a lower value for k). That is, the matrix dimensions are given merely as a relative guide to calculation time. Do not be too concerned about the sizes of some of these matrices, they are exceptionally sparse and the elimination techniques can often reduce seemingly large matrices efficiently, as can be seen by observing the dimensions in some of the problems described in this tutorial.

The largest obstruction to calculations appears to come from the presence of moduli. Examples suggest that calculations for families with 3 or more moduli become infeasible in jet-spaces of degree in the region of 10 to 20 (depending on n and p and still being deliberately vague here!). This is an inherent problem caused by the creation of symbolic expressions during the elimination process which rapidly become large, often too large for Maple to handle.

Some of the above points are discussed further in the article [17] which describes the `Transversal` package.

5.2 $\mathcal{A}$-Classification of Map-Germs

A comprehensive classification of map-germs $\mathbf{R}^2, 0 \to \mathbf{R}^4, 0$ under $\mathcal{A}$-equivalence was carried out by Bruce, Kirk and West and described in [6, 16]. We will consider several branches of this classification and perform the calculations using `Transversal`.

5.2.1 Complete Transversals and Determinacy

In this example we consider the $J^3\mathcal{A}$-orbits over the 2-jet $(x, y^2, 0, 0)$. Let (x, y) denote coordinates in the source and (u_1, u_2, u_3, u_4) those in the target. Let $\mathcal{G}$ be the unipotent subgroup of $\mathcal{A}$ having nilpotent Lie algebra

$$L = L\mathcal{A}_1 \ \oplus\ \mathbf{R}\{x\partial/\partial y\} \ \oplus\ \mathbf{R}\{u_i\partial/\partial u_j \quad \text{for} \quad i > j\}.$$

This group will be used in all of the complete transversal (CT) and determinacy calculations. Consider the jet-spaces $J^{r,s}(2,4)$ induced by the nilpotent filtration. The monomial vectors of (standard) degree r are partitioned into their (r, s)-levels as described in Section 3.2 using the weights $\alpha = (2, 1)$ and $\beta = (-3, -2, -1, 0)$. The following command sets up the global variables defining the nilpotent Lie algebra L and Maple responds by printing out the values.

```
> setup_Aclassn(2,4,[0,x2]);
                              liealg = stdjacobian
                                   equiv = A
                               compltrans = true
                                source_dim = 2
                               source_power = 2
                               target_power = 2
                              nilp = true_order
                                    R_nilp:
```

```
                            [[x1, 2]]
                             L_nilp:
          [[2, 1], [3, 1], [4, 1], [3, 2], [4, 2], [4, 3]]
                         nilp_source_wt:
                              [2,1]
                         nilp_target_wt:
                           [-3,-2,-1,0]
                       jetcalc_verbosity = 1
```

The function setup_Aclassn is described in detail in Section 4.4. Basically it defines the space $L\mathcal{A}_1$ and, if the optional arguments specifying the target dimension and the 'Lie algebra flag' (4 and `[0,x2]` above) are present, assigns the global 'nilpotent' variables `nilp`, `R_nilp`, `L_nilp`, `nilp_source_wt` and `nilp_target_wt` accordingly.

We now specify the germ $f = (x, y^2, 0, 0)$ and calculate the orbit in the 3-jet-space $J^3(2,4)$. Recall that since `liealg` = `stdjacobian` and `source_dim` = 2, the source coordinates x, y are denoted by `x1`, `x2` for the purposes of `Transversal`.

```
> f := [x1,x2^2,0,0];

                                      2
                       f := [x1, x2 , 0, 0]

> jetcalc(f,3);

                           defined map:
                                 2
                         [x1, x2 , 0, 0]

             working in 3-jet space with A-equivalence

                       defined coordinates:
                            [x1, x2]

           using ordering induced by the nilpotent weights

                              Ready.
```

Remark. In the remainder of this section we will always provide the commands one should enter but, for brevity, we will not always show the corresponding Maple response (for example, the response to an assignment such as `f` above may be omitted). Likewise, the output from `jetcalc` shown above will be omitted from now on also. This may be achieved in your actual session by setting the global variable `jetcalc_verbosity` to the value 0 (which causes `jetcalc` to operate in silence). However ...

Important. In practice one is strongly advised to set `jetcalc_verbosity` to the value 1 (or a higher value for more diagnostics; see Chapter 4) and check that `jetcalc` is working with the intended settings (map-germ, jet-space degree, equivalence, and so forth). For example, in a session which uses a graphics interface it is possible to simply "scroll back" and redefine the jet `f` and then re-call `jetcalc`, thus saving on typing, but it is also easy to introduce typos using this technique. Similarly, if several different jets have been defined it is an easy mistake to pass the wrong one to `jetcalc`. Thus: it is advisable to always check the messages displayed by `jetcalc` otherwise such errors may go undetected. Also note that the warning displayed by `jetcalc` when the "checklist" is non-empty is suppressed if `jetcalc_verbosity` is set to 0.

The above command calculates a basis for $L \cdot f$ and its complementary space in $J^3(2, 4)$ and stores all of the results — these may be viewed using the various 'print' routines. For example, to display a complete transversal we type

```
> pcomp();

                                 3
                     [0, 0, 0, x2 ]
                              3
                     [0, 0, x2 , 0]
                               2
                  [0, 0, 0, x1  x2]
                            2
                  [0, 0, x1  x2, 0]
```

Under normal circumstances `pcomp` displays a basis for the complementary space but in the present scenario, where the variable `compltrans` is set to the value `true`, `pcomp` will just display the basis elements of degree 3 (these basis elements are necessarily homogeneous). Since `jetcalc` orders monomial jets as dictated by the nilpotent filtration induced by L, a complete transversal can simply be "selected" from the basis for the complementary space by just considering those elements of degree 3. More precisely, considering $f = (x, y^2, 0, 0)$ as a $(3, 0)$-jet, a $(3, 1)$-transversal is (spanned by) $\{(0, 0, 0, y^3)\}$, while considering f as a $(3, 1)$-jet, a $(3, 2)$-transversal is $\{(0, 0, y^3, 0)\}$. Similarly, a $(3, 3)$-transversal is $\{(0, 0, 0, x^2y)\}$, and a $(3, 4)$-transversal is $\{(0, 0, x^2y, 0)\}$. Note that the partition of monomial vectors into their various $(3, s)$-levels can be displayed using the command

```
> pmons(3);
```

Note: this would *not* have worked before calling `jetcalc` unless the user had specified the source coordinates by assigning the global variable `coords` to be the list `[x1,x2]` (this assignment was done above by the `liealg` routine `stdjacobian` which is called from within `jetcalc`).

Although all of the degree 3 terms in the complete transversal are output, we should consider each separately at its own $(3,s)$-level. It turned out to be more convenient, both from a user's point of view and a programming point of view, to output the whole set of homogeneous terms and let the user select the appropriate (r,s)-level, with the help of the function `pcomp` if necessary.

Thus, at the $(3,1)$-level we obtain the orbits $(x, y^2, 0, ay^3)$ for $a \in \mathbf{R}$ and, after scaling, these reduce to $(x, y^2, 0, y^3)$ and $(x, y^2, 0, 0)$. If we continue with the first, classifying the higher $(3,s)$-orbits, we obtain the normal forms (x, y^2, x^2y, y^3) and $(x, y^2, 0, y^3)$. These are equivalent to normal forms obtained by classifying the higher $(3,s)$-orbits over the other $(3,1)$-jet: $(x, y^2, 0, 0)$. We shall therefore concentrate on this case and leave it as an exercise to the reader to repeat the process for $(x, y^2, 0, y^3)$. Note that one occasionally finds that such redundancies occur, even when using nilpotent filtrations (which are, of course, much finer than the standard filtration by degree)

Consider the $(3,1)$-jet $(x, y^2, 0, 0)$. From the earlier calculation, the $(3,2)$-transversal gives, after scaling, the orbits $(x, y^2, y^3, 0)$ and $(x, y^2, 0, 0)$. Consider the first; we calculate the higher $(3,s)$-transversals by calling `jetcalc` again and specifying the same degree, 3.

```
> f := [x1,x2^2,x2^3,0];
> jetcalc(f,3);
```

Once `jetcalc` has finished we display the complete transversal using the function `pcomp`. Exactly the same vectors as before are output, but now we only consider the homogeneous $(3,3)$ terms. Thus, since $(0,0,0,x^2y)$ is the only one this gives, after scaling, the $(3,3)$-orbits (x, y^2, y^3, x^2y) and $(x, y^2, y^3, 0)$. Continuing with the first ...

```
> f := [x1,x2^2,x2^3,x1^2*x2];
> jetcalc(f,3);
> pcomp();
```

```
                     3
          [0, 0, 0, x2 ]
                 3
          [0, 0, x2 , 0]
                   2
         [0, 0, 0, x1  x2]
```

Remember that we are now only considering the monomial jets at the $(3,4)$-level (and higher). This indicates that all the higher $(3,s)$-transversals for the $(3,3)$-jet (x, y^2, y^3, x^2y) are empty and this therefore provides a representative for a $J^3\mathcal{A}$-orbit. Returning to the other $(3,3)$-jet $(x, y^2, y^3, 0)$, from the previous `jetcalc` calculation using this jet we see that the only higher non-empty $(3,s)$-transversal is the $(3,4)$-transversal, $\{(0,0,x^2y,0)\}$. We obtain the $(3,4)$-orbits $(x, y^2, y^3 \pm x^2y, 0)$ and $(x, y^2, y^3, 0)$. We already know the higher $(3,s)$-transversals are empty in the latter case, while in the former this is easily checked by a further call `jetcalc(f,3)` with $f = (x, y^2, y^3 \pm x^2y, 0)$. We therefore obtain two more $J^3\mathcal{A}$-orbits and the complete list over the $(3, 2)$-jet $(x, y^2, y^3, 0)$ is therefore

$$(x, y^2, y^3, x^2y), \quad (x, y^2, y^3 \pm x^2y, 0), \quad (x, y^2, y^3, 0).$$

We now return to the $(3, 2)$-jet $(x, y^2, 0, 0)$. From the original calculation for $(x, y^2, 0, 0)$ we see that a $(3, 3)$-transversal is $\{(0, 0, 0, x^2y)\}$. For $f = (x, y^2, 0, x^2y)$ the higher $(3, s)$-transversals are empty. However, this $J^3\mathcal{A}$-orbit is redundant, for consider the other $(3, 3)$-jet, $(x, y^2, 0, 0)$. Again, from the original calculation, the only higher transversal is the $(3, 4)$-transversal, $\{(0, 0, x^2y, 0)\}$. For both of the resulting $(3, 4)$-orbits the higher transversals are empty. Hence, the complete list of $J^3\mathcal{A}$-orbits over $(x, y^2, 0, 0)$ is as follows. The numbers alongside each orbit indicate its corresponding $J^3\mathcal{A}$-codimension. It is often useful to calculate these invariants during the classification process to determine, among other things, whether the orbits are all distinct. The codimension can be easily calculated using `Transversal`, this is discussed in Section 5.2.3 below.

(x, y^2, y^3, x^2y)	5
$(x, y^2, y^3 \pm x^2y, 0)$	6
$(x, y^2, y^3, 0)$	7
$(x, y^2, x^2y, 0)$	7
$(x, y^2, 0, 0)$	9

Remark. The above calculation is typical of the lower degree levels of a classification where nilpotent methods come into there own. Although the process is quite tedious it is worth pointing out that it provides a very quick and algorithmic means of classifying the jets lying over a given jet. In addition, at the higher levels of a classification the CTs are typically either empty or contain just one term and classification of jets over the given jet is immediate; see the discussion below on determinacy for an example of this. To further stress the usefulness of nilpotent CT methods consider repeating the above calculation using $\mathcal{A}_1$ methods instead. That is set $L = L\mathcal{A}_1$, which may be achieved using the command

```
> setup_Aclassn(2);
```

and then call `jetcalc` as above with $f = (x, y^2, 0, 0)$ (only one call is needed now) and then `pcomp`. A 3-$\mathcal{A}_1$-transversal is

$$(x, y^2, a_1y^3 + a_2x^2y, a_3y^3 + a_4x^2y) \quad \text{for} \quad a_i \in \mathbf{R}.$$

Of course, classifying this family is preferable to a direct classification of the affine space of all 3-jets over $(x, y^2, 0, 0)$, but it would still involve a lot of (ad-hoc) work to reduce to the five cases above. An even more marked example is given by the exercise at end of this section.

Note: if you defined the global setup variables to specify $L = L\mathcal{A}_1$ as described in the above remark then do not forget to specify the original nilpotent space by re-issuing the command

```
> setup_Aclassn(2,4,[0,x2]);
```

We will demonstrate determinacy calculations by considering the first and second cases above. Note that if a germ f has 2-jet $(x, y^2, 0, 0)$ we can appeal to the fact that $\mathcal{M}_2^{k+1}.f^*(\mathcal{M}_4).\mathcal{E}_2 \supset \mathcal{M}_2^{k+3}$ and by Theorem 2.5 can work in the jet-space $J^{k+2}(2,4)$ to prove k-determinacy.

We firstly show that (x, y^2, y^3, x^2y) is 3-determined. Even if we do not suspect this we still have to calculate the higher transversals in order to extend the classification so may as well check determinacy in the process using the `classify` procedure. This procedure calculates (via `jetcalc`) a succession of transversals, stopping if a non-empty transversal is produced or a given jet-level is reached. The appropriate command specifies the jet f and the CT degree limits. The Maple output shown below is produced when `jetcalc_verbosity` is set to 0. However, recall that the recommended setting is 1 and in this case messages from `jetcalc` will appear before each message reporting an empty transversal.

```
> f := [x1,x2^2,x2^3,x1^2*x2];
> classify(f,4,5);

                          the 4 transversal is empty

                             --------------------

                          the 5 transversal is empty

                             --------------------

                                      2     3     2
                         germ, [x1, x2 , x2 , x1  x2]

                             degree limits, 4, 5

                         all transversals were empty
```

It follows that

$$\mathcal{M}_2^4\mathcal{E}(2,4) \subset L \cdot f + \mathcal{M}_2^6\mathcal{E}(2,4)$$

so by Theorem 2.5 we can conclude that f is 3-$\mathcal{A}$-determined. Of course, this conclusion can also be achieved by making separate calls to `jetcalc` and verifying that each CT is empty using `pcomp`. The routine `classify` simply provides a convenient means of doing this, especially when several levels of CTs need checking.

Note: in cases where f contains external parameters (such as moduli) one should make separate calls to `jetcalc` in order to determine the degenerate behaviour at each jet-level. This is done by examining the "checklist", substituting in the degenerate values of the moduli, and then recalling `jetcalc` with the corresponding jet to see if the CT remains empty (ie. does *not* in fact degenerate) or gives rise to a new branch in the classification tree. An example of this is given in Section 5.2.2; see also Sections 3.2 (the part on symbolic pivots) and 4.4 (the entries for `jetcalc` and `plist`).

As an alternative we could check the above determinacy condition directly by using the procedure `determined`. However, the inductive approach offered by `classify`, whereby the next non-empty transversal provides the next level of the classification tree when the given germ is not finitely determined, is generally more practical and is recommended for most applications. The routine `determined` takes the jet f as an argument, together with the determinacy degree to be tested and the degree of the jet-space in which one may perform the calculation. The following messages outlining the state of the calculation are output along with the usual `jetcalc` messages.

```
> determined(f,3,5);

                 *** calculating tangent space ***

                       --------------------

               *** checking determinacy condition ***

                    number of vectors to check, 44

                       --------------------

                                 2    3    2
                   germ, [x1, x2 , x2 , x1  x2]

                      determinacy degree, 3

                        jetspace degree, 5

                   determinacy criterion holds
```

We now consider the 3-jet $(x, y^2, y^3 \pm x^2y, 0)$. Using `jetcalc` we calculate the 4-transversal in the usual manner. The cases $(x, y^2, y^3+x^2y, 0)$ and $(x, y^2, y^3-x^2y, 0)$ must, of course, be considered separately; they produce identical results so we only discuss the first.

```
> f := [x1,x2^2,x2^3+x1^2*x2,0];
> jetcalc(f,4);
> pcomp();

                                      3
                       [0, 0, 0, x1  x2]
```

It was noted above that `pcomp` outputs the whole set of (in this case) degree 4 homogeneous terms in the basis, leaving the user to identify the appropriate terms from the $(4, s)$-level. This example demonstrates why this is desirable. Here we see that all $(4, s)$-transversals are empty except the $(4, 4)$-transversal, $\{(0, 0, 0, x^3y)\}$ and we can immediately conclude that $(x, y^2, y^3 + x^2y, x^3y)$ and $(x, y^2, y^3 + x^2y, 0)$ are the $J^4\mathcal{A}$-orbits lying over the 3-jet $(x, y^2, y^3 + x^2y, 0)$. We could have used `classify` to obtain this (`classify(f,4,5)`), though it is clear that all the 4-transversals could not have been empty. `classify` is generally used when we suspect the germ may be determined and need to check several levels of CTs to verify this. It is appropriate to use `classify` to investigate $(x, y^2, y^3 + x^2y, x^3y)$. Below we show this germ is 4-determined and then proceed with the second case $(x, y^2, y^3 + x^2y, 0)$. Note that we have used two separate variables `g` and `f` to denote these germs (this would be clear if the message output by `jetcalc` was shown).

```
> g := [x1,x2^2,x2^3+x1^2*x2,x1^3*x2];
> classify(g,5,6);

                  the 5 transversal is empty

                     --------------------

                  the 6 transversal is empty

                     --------------------

                        2     3     2       3
             germ, [x1, x2 , x2  + x1  x2, x1  x2]

                     degree limits, 5, 6

                 all transversals were empty

> jetcalc(f,5);
> pcomp();

                                      4
                       [0, 0, 0, x1  x2]
```

Further calculation shows that $(x, y^2, y^3 + x^2y, x^4y)$ is 5-determined and proceeding further gives the first few germs in the series $(x, y^2, y^3 + x^2y, x^ky)$. To prove that this series exists we have to resort to hand calculations but such calculations are straightforward once the actual result is known (as is so often the case!). In practice obtaining the first few terms of a series (and conjecturing on the general form) is often the difficult part.

Exercise. A comprehensive classification of map-germs $\mathbf{R}^3, 0 \to \mathbf{R}^4, 0$ under $\mathcal{A}$-equivalence was carried out by Houston and Kirk and described in [15]. A similar example to the calculations carried out above is given by considering the classification of 3-jets over the 2-jet (x, y, yz, xz) (this is one of the five corank-1 2-jets in the classification). Here (x, y, z) denote coordinates in the source and (u_1, u_2, u_3, u_4) will denote those in the target. Let $\mathcal{G}$ be the unipotent subgroup of $\mathcal{A}$ having nilpotent Lie algebra

$$L\mathcal{A}_1 \;\oplus\; \mathbf{R}\{x\partial/\partial y, x\partial/\partial z, y\partial/\partial z\} \;\oplus\; \mathbf{R}\{u_i\partial/\partial u_j \quad \text{for} \quad i > j\}.$$

This can be setup in `Transversal` by issuing the command

```
> setup_Aclassn(3,4,[0,x3]);
```

We leave it as an exercise to show that the complete list of 3-jets is

$$(x, y, yz, xz + z^3), \quad (x, y, yz, xz),$$

having $J^3\mathcal{A}$-codimension 4 and 6, respectively. As an instructive example we note that if the same calculation was repeated using, instead, the space $L = L\mathcal{A}_1$ that a 3-CT over (x, y, yz, xz) is

$$(x, y, yz + a_1z^3 + a_2xz^2, xz + a_3z^3 + a_4xz^2 + a_5yz^2) \quad \text{for} \quad a_i \in \mathbf{R}.$$

Of course, one may reduce this to the two cases above, but this would involve a lot of (ad-hoc) work. To classify the 3-jets over (x, y, yz, xz) without the use of techniques such as CTs would be a very unenviable task!

Continuing the classification over the first of the above 3-jets gives the series $(x, y, yz + z^k, xz + z^3)$, k-determined for $k \geq 4$, k not a multiple of 3. (Note that this is not the complete classification of all jets over this 3-jet. Further branching occurs at the 6-level and 7-level. This is an important example in theoretical singularity theory in that the 3-jet gives rise to a series but is not stem.) As an exercise we suggest repeating the determinacy calculation for the first member of this series. (Hint: using Theorem 2.5 and noting that $f^*(\mathcal{M}_3).\mathcal{E}(3,4) \supset \mathcal{M}_3^3.\mathcal{E}(3,4)$, we may establish 4-determinacy for $k = 4$ by showing that the CTs from degree 5 to degree 7 are empty.)

5.2.2 Working with the $\mathcal{A}$-group: Mather's Lemma and Moduli

We will continue with the classification introduced in the previous section but will consider a higher branch in the classification tree. The examples were chosen to demonstrate the use of `Transversal` in the following areas.

- prove that a parameter in a family of jets is a modulus.
- identify the degenerate values of moduli (we observe where the classification process breaks down by inspecting the "checklist" calculated by `jetcalc`).
- apply Mather's lemma to a family of jets, thus reducing the family to a finite number of orbits (inspection of the "checklist" identifies any degenerate members which form an obstruction to triviality along the whole family).

To start with consider the family of 7-jets $f = (x, y^2, xy^3 + x^4y, y^5 + ax^6y)$.

Exercise. Using the methods of the previous section show that this is the family of 7-jets having 6-jet $(x, y^2, xy^3 + x^4y, y^5)$.

We attempt to 'scale' a to a unit using simple 'scaling' coordinate changes in the source and target. This is an elementary problem in linear algebra, but the resulting system of linear equations is overdetermined and we suspect a is a modulus. To actually prove a is a modulus we appeal to Lemma 2.8 which, in this case, amounts to showing that $(0, 0, 0, x^6y)$ is contained in $L(J^7\mathcal{A}) \cdot f$ only for isolated points a (if at all).

Firstly we define the global setup variables to specify the space $L = L\mathcal{A}$. This may be done manually, however, for the case of $\mathcal{A}$-equivalence the routine `setup_Agroup` is provided for convenience. This routine is described in Section 4.4, but it should be clear why the following settings define the Lie algebra of the $\mathcal{A}$ group. Note that the nilpotent variables remain undefined.

```
> setup_Agroup(2);
                        liealg = stdjacobian
                              equiv = A
                         compltrans = false
                           source_dim = 2
                          source_power = 1
                          target_power = 1
                            nilp = false
                              R_nilp:
                               R_nilp
                              L_nilp:
                               L_nilp
                          nilp_source_wt:
```

```
                    nilp_source_wt
                    nilp_target_wt:
                    nilp_target_wt
                  jetcalc_verbosity = 1
```

Warning: (to further stress an earlier remark) if the above is done during a classification session it is easy to return to the CT calculations while *forgetting* to reset the global variables to define the corresponding nilpotent space L. It is advisable to carry out the new calculations, where $L = L\mathcal{A}$, by initiating a second, different Maple session.

Next we define f and use `jetcalc` to calculate the tangent space $L(J^7\mathcal{A}) \cdot f$. The condition which shows a is a modulus is then checked using the function `intangent`.

```
> f := [x1,x2^2,x1*x2^3+x1^4*x2,x2^5+a*x1^6*x2];
> jetcalc(f,7);
> intangent([0,0,0,x1^6*x2]);
                                      false
```

The function `intangent` returns `false` when the set of vectors passed to it as arguments forms an independent set to the tangent space calculated to `jetcalc`, and `true` otherwise. In the case where only one argument is passed (as here) it therefore checks whether the given vector is "in the tangent space" or not. The response `false` indicates this is not the case and therefore that a is a modulus. This holds for all generic values of a except the exceptional cases indicated by the "checklist" returned by `jetcalc` (more on this next). The checklist is in fact empty in this case.

We now continue with the classification and return to CT calculations, in particular CT and determinacy calculations for families. Remember to define L to be the nilpotent space specified in Section 5.2.1 (using the routine `setup_Aclassn`) or move to a separate Maple session reserved for this purpose, as recommended above. We could use `classify` to show that f is 7-determined, but this would only apply to generic a. For families it is more convenient to check determinacy by calculating each transversal separately using `jetcalc`, identifying any exceptional values and, if required, determining how the classification branches for these exceptional values.

```
> f := [x1,x2^2,x1*x2^3+x1^4*x2,x2^5+a*x1^6*x2];
> jetcalc(f,8);
```

Observe that when `jetcalc` has finished the calculation it responds with the warning:

```
        WARNING: global variable 'checklist' is non-empty !!!
```

The Gaussian elimination routines in `jetcalc` try to choose the best pivotal elements. They give preference to numeric (non-symbolic) pivots but this is not always possible and the global variable `checklist` is used to store the non-numeric pivots. To display these we use the command `plist`.

```
> plist();

                              #1, 1 + a
                              #2, 1 + a
                              #3, 1 + a
                              #4, 1 + a
```

The first column indicates the index number of the pivotal element as an entry in the table `checklist` and the second column the actual pivotal element (this is the important bit). By default `plist` factorises the pivots before it displays them as an aid to determining the exceptional values (that is, where the pivot vanishes) though in this case it is already clear! If need be we can access each pivotal element directly, for example the pivot numbered $\#i$ can be obtained using the expression `checklist[i][1]`. Next we display the complete transversal.

```
> pcomp();
                *** THE NORMAL SPACE IS EMPTY ***
```

Thus, the 8-transversal is empty, but only provided $1 + a \neq 0$. To investigate the exceptional value we must re-run `jetcalc` using the corresponding jet ...

```
> h := subs(a=-1,f);
> jetcalc(h,8);
> pcomp();

                                    7
                        [0, 0, 0, x1  x2]
```

The case $a = -1$ therefore produces a separate branch to the classification tree. We will return to this shortly but will firstly continue with the determinacy calculation in the generic case.

```
> jetcalc(f,9);
      WARNING: global variable 'checklist' is non-empty !!!
```

Again, we must use `plist()` to display the non-numeric pivots. We find the conditions on a are the same as before. Using `pcomp()` shows that the 9-transversal is empty and it follows that, provided $a \neq -1$,

$$\mathcal{M}_2^8\mathcal{E}(2,4) \subset L \cdot f + \mathcal{M}_2^{10}\mathcal{E}(2,4)$$

so by Theorem 2.5 we can conclude that f is 7-determined.

Note: sometimes `plist` returns conditions on parameters such as a which turn out to be redundant, that is, re-running `jetcalc` with the 'exceptional' values substituted in causes no change to the complete transversal. These conditions only appeared in the first place because `jetcalc` had no other choice for the corresponding pivotal elements.

Important Remark. The Gaussian elimination routines within `jetcalc` use the Maple function `normal` when reducing each row. This is a basic form of simplification (chosen for efficiency purposes) which recognises those expressions equal to zero which lie in the domain of "rational functions". For expressions containing subexpressions such as square roots, powers, and functions, `normal` may not recognise when an expression is equal to zero. Thus, `jetcalc` should not be called using exceptional values such as square roots and other (non-integer) powers. This restriction could be overcome by re-writing the elimination routines to use the Maple function `simplify`. However, when the entries in the checklist become complicated (this is especially the case when more than one modulus is present) investigation of the exceptional values is generally quite limited.

We now consider the exceptional case in the above example, namely the 7-jet $h = (x, y^2, xy^3 + x^4y, y^5 - x^6y)$. From above, an 8-transversal is $\{(0, 0, 0, x^7y)\}$ giving the family of 8-jets

$$f = (x, y^2, xy^3 + x^4y, y^5 - x^6y + ax^7y).$$

A simple, though somewhat tedious, exercise in linear algebra shows that this can be reduced to the normal forms $(x, y^2, xy^3 + x^4y, y^5 - x^6y \pm x^7y)$ and $(x, y^2, xy^3 + x^4y, y^5 - x^6y)$ via 'scaling' coordinate changes in the source and target. Alternatively, this can also be deduced from Mather's Lemma (Lemma 2.7) and we take this opportunity to demonstrate the method using `Transversal`. There are situations where the use of Mather's Lemma is more important, for instance, where 'scaling' will not work and applying the Mather Lemma shows triviality for the whole family.

Remember to begin by setting the global variables to specify $L = L\mathcal{A}$ as described above, or using a separate Maple session which was reserved for such calculations. Now calculate the tangent space $L(J^8\mathcal{A}) \cdot f$, determine whether the vector $(0, 0, 0, x^7y)$ belongs to this space, and display the dimension or codimension of the space.

```
> f := [h[1],h[2],h[3],h[4]+a*x1^7*x2];
> jetcalc(f,8);
        WARNING: global variable 'checklist' is non-empty !!!
> plist();

                              #1, 2/3 a
```

```
> intangent([0,0,0,x1^7*x2]);
   WARNING: original matrix contains non-numeric elements, check
                           checklist !!!
                              true
> basis_dim;
                              162
```

This tells us that $L(J^8\mathcal{A})\cdot f$ is of constant dimension 162 and contains $(0,0,0,x^7y)$ provided $a \neq 0$. The warning output from `intangent` also reminds us that non-numeric pivotal elements exist. On some occasions non-numeric terms are introduced by the elimination routine in `intangent`. When this happens `intangent` outputs a vector containing the offending term. (This vector is a coordinate vector of coefficients but the specific interpretation does not matter. The important point is that the non-numeric terms should be treated in the same manner as those given by `plist` — the values for which they vanish should be investigated.) Substituting $a = 0$ into f and repeating the above procedure shows $L(J^8\mathcal{A})\cdot f$ is of dimension 161 and does not contain $(0,0,0,x^7y)$. Thus, by Mather's Lemma, the family can be reduced to the orbits

$$(x, y^2, xy^3 + x^4y, y^5 - x^6y \pm x^7y), \quad (x, y^2, xy^3 + x^4y, y^5 - x^6y).$$

Note that in the complex case the two orbits

$$(x, y^2, xy^3 + x^4y, y^5 - x^6y + x^7y), \quad (x, y^2, xy^3 + x^4y, y^5 - x^6y - x^7y)$$

reduce to one, but in the real case this simplification may or may not occur (further analysis is required). Also note that the above can be concluded by examining the global variable `codim`, which gives the codimension, instead of `basis_dim` (`codim` takes the value 18 for $a \neq 0$, but jumps to 19 for $a = 0$).

Exercise. Continue the classification to obtain the first few terms of the series $(x, y^2, xy^3 + x^4y, y^5 - x^6y \pm x^ky)$, which is $(k+1)$-determined for $k \geq 7$.

Remark (Families of Higher Modality). `Transversal` can be used to detect families of higher modality. For example, consider the family

$$g = (x, y^2, x^3y \pm xy^5 + by^7, x^2y^3 + ay^7)$$

Make sure the global variables define $L = L\mathcal{A}$ and calculate $L(J^7\mathcal{A})\cdot g$.

```
> g := [x1,x2^2,x1^3*x2+x1*x2^5+b*x2^7,x1^2*x2^3+a*x2^7];
> jetcalc(g,7);
> intangent([0,0,x2^7,0],[0,0,0,x2^7]);
                              false
```

The response `false` indicates $\{(0,0,y^7,0),(0,0,0,y^7)\}$ forms an independent set to $L(J^7\mathcal{A})\cdot g$ in $J^7(2,4)$ (for all a and b) so by Lemma 2.8 g defines a bimodular family of 7-jets.

5.2.3 Unfoldings and Codimension

In this final section we will demonstrate how to use `Transversal` to calculate versal unfoldings for finitely-determined map-germs, thus completing the armoury of techniques typically required when performing $\mathcal{A}$-classifications. After the rather technical 'goings-on' of the previous section the examples given below should be relatively straight forward.

We will begin with a simple example demonstrating how one calculates the $J^3\mathcal{A}$-codimension of the 3-jets considered in Section 5.2.1. When the jet is 3-determined these numbers equal the $\mathcal{A}$-codimension of the corresponding germ. However, regardless of this, it is still useful to calculate them as they provide useful invariants at the jet-level. Firstly we define the global setup variables to specify the space $L = L\mathcal{A}$. This may be done using the routine `setup_Agroup` as described in Section 5.2.2 above. Next we define the 3-jet $f = (x, y^2, y^3, x^2y)$ and call `jetcalc` in the usual way. The codimension is stored as the variable `codim`, while a basis for the complementary space to $L(J^3\mathcal{A})\cdot f$ in $J^3(2,4)$ may be displayed using the function `pcomp`. Note that the whole set of basis elements is output since `compltrans` is set to `false`.

```
> f := [x1,x2^2,x2^3,x1^2*x2];
> jetcalc(f,3);
> codim;
                                  9

> pcomp();

                           [1, 0, 0, 0]
                           [0, 1, 0, 0]
                           [0, 0, 1, 0]
                           [0, 0, 0, 1]
                          [0, x2, 0, 0]
                          [0, 0, x2, 0]
                          [0, 0, 0, x2]
                         [0, 0, x1 x2, 0]
                         [0, 0, 0, x1 x2]
```

Important Remark. Observe that `jetcalc` calculates a basis for the complementary space in $J^3(2,4)$, but the $J^3\mathcal{A}$-codimension is defined by working in the space of jets which vanish at 0. We therefore discard the constant jets from the above result and conclude that the $J^3\mathcal{A}$-codimension of f is 5. One must always be aware of this caveat when working with, say, $\mathcal{A}$-codimension calculations. The use of the function `pcomp`, along with simply displaying `codim`, is recommended.

Exercise. Calculate the $J^3\mathcal{A}$-codimension of the remaining 3-jets lying over the 2-jet $(x, y^2, 0, 0)$. The results were given in Section 5.2.1.

Generally, if a germ $f : \mathbf{R}^n, 0 \to \mathbf{R}^p, 0$ is k-$\mathcal{A}$-determined then, by the determinacy theorems of [7], there exists some unipotent subgroup $\mathcal{G}$ of $\mathcal{A}$ such that

$$\mathcal{M}_n^{k+1}\mathcal{E}(n,p) \subset L\mathcal{G} \cdot f$$

and, in particular,

$$\mathcal{M}_n^{k+1}\mathcal{E}(n,p) \subset L\mathcal{A} \cdot f$$

so that the $\mathcal{A}$-codimension of f is equal to its $J^k\mathcal{A}$-codimension. Similarly, we can calculate an $\mathcal{A}$-versal unfolding by working in the k-jet-space. Thus, since $f = (x, y^2, y^3, x^2y)$ is 3-determined we can conclude that it is of $\mathcal{A}$-codimension 5, having $\mathcal{A}$-versal unfolding

$$\begin{aligned}(x, y, u_1, u_2, u_3, u_4, u_5) \quad \mapsto \quad & (x, y^2 + u_1 y, y^3 + u_2 y + u_3 xy, x^2 y + u_4 y + u_5 xy, \\ & u_1, u_2, u_3, u_4, u_5).\end{aligned}$$

Exercise. Consider the family of 8-jets $(x, y^2, xy^3 + x^4y, y^5 - x^6y + ax^7y)$. Show that each member of this family is of $J^8\mathcal{A}$-codimension 18 for $a \neq 0$, but of $J^8\mathcal{A}$-codimension 19 when $a = 0$. This is an exercise in using the `checklist` and is almost a repeat of the calculation performed in Section 5.2.2 which used Mather's Lemma to reduce this family to a finite number of orbits. Since the family is 8-determined for $a \neq 0$ this shows the $\mathcal{A}$-codimension is 18 in this case.

To finish we will discuss the calculation of $\mathcal{A}_e$-codimension and unfoldings. This should now be a straightforward exercise, it only really entails redefining the global variables to specify the space $L = L\mathcal{A}_e$, and may be done via the routine `setup_Aunf`, described in Section 4.4, as follows.

```
> setup_Aunf(2);
```

Alternatively, with the global variables already set to define $L = L\mathcal{A}$, we could achieve the settings for $L = L\mathcal{A}_e$ by putting the powers of the maximal ideals equal to 0. Whenever one assigns the global setup variables "by-hand" it is extremely advisable to double check these settings by printing them out as typos (in say the variable name) can easily go undetected.

```
> source_power := 0;
> target_power := 0;
> pvars();
```

(We have omitted the Maple response.)

We shall consider the germ $f = (x, y^2, y^3, x^2y)$ discussed above. A basis for the space $J^3(L\mathcal{A}_e) \cdot f$ and its complementary space in $J^3(2, 4)$ are calculated using `jetcalc` in the usual way.

```
> f := [x1,x2^2,x2^3,x1^2*x2];
> jetcalc(f,3);
> codim;
                                   3
> pcomp();

                             [0, 0, x2, 0]
                             [0, 0, 0, x2]
                            [0, 0, x1 x2, 0]
```

Since f is 3-determined, the same reasoning as above for the $\mathcal{A}$ case shows that we can conclude that f has $\mathcal{A}_e$-codimension 3 and an $\mathcal{A}_e$-versal unfolding is

$$(x, y, u_1, u_2, u_3) \mapsto (x, y^2, y^3 + u_1 y + u_2 xy, x^2 y + u_3 y, u_1, u_2, u_3).$$

We note (reassuringly!) that our results are in agreement with "Wilson's" formula relating $\mathcal{A}$ and $\mathcal{A}_e$-codimension.

Proposition. Let $f : \mathbf{R}^n, S \to \mathbf{R}^p, 0$ be an $\mathcal{A}$-finite multigerm of multiplicity r. If f is not stable then the following relation holds:

$$\mathcal{A}_e\text{-codim} = \mathcal{A}\text{-codim} + r(p - n) - p.$$

This formula was proved in unpublished notes by Les Wilson and, in the case of singled branched germs ($r = 1$), in the survey article of Wall [27, p.510].

5.3 $\mathcal{R}$-Classification of Functions on the Cusp

In this section we describe how to perform calculations using weighted filtrations. The tutorial also mentions how `liealg` routines work. For brevity we will assume the reader has understood the material given in Section 5.2 and, while describing how one performs the relevant calculations using `Transversal_W`, we will not provide such detailed discussions and explanations. For example, we assume the reader knows what each routine does, how to interpret the output, what the `checklist` is, etc.; please refer back to Section 5.2 if necessary.

5.3.1 Background: Defining the Space $L = L\mathcal{R}(\mathcal{D})$

As our example we consider the classification of function-germs $\mathbf{C}^2, 0 \to \mathbf{C}, 0$ using coordinate changes in the source which preserve the cusp discriminant variety $\mathcal{D}$. This equivalence is induced by a subgroup of the standard $\mathcal{R}$ group which is commonly denoted by $\mathcal{R}(\mathcal{D})$. Arnold gave the beginning of this classification in [2], it was extended by Bruce and Kirk as described in [5, 16]. Note that we restrict to the complex case, this is mainly a technical convenience, the classification can be performed over the reals using the same methods.

For the appropriate classification theorems and technical machinery we refer the reader to [5, Section 4], this also provides useful background and appropriate references to existing work. Some notation: (u_1, u_2) will denote coordinates on $\mathbf{C}^2$, the variety defined by $4u_1^3 + 27u_2^2 = 0$ will be denoted by $\mathcal{D}$, and $\Theta(\mathcal{D})$ will denote the $\mathcal{O}_2$-module of vector fields tangent to $\mathcal{D}$. $\Theta(\mathcal{D})$ is a free module generated by the Saito vector fields

$$\theta_1 = 9u_2\partial/\partial u_1 - 2u_1^2\partial/\partial u_2, \qquad \theta_2 = 2u_1\partial/\partial u_1 + 3u_2\partial/\partial u_2.$$

It is natural to assign weights $(2, 3)$ to the source coordinates (u_1, u_2). (The weight 0 is always assigned to the target coordinate when dealing with *function* germs.) One easily verifies that the Saito vector fields θ_1 and θ_2 are then weighted homogeneous (with respect to these weights) having weight 1 and 0 as vector fields, respectively (see [5]). The vector field θ_2 of weight 0 is called the Euler vector field. Note that one can show that the Lie algebra $L\mathcal{R}(\mathcal{D})$ (as defined in [7, p.540]) is equal to $\Theta(\mathcal{D})$. Finally, we define L_{CT} to be the subspace of $\Theta(\mathcal{D})$ consisting of all vector fields of positive weight, thus

$$L_{CT} = \langle\theta_1\rangle + F^1\mathcal{O}_2.\langle\theta_2\rangle,$$

where $F^r\mathcal{O}_2$ denotes the ideal in $\mathcal{O}_2$ generated by the monomials of weight r and higher. The space L_{CT} is used for complete transversal and determinacy calculations with respect to the given weighted filtration of $\mathcal{O}_2$.

Figure 5.1 provides a listing of the code for the `liealg` routine named `cusp`. The global setup variable `liealg` is assigned the value `cusp` by the user, thus when the call

```
liealg(f,target_dim,tgtspace);
```

is made from within `jetcalc` it actually calls the routine `cusp`. The purpose of this routine is to define the generators θ_1 and θ_2 by specifying how they operate on the jet `f`. The result is stored in the table `tgtspace` (passed to `cusp` as an argument) as the entries `tgtspace[1]` and `tgtspace[2]`. Note that each entry of `tgtspace` must *itself* be of type 'table', even in the case when the target dimension is 1, so we *must* use expressions of the form

```
tgtspace[i][1] := ... ;
```

Another job of the `liealg` routine is to specify the source coordinates by assigning a Maple name to each entry in the global list `coords`, here the names `u1,u2` are used. In this particular case it is convenient to also assign the global lists `source_wt` and `target_wt` since the weighted filtration used in this classification is integral to all calculations we will perform. Similarly, it is also convenient to specify the nilpotent variables `R_nilp` and `L_nilp` (these will be discussed next). For further details on `liealg` routines refer to Section 4.2.2 and examine the program code of the examples which come with the package. (The argument `target_dim` is

```
# procedure to define Lie algebra tangent to the cusp discriminant

cusp := proc(f,target_dim,tgtspace)
   global L_nilp, R_nilp, coords, source_wt, target_wt;

      # DEFINE COORDINATES (global variable, data type 'list')
   if assigned(`u1`) or assigned(`u2`) then
      ERROR(`not all source coordinates are unassigned Maple names`);
   fi;
   coords := [u1,u2];

      # DEFINE LIE ALGEBRA GENERATING SET (data type 'table')
   tgtspace := table();
   tgtspace[1][1] := 9*u2*diff(f[1],u1) - 2*u1^2*diff(f[1],u2);
   tgtspace[2][1] := 2*u1*diff(f[1],u1) + 3*u2*diff(f[1],u2);

      # DEFINE WEIGHTS (global variables, data type 'list')
   source_wt := [2,3];
   target_wt := [0];

      # DEFINE NILPOTENT VECTORS (global variables, data type 'list')
   R_nilp := [ [1,1] ];
   L_nilp := [];

      # RETURN NULL
   NULL;

end:
```

Figure 5.1: A Listing of the `cusp` Routine.

calculated in `jetcalc` and passed into the `liealg` routine for convenience, for example the `liealg` routine `stdjacobian` requires use of it. This argument must always be present but is rarely used.)

It is now up to the user to assign the global setup variables to define the space $L = L\mathcal{R}(\mathcal{D})$ or $L = L_{CT}$ as required. Observe that

$$L\mathcal{R}(\mathcal{D}) \;=\; \langle\theta_1,\theta_2\rangle, \qquad L_{CT} \;=\; F^1\mathcal{O}_2.\langle\theta_1,\theta_2\rangle \;\oplus\; \mathbf{C}\{\theta_1\}.$$

Thus, L_{CT} is specified by multiplying the ideal $\langle\theta_1,\theta_2\rangle$ by $F^1\mathcal{O}_2$ and then including constant multiples of the 'nilpotent' vector θ_1. The space $L = L\mathcal{R}(\mathcal{D})$ is therefore defined by setting

```
> liealg := cusp;
> equiv := R;
> compltrans := false;
> source_power := 0;
> target_power := 0;
> nilp := false;
> jetcalc_verbosity := 1;
> pvars();
```

and the space $L = L_{CT}$ is defined by

```
> liealg := cusp;
> equiv := R;
> compltrans := true;
> source_power := 1;
> target_power := 0;
> nilp := true;
> jetcalc_verbosity := 1;
> pvars();
```

Recall that the remaining variables required by `jetcalc` (namely `R_nilp`, `L_nilp`, `source_wt` and `target_wt`) are set within the routine `cusp` for convenience.

Some Remarks. `R_nilp` is assigned the value `[ [1,1] ]` which means include one extra 'nilpotent' term, namely `[1,1]` which denotes $1 \times \theta_1 = \theta_1$. Whether the actual nilpotent terms specified by `R_nilp` and `L_nilp` are actually included is decided by the user by setting the global variable `nilp` as required. (In the case $L = L\mathcal{R}(\mathcal{D})$ above this setting does not effect L since, with `source_power` set equal to 0, L will already contain this vector. However, it is good practice to set `nilp` to `false`. In the case $L = L_{CT}$ the setting for `nilp` is of course crucial.) Note that `L_nilp` *must* be assigned the value of the empty list since `jetcalc` performs a certain amount of type-checking on the global variables. Similarly, the value given to `target_power` is irrelevant in this example as we are using $\mathcal{R}$-equivalence, but it must be assigned a non-negative integer. The value of `compltrans` does not affect how the space L is defined; the values given above were chosen only to suit the typical calculations which will be performed using `jetcalc` with each of the two spaces. Similarly, the value of `jetcalc_verbosity` does not effect L but must be set; the value 1 is standard. Finally, it is advisable to check the settings, just in case any typos went unnoticed. This may be achieved using the function `pvars`, thought note that the variables `R_nilp`, `L_nilp`, `source_wt` and `target_wt` will not have been assigned until at least one call to `jetcalc` has been made.

5.3.2 Some Calculations

We will begin by stating the necessary results from classification theory. For brevity the results are specialised to the case of the cusp discriminant; see [5, 16] for a general formulation.

Weight	2	3	4	5	6	7	8	9
Monomials	u_1	u_2	u_1^2	u_1u_2	u_1^3, u_2^2	$u_1^2u_2$	u_1^4, $u_1u_2^2$	$u_1^3u_2$, u_2^3

Table 5.1: Monomials up to Weight 9 for the A_2 Filtration.

Theorem 5.1 (Complete Transversal) *Let $f : \mathbf{C}^2, 0 \to \mathbf{C}, 0$, $k \geq 1$ and T be a subspace of $F^{k+1}\mathcal{O}_2$ such that*

$$F^{k+1}\mathcal{O}_2 \subset T + L_{CT} \cdot f + F^{k+2}\mathcal{O}_2,$$

then any germ f_1 with $f_1 - f \in F^{k+1}\mathcal{O}_2$ is $F^1\mathcal{R}(\mathcal{D})$-equivalent to $f + t + \phi$ for some $t \in T$ and $\phi \in F^{k+2}\mathcal{O}_2$.

Note that in [5] the notation h, h_1 is used where f, f_1 is used above (f had already been used to define $\mathcal{D}$ in that article). We chose the above notation for consistency with our examples.

As a corollary we obtain the following determinacy result. Here we quote a finite dimensional version: w_{max} denotes the maximum weight — in the present case this is 3 — so to check determinacy we can therefore work in the weighted $(k+3)$-jet-space. For example, it would be enough to show all CTs of weight $k+1$ to weight $k+3$ are empty.

Corollary 5.2 *A germ $f : \mathbf{C}^2, 0 \to \mathbf{C}, 0$ is k-$\mathcal{R}(\mathcal{D})$-determined if*

$$F^{k+1}\mathcal{O}_2 \subset L_{CT} \cdot f + F^{k+1+w_{max}}\mathcal{O}_2.$$

Note that one extremely useful class of coordinate change denied us by the complete transversal method is scaling. Generally such coordinate changes will not preserve the discriminant $\mathcal{D}$. There is, however, one scale change that does.

Proposition 5.3 *For $t \in \mathbf{C}$, $t \neq 0$, the map-germ $\mathbf{C}^2, 0 \to \mathbf{C}^2, 0$ defined by*

$$(u_1, u_2) \mapsto (t^2u_1, t^3u_2)$$

is an element of the group $\mathcal{R}(\mathcal{D})$.

Finally, for reference, Table 5.1 lists the monomials of lower weight. These can be displayed using the command `pmons(1,9);` (once the appropriate global variables have been set up).

Now to an example using `Transversal_W`. We will make implicit use of the above results without further qualification. The package is used in pretty much the same way as the standard `Transversal` package. Indeed, although certain functions in the packages, like `jetcalc`, differ, they store their results in exactly the same way so that many of the 'output' routines can be shared across the packages and used in precisely the same manner; see Section 4.4 for more details.

Remember to initialise the package in a *new* Maple session via the command

```
> restart;
> with(transversal_W);
```

Consider the 5-jet 0: a 6-CT is $\{u_1^3, u_2^2\}$. Applying the scaling coordinate changes described in Proposition 5.3 we can reduce the resulting family to the three cases: $u_1^3 + au_2^2$, u_2^2, 0. We will consider the first in what follows. The parameter a is a modulus; one should really check this now (with $L = L\mathcal{R}(\mathcal{D})$) but we will proceed with the classification (in fact more moduli occur as we will show later). Thus, set the global variables to define $L = L_{CT}$ as described in Section 5.3.1 and calculate a 7-CT.

```
> f := [u1^3+a*u2^2];
> jetcalc(f,7);
> pcomp();
                      *** THE NORMAL SPACE IS EMPTY ***
> plist();

                                 #1, 27 - 4 a
```

Note that in this, and subsequent, calculations `jetcalc` outputs the usual warning:

```
        WARNING: global variable 'checklist' is non-empty !!!
```

Thus, if $4a \neq 27$ then the 7-CT is empty. Continuing with this case ...

```
> jetcalc(f,8);
> pcomp();

                                         4
                                     [u1 ]

> plist();

                                 #1, 27 - 4 a
                                 #2, 6 a
```

Thus, an 8-CT is $\{u_1^4\}$ provided $4a \neq 27$ and $a \neq 0$. However, there is usually more than one choice for a CT (just as there is more than one choice for a basis) and the above choice is determined by the ordering of the monomials of weight 8 employed by `jetcalc`. Another choice of 8-CT is $\{u_1 u_2^2\}$. The latter was used in published material on this work so we do the same here. (This choice is more natural if one performs the calculations by hand because it is then clear that the condition $a \neq 0$ is not necessary; this will be demonstrated using the computer as well. However, the condition still appears at the higher jet-levels when one performs determinacy calculations. Either choice is acceptable and, of course, gives the same list of orbits. Such discrepancies are bound to occur when comparing automated/computer calculations with hand calculations.) Continuing ... we can indeed show that $\{u_1u_2^2\}$ is a valid alternative choice for an 8-CT using the function `ptangent` to inspect the basis for $J^8L_{CT}\cdot f$ calculated by `jetcalc`. If this basis is large we would restrict the output from `ptangent` by supplying the argument `[u1^4]` so that only basis elements containing the monomial term u_1^4 are output. However, in this case the basis has dimension 2 so we output the whole thing.

```
> basis_dim;
                                  2
> ptangent();
                ***   basis for tangent space   ***
 vectors output as monomial terms and corresponding coefficients

                               vector1
                                     2
                         27 - 4 a, [u2 u1 ]

                               vector2
                                   2
                           6 a, [u1 u2 ]
                                  4
                             6, [u1 ]
```

This indicates that a basis consists of the jets $(27-4a)u_1^2u_2$ and $6au_1u_2^2+6u_1^4$. Thus it follows that $\{u_1u_2^2\}$ is a perfectly good choice for an 8-CT instead of $\{u_1^4\}$ (and it is clear where the condition $a\neq 0$ originally came from).

After this small digression we can therefore surmise that the $J^8\mathcal{R}(\mathcal{D})$-orbits lying over the (weighted) 7-jet $u_1^3+au_2^2$ form the family $u_1^3+au_2^2+bu_1u_2^2$. This is a bimodular family though, for brevity, we will continue with the classification using CTs and return to the modality question shortly.

```
> f := [u1^3+a*u2^2+b*u1*u2^2];
> jetcalc(f,9);
> jetcalc(f,10);
> jetcalc(f,11);
```

After each call to `jetcalc` we must, of course, call `pcomp` and `plist`: we find that the 9, 10 and 11-CTs are empty provided $4a\neq 27$ and $a\neq 0$. Thus

$$F^9\mathcal{O}_2\subset L_{CT}\cdot f+F^{12}\mathcal{O}_2$$

and $u_1^3+au_2^2+bu_1u_2^2$ is (weighted) 8-$\mathcal{R}(\mathcal{D})$-determined under these conditions.

We now return to the modality questions and define $L=L\mathcal{R}(\mathcal{D})$. With the present settings (defining $L=L_{CT}$) this may be achieved by setting `source_power := 0` (and `nilp := false`, though this is not strictly necessary). In addition, for the type of calculations we have in mind, we set `compltrans := false`. Now we calculate $L\mathcal{R}(\mathcal{D})\cdot f$ in the 8-jet-space.

```
> f;
                              3       2           2
                          [u1  + a u2  + b u1 u2 ]
> jetcalc(f,8);
> plist();
                                  #1, 6 a
                                  #2, 27 - 4 a
                                  #3, 6 a
> intangent([u2^2],[u1*u1^2]);
                                    false
> pcomp();
                                     [1]
                                    [u1]
                                    [u2]
                                       2
                                    [u1 ]
                                   [u1 u2]
                                       3
                                    [u1 ]
                                       4
                                    [u1 ]
```

Recall that the $\mathcal{R}(\mathcal{D})$-codimension of a germ f is defined to be $\dim_{\mathbf{C}} \mathcal{O}_2/L\mathcal{R}(\mathcal{D})\cdot f$. Thus we may conclude that $u_1^3 + au_2^2 + bu_1u_2^2$ is a bimodular family, 8-determined, of codimension 7, provided $4a \neq 27$ and $a \neq 0$. (Note that the call to `intangent` above also outputs a warning about the degenerate conditions. However, such conditions do not affect the modality, the bottom line is that these conditions just signify when the calculation breaks downs, resulting in a possible degeneration in the tangent space.)

Exercise. Returning to the 8-jet $u_1^3+au_2^2+bu_1u_2^2$, show that if $a=0$ then a 9-CT is $u_1^3 + bu_1u_2^2 + cu_2^3$. Continue the classification to obtain the trimodular family of 12-jets $u_1^3 + bu_1u_2^2 + cu_2^3 + du_2^4$, 12-determined, codimension 9, provided $b \neq 0$ and $4b^3 + 27c^2 \neq 0$.

Exercise. Returning to the 6-jet $u_1^3 + au_2^2$, show that if $a = 27/4$ then a 7-CT is $u_1^3 + \frac{27}{4}u_2^2 + bu_1^2u_2$. Continue the classification to obtain the bimodular family of 9-jets $u_1^3 + \frac{27}{4}u_2^2 + bu_1^2u_2 + cu_1^3u_2$, 9-determined, codimension 8, provided $b \neq 0$.

The above give the initial members of quite complex series lying above the respective jets. The following classification was obtained in [16]; see also [5].

Theorem 5.4 *Every function-germ $h : \mathbf{C}^2, 0 \to \mathbf{C}, 0$ on the cusp discriminant variety $\mathcal{D}$ of $\mathcal{R}(\mathcal{D})$-modality ≤ 2 is $\mathcal{R}(\mathcal{D})$-equivalent to one of the following finitely determined germs. The first germs of modality ≥ 3 to occur during the classification* *are given in the second section of the table; they occur as the initial singularities in the stated series. Note that u_1 is the only stable singularity.*

Singularity	Determinacy Degree	Codim
u_1	2	1
u_2	3	2
$u_1^2 + au_2^n$	$3n, \quad n \geq 2, \quad a \neq 0$	$n+2$
$u_1u_2 + au_1^3$	6	5
$u_1^3 + au_2^2 + bu_1u_2^2$	$8, \quad 4a \neq 27, \quad a \neq 0$	7
$u_1^3 + \frac{27}{4}u_2^2 + au_1^nu_2 + bu_1^{n+1}u_2$	$2n+5, \quad n \geq 2, \quad a \neq 0$	$2n+4$
$u_1^3 + \frac{27}{4}u_2^2 + au_1^n + bu_1^{n+1}$	$2n+2, \quad n \geq 4, \quad a \neq 0$	$2n+1$
$u_2^2 + au_1^n + bu_1^{n+1}$	$2n+2, \quad n \geq 4, \quad a \neq 0$	$n+4$
$u_1^3 + au_1u_2^n + bu_2^{n+1} + cu_2^{n+2}$	$3n+6, \quad n \geq 2, \quad a, b \neq 0\,(\dagger)$	$2n+5$
$u_1^3 + au_2^n + bu_1u_2^n$	$3n+2, \quad n \geq 3, \quad a \neq 0$	$2n+3$
$u_1^2u_2 + au_1^4 + bu_2^n + cu_2^{n+1}$	$3n+3, \quad n \geq 3, \quad b \neq 0$	$n+7$

($\dagger$) *For the case $n = 2$ the condition $b \neq 0$ needs to be replaced by $4a^3 + 27b^2 \neq 0$.*

In closing we note that it is possible to perform this classification by hand and the above was chosen as an instructive example; see [16]. However, even in the early stages of the classification, when the space $J^kL_{CT} \cdot f$ is generally of a low dimension, this can be tedious as one needs to keep track of which monomials generate the various weighted jet-spaces. The determinacy calculations can be particularly tedious, and the presence of moduli, even at the lower levels, complicates matters. At higher jet-levels the use of `Transversal_W` is particularly welcome!

5.4 In Preparation: Multigerms and Lowerable Fields

As mentioned in Section 4.3, other extensions to the standard `Transversal` package exist which deal with multigerms and equivalence relations defined using lowerable diffeomorphisms. These packages are not publicly available at present. However, if you are interested then contact me and I will try and provide some version together with a few examples.

Bibliography

[1] Arnold, V.I., Critical points of smooth functions and their normal forms, *Russian Math. Surveys*, **30**:5 (1975), 1–75.

[2] Arnold, V.I, Wavefront evolution and equivariant Morse lemma, *Comm. Pure Appl. Maths.*, **29** (1976), 557–582.

[3] Bayer, D., Stillman, M., *Macaulay: A System for Computation in Algebraic Geometry and Commutative Algebra*, 1982–1994. Source and object code available for Unix and Macintosh computers. Contact the authors, or download from `math.harvard.edu` via anonymous ftp. See also:
`http://www.math.uiuc.edu/Macaulay2`

[4] Bruce, J.W., Generic geometry and duality, in *Singularities, Lille 1991*, Brasselet, J-P. (ed.), London Math. Soc. Lecture Note Series 201, (Cambridge University Press, 1994).

[5] Bruce, J.W., Kirk, N.P., du Plessis, A.A., Complete transversals and the classification of singularities, *Nonlinearity*, **10** (1997), 253–275.

[6] Bruce, J.W., Kirk, N.P., West, J.M., Classification of map-germs from surfaces to four-space, Preprint, *University of Liverpool*, (1995).

[7] Bruce, J.W., du Plessis, A.A., Wall, C.T.C., Determinacy and unipotency, *Invent. Math.*, **88** (1987), 521–554.

[8] Char, B.W., Geddes, K.O., Gonnet, G.H., Leong, B.L., Monagan, M.B., Watt, S.M., *Maple V Language Reference Manual*, (Springer-Verlag and Waterloo Maple Publishing, 1991).

[9] Cowell, R.G., Wright, F.J., CATFACT: computer algebraic tools for applications of catastrophe theory, *Proc. Eurocal '87, Lecture Notes in Computer Science*, (Springer, 1989), 72–81.

[10] Damon, J.N., The unfolding and determinacy theorems for subgroups of $\mathcal{A}$ and $\mathcal{K}$, *Mem. Amer. Math. Soc.*, **306** (1984). Also: *Singularities, Proc. Symp. Pure Math.*, Amer. Math. Soc. Providence, R.I., **40** (part 1) (1983), 233–254.

[11] Damon, J.N., $\mathcal{A}$-equivalence and the equivalence of sections of images and discriminants, in *Singularity Theory and its Applications, Part I, Warwick 1989*, Mond, D.M.Q., Montaldi, J. (eds.), Springer Lecture Notes in Math 1462, (Springer 1991), 93–121.

[12] Fox, L., *An Introduction to Numerical Linear Algebra*, Clarendon Press, (Oxford, 1964).

[13] Hawes, W., Multi-dimensional motions of the plane and space, Ph.D. thesis, *University of Liverpool*, (1994).

[14] Hobbs, C.A., Kirk, N.P., On the classification and bifurcation of multigerms of maps from surfaces to 3-space, to appear in *Mathematica Scandinavica.*

[15] Houston, K., Kirk, N.P., On the classification and geometry of map-germs from 3-space to 4-space, to appear in *Singularities, Liverpool 1996*, Proceedings of the European Singularities Conference in honour of C.T.C. Wall on the occasion of his 60th Birthday, Bruce, J.W., Mond. D.M.Q., (eds.), (Cambridge University Press, 1999).

[16] Kirk, N.P., Computational aspects of singularity theory, Ph.D. thesis, *University of Liverpool*, (1993).

[17] Kirk, N.P., Computational aspects of classifying singularities, Preprint, *University of Liverpool*, (1998).

[18] Martinet, J., *Singularities of Smooth Functions and Maps*, London Math. Soc. Lecture Note Series 58, (Cambridge University Press, 1982).

[19] Millington, K., Wright, F.J., Algebraic computations in elementary catastrophe theory, *Eurocal '85, Lecture Notes in Computer Science*, **204**, (Springer, 1985), 116–125.

[20] Mond, D.M.Q., On the classification of germs of maps from $\mathbf{R}^2$ to $\mathbf{R}^3$, *Proc. London Math. Soc.* (3), **50** (1985), 333–369.

[21] du Plessis, A.A., On the determinacy of smooth map-germs, *Invent. Math.*, **58** (1980), 107–160.

[22] Poston, T., Stewart, I.N., *Catastrophe Theory and its Applications*, (Pitman, 1978).

[23] Ratcliffe, D., A classification of map-germs $\mathbf{C}^2, 0 \to \mathbf{C}^3, 0$ up to $\mathcal{A}$-equivalence, Preprint, *University of Warwick*, (1994).

[24] Rieger, J.H., Families of maps from the plane to the plane, *J. London Math. Soc.* (2), **36** (1987), 351–369.

[25] The SINGULAR project; directed and coordinated by Greuel, G.-M., Pfister, G., Schoenemann, H., Department of Mathematics, University of Kaiserslautern. Latest information available from `http://www.mathematik.uni-kl.de/~zca/Singular`

[26] Tari, F., Recognition of $\mathcal{K}$-singularities of functions, *Experimental Mathematics*, **1** (3) (1992), 225–229.

[27] Wall, C.T.C., Finite determinacy of smooth map-germs, *Bull. London Math. Soc.*, **13** (1981), 481–539.

[28] West, J.M., The differential geometry of the crosscap, Ph.D. thesis, *University of Liverpool*, (1995).