Merging Solutions of Polynomial Algebraic Systems

Thomas Wolf
Department of Mathematics
Brock University
St.Catharines, Ontario, Canada L2S 3A1
twolf@brocku.ca

ABSTRACT

In solving large polynomial algebraic systems that are too big for standard Gröbner basis techniques one way to make progress is to introduce case distinctions. This divide and conquer technique can be beneficial if the algorithms and computer programs know how to take advantage of inequalities. A further hurdle is the form of the resulting general solutions which often have unnecessarily many branches. In this paper we discuss a procedure to merge solutions by dropping inequalities which are associated with them and, if necessary, by re-parametrizing solutions.

In the appendix the usefulness of the procedure is demonstrated in the classification of quadratic Hamiltonians with a Lie-Poisson bracket $e(3)$. This application required the solution of algebraic systems with over 200 unknowns, 450 equations and between 5000 and 9000 terms.

Categories and Subject Descriptors

I.1 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation; I.1.2 [**Algorithms**]: Algebraic Algorithms—*merging solutions of polynomial algebraic problems*

General Terms

Algorithms

Keywords

solving polynomial systems, inequalities

1. INTRODUCTION

If algebraic systems to be solved get very large then a way to convert memory needs into increased execution time is to introduce artificial case distinctions. These could be tried for the only reason that some variable, say x_j, occurs very often and one expects large simplifications in setting x_j to zero and non-zero[1]. On the other hand Gaussian elimination could require case distinctions. For example, if a, b, c are unknowns and one of the equations of a polynomial system is $0 = ab + c^3$ then the case distinction $a = 0, a \neq 0$ leads in both cases to a useful simplification of the remaining system, in one case using the substitutions $a = c = 0$ and in the other case the substitution $b = -c^3/a$ and the inequality $a \neq 0$. Less artificial are case distinctions resulting from factorizable equations where one distinguishes between the two cases that one factor is zero or non-zero and therefore can be dropped. For the large polynomial systems as described in appendix A both circumstances arise. In these bi-linear systems, unknowns occur often only with first degree so eliminations become possible if coefficients are known to be non-zero. It also happens relatively often that equations can be factorized.

The problem to be addressed in this contribution concerns the form of the general solution. Introducing artificial case distinctions will usually result in a general solution consisting of many more branches than one would obtain by computing a Gröbner basis and a primary or triangular decomposition. The aim of the paper is to describe an effective procedure that merges special solutions of polynomial algebraic systems by dropping inequalities that are attached to them and, if necessary, by a re-parametrization of solutions.

In a ground laying and very detailed paper [1] William Sit described algorithms to solve parametric linear systems. Applications like the one in appendix A fall into this class. The purpose of his work was exactly to avoid the generation of unnecessarily many branches of solutions and to avoid the need for merging. Unfortunately, for polynomial systems with several hundred unknowns and thousands of terms, although sparse and over-determined, the computational tools used in [1], like the computation of Gröbner bases can not be performed. For problems of this size the computation of a general solution is mainly geared by complexity issues, like the length of equations to be manipulated and less by algorithmic issues.

For example, in applications described in [2] we found that if any case distinctions are to be done later in the computation then for problems which are too large for normal computations it is best to do these case distinctions at the very beginning, leading to short conditions which can be used to

[1]If a computer program could not take advantage of inequalities then case splitting would not make sense. The application in appendix A generates large over-determined systems for which the knowledge of inequalities is very beneficial to the program CRACK that is used to solve these systems.

shorten other conditions and so on instead of introducing case distinctions when already many eliminations have been made using long expressions which could have been much shorter if the case distinction would have been introduced earlier. Early growth of expressions introducing higher non-linearity is not completely reversed by setting some variables to zero later in the computation.

In short, the strategy we followed successfully is to give the solution process as much freedom as possible and to face the task of cleaning up the general solution by merging special branches afterwards.

After setting up a notation in the following section the merging procedure is explained in section 3. Preliminary tests and two instructive examples are given in subsection 3.1 followed by an outline of the method in subsection 3.2 and the computational steps in subsection 3.3. In appendix A a recent larger application is described where the method proved to be useful.

2. NOTATION

The formulation of the algorithm and the notation are tailored to their use which is the merging of branches of solutions as they are produced by the package CRACK ([2]).

For a polynomial algebraic system and a set X of unknowns $X = \{x_j\}$ we will call a solution S_i a list $S_i = \{E_i, A_i, F_i, U_i\}$ where

- E_i is a set $E_i = \{e_{i1}, \ldots, e_{in_i}\}$ of unsolved polynomial equations $0 = e_{ij}$, which is a Gröbner basis according to some lexicographical ordering.[2]
- A_i is a list $A_i = \{x_j = R_{ij}(x_k), \ldots\}$ of assignments of unknowns x_j by rational expressions R_{ij}. None of the x_j on left hand sides of assignment A_i is supposed to appear in any expression of E_i (otherwise the substitution for x_j is to be carried out in E_i). Assignments A_i will also be used in form of a list of vanishing expressions $Z_i = \{x_j - R_{ij}(x_k), \ldots\}$.
- F_i is a list of unknowns that appear only on the right hand sides R_{ij} of assignments in A_i and/or in the unsolved conditions E_i and/or in U_i below. Apart from having to satisfy any conditions $0 = e_{ij}$ from E_i they are free.
- U_i is a list of inequalities associated with the solution S_i.

Let X_i be the set of x_j that occur on left hand sides in A_i, i.e. $X_i = X \setminus F_i$. We partition $X_i = {}^*X_i + X_i^*$ (and by that $A_i = {}^*A_i + A_i^*$ and $Z_i = {}^*Z_i + Z_i^*$) through ${}^*X_1 = {}^*X_2 = X_1 \cap X_2$ and therefore $X_1^* \cap X_2^* = \emptyset$ and consequently

$$X_1^* \subset F_2, \quad X_2^* \subset F_1. \tag{1}$$

An over-line accent will mark a re-parametrized solution $\overline{S_2} = \{\overline{E_2}, \overline{A_2}, \overline{F_2}, \overline{U_2}\}$. With $\#()$ we denote the number of elements of a set, e.g. $\#(F_i)$ is the number of unknowns that are free or undetermined in solution S_i. With num() we will denote the numerator of a rational expression. The ideal of a set of polynomials is denoted by I, for example, the ideal of E_i by $I(E_i)$. With the notation $A_i(H)$ we will indicate that the list of substitutions A_i is carried out in a list H of expressions, for example, $A_1(E_2)$. With := we will denote assignments as used in programming languages, e.g. $Z_2 := A_1(Z_2)$ stands for performing substitutions A_1 in Z_2 and calling the new list Z_2.

[2] The condition of being a Gröbner basis wrt. some lexicographical ordering is only necessary for the quick preliminary merging condition (2).

3. MERGING SOLUTIONS

3.1 Preliminary Considerations

When is a solution S_1 contained in a solution S_2 after dropping inequalities from S_2 or re-parametrizing S_2? If assignments A_1 are regular in solution S_2 (i.e. $A_1(E_2)$ and $A_1(Z_2)$ do not produce singularities through zero denominators), then we say S_2 generalizes S_1 if and only if all resulting conditions are consequences of equations E_1:

$$\text{num}(A_1(E_2)) \subset I(E_1), \quad \text{num}(A_1(Z_2)) \subset I(E_1).$$

To perform substitutions, to take a numerator and test membership of the resulting polynomials in the ideal $I(E_1)$ does not pose a problem and can be done efficiently.

This leaves us with the case that $A_1(Z_2)$ creates zero denominators. In that case we have at least the following necessary conditions for merger:

1. If
$$\#(F_1) - \#(E_1) > \#(F_2) - \#(E_2) \tag{2}$$
then S_1 can not be a special case of S_2 as a special case can not have a higher number of free parameters. For this condition to be correct we assumed above that E_i is a Gröbner Bases according to some lexicographical ordering.

2. Substitutions $A_1(E_2)$ are always regular as E_2 is a list of polynomials and any variable on a left hand side of an assignment in A_1 does not occur on a rhs of another assignment in A_1. Therefore
$$\text{num}(A_1(E_2)) \subset I(E_1) \tag{3}$$
is an easy to test necessary condition.

3. If $A_1(Z_2)$ is singular then at least a necessary test can be performed by taking repeatedly the numerator of all expressions in Z_2 before performing each of the substitutions A_1 and finally testing:
$$\text{num}(A_1(\text{num}(Z_2))) \subset I(E_1). \tag{4}$$

All three tests are only necessary and not sufficient as can be seen from the following 2 examples, both arising from the application of appendix A and both giving singularities when computing $A_1(Z_2)$.

Example 1:
Solutions S_1, S_2 involve among other variables the unknowns $x_1, \ldots, x_5$ (which we re-named to make the example better

readable). We have

$$\begin{array}{lll} S_1: & E_1: & \text{no unsolved equation} \\ & A_1: & x_1 = x_2 = x_3 = 0 \\ & F_1: & x_4, x_5 \ \text{ are free} \\ & U_1: & \text{no inequalities} \\ S_2: & E_2: & \text{no unsolved equation} \\ & A_2: & x_2 = -(x_3x_5)/x_1, \quad x_4 = (x_3x_5)/x_1, \\ & F_2: & x_1, x_3, x_5 \ \text{ are free} \\ & U_2: & \text{no inequalities} \end{array}$$

Both solutions satisfy conditions (2)-(3) but solution S_2 also satisfies the condition $x_4 + x_2 = 0$ which would be a restriction on the generality of solution S_1. This observation is unaffected by whatever re-parametrization one would choose for solution S_2, so S_1 can not be a special case of S_2.

Example 2:
Solutions S_1, S_2 involve among other variables the unknowns $x_1, \ldots, x_5$. We have

$$\begin{array}{lll} S_1: & E_1: & \text{no unsolved equation} \qquad (5) \\ & A_1: & x_5 = x_6 = 0, \qquad (6) \\ & & x_7 = (x_4x_2)/(2x_1), \qquad (7) \\ & F_1: & x_1, x_2, x_3, x_4 \ \text{ are free} \\ & U_1: & x_1 \neq 0, \\ S_2: & E_2: & \text{no unsolved equation} \qquad (8) \\ & A_2: & x_2 = (x_6^2 - 4x_5x_7)/x_6, \qquad (9) \\ & & x_4 = (x_5 - x_1)x_6/(2x_5), \qquad (10) \\ & F_2: & x_1, x_3, x_5, x_6, x_7 \ \text{ are free} \\ & U_2: & x_5 - x_1 \neq 0, \ x_5 \neq 0, \ x_6 \neq 0, \end{array}$$

It is possible to merge both solutions by re-parametrizing S_2, i.e. instead of parametrizing x_2, x_4 in terms of x_1, x_5, x_6, x_7 to have x_6, x_7 parametrized by x_1, x_2, x_4, x_5 by inverting (10), (9) and by dropping the inequalities $x_5 \neq 0$ and $x_6 \neq 0$:

$$\begin{array}{lll} \overline{S_2}: & \overline{E_2}: & \text{no unsolved equation} \qquad (11) \\ & \overline{A_2}: & x_6 = \dfrac{2x_4x_5}{x_5 - x_1} \qquad (12) \\ & & x_7 = \dfrac{x_4^2x_5}{(x_5 - x_1)^2} - \dfrac{x_2x_4}{2(x_5 - x_1)}. \qquad (13) \\ & \overline{F_2}: & x_1, x_2, x_3, x_4, x_5 \ \text{ are free} \\ & \overline{U_2}: & x_5 - x_1 \neq 0. \end{array}$$

The assignments (6) do not cause problems anymore when done to the new form of solution S_2. Equation (12) becomes an identity and (13) reduces to (7).

From now on we are concerned only with situations where the tests (2)-(4) are passed but where $A_1(Z_2)$ creates singularities.

3.2 The Strategy

In analogy to example 2, the procedure described in the following subsection computes a re-parametrization of the assignments A_2 of solution S_2 to assignments $\overline{A_2}$ such that $A_1(\overline{Z_2})$ is singularity free and $A_1(\overline{A_2}(E_2))$ is singularity free. Then a simple membership test can establish whether the numerators of all these expressions are in the ideal of E_1. If that is the case then S_1 can be dropped as it is included in solution $\overline{S_2} = \{\overline{E_2}, \overline{A_2}, \overline{F_2}, \overline{U_2}\}$ where $\overline{A_2}$ is the new list of assignments, $\overline{E_2} = \text{num}(\overline{A_2}(E_2))$ is the new list of unsolved equations, $\overline{F_2}$ are the new parameters or undetermined unknowns and $\overline{U_2}$ is the subset of expressions of $\text{num}(\overline{A_2}(U_2))$ for which $\text{num}(A_1(\overline{U_2})) \not\subset I(E_1)$.

3.3 The Merging Steps

In the following the *steps of the algorithm* are distinguished from comments using an italic font.

• The computation starts with performing *preliminary necessary tests 1. and 2.* as described in section 3.1. If one of them fails then a merger is not possible.

• We recall from section 2 that ${}^*A_1, {}^*A_2$ share the same x_j on left hand sides of their assignments. The next step is to *perform substitutions* $Z_2 :={}^*A_1(Z_2)$.

Do we loose generality in performing these substitutions before considering parametrizations? The substituted $x_j \in {}^*X_1$ occured only on left hand sides in ${}^*A_1, {}^*A_2$ (i.e. only in ${}^*Z_i$, not Z_i^*) and nowhere else in the remaining objects of interest: $E_1, E_2, A_1^*, A_2^*, Z_2, Z_2^*$. The expressions in the new Z_2 would have to vanish whatever the re-parametrization would be. We therefore do not loose generality. Instead, because the number of free parameters in a solution, for example $\#(F_1)$ for solution S_1, is usually considerably smaller than the number of unknowns $\#(X)$ and because of relation (1) we have $\#(X_2^*) \ll \#(X)$. Therefore, eliminating most of the x_j from further consideration by doing ${}^*A_1(Z_2)$ narrows and simplifies the choice for re-parametrizations below without loosing generality.

• From section 2 we recall that $A_1^* = A_1 \setminus {}^*A_1$. *On a copy of Z_2 test whether the remaining substitutions $A_1^*(Z_2)$ are regular. If so, S_1, S_2 can be merged iff num$(A_1^*(Z_2)) \subset I(E_1)$.*

• If execution reaches this point then substitutions $A_1^*(Z_2)$ are singular. *The preliminary necessary test 3. of section 3.1 is done next* performing each substitution of A_1 only on the numerator of the result of the previous substitution to avoid singularities.

• If execution reaches this point then a re-parametrization of S_2 is necessary to decide whether merging is possible.

What form should the re-parametrized solution $\overline{S_2}$ take to make obvious that S_1 is contained? We will require that all x_j on left hand sides of $\overline{A_2}$ (i.e. all $x_j \in \overline{X_2}$) are also on left hand sides of A_1, i.e. $\overline{X_2} \subset X_1$ and that $A_1(\overline{Z_2})$ is regular and further $\text{num}(A_1(\overline{Z_2})) \subset I(E_1)$.

How can S_2 be re-parametrized to reach this? Assignments ${}^*A_2$ can not be changed as the x_j on left hand sides of ${}^*A_2$ are assigned a value in A_1 too. All that can be done is to re-write A_2^*, i.e. to solve the system of equations Z_2^* (i.e. the equations when setting all expressions in Z_2^* to zero) for other variables than X_2^*. Z_2^* has to be solved for a subset of X_1^* with $\#(Z_2^*)$ many variables. In a complete investigation Z_2^* would be solved for any such subset, for example by the methods described in [1][3]. As only one regular re-parametrization is needed, in the program Crack a less complete but efficient and in all applications so far successful search is performed. For example, in the application of appendix A it found all possible mergings. In addition, at this stage of the computation the program gives notice that

[3]In the application of appendix A the 200 unknown coefficients in the ansatz for the first integral appear linear in the beginning, during the computation and in each solution. That means, at all times the system can be seen as a parametric linear system as considered in [1].

the 3 preliminary tests of section 3.1 have all been passed so that if no suitable re-parametrization is found below then the user could check manually.

• *The* x_j *in the list* X_1^* *are sorted so that those* x_j *come first which produce singularities when substituted through* A_1^* *in* Z_2. The purpose is to try solving Z_2^* for such variables first to avoid singularities in the later substitutions of $A_1^*(Z_2)$.

A side issue:
Successful re-parametrizations are not unique. As CRACK has been used to solve bi-linear systems with the strategy to keep one of the two linearly occuring sets of variables linear during computation, this aim is also pursued in a re-parametrization. As a consequence, variables $x_j \in X_1^*$ which occur linearly in Z_2 come first in a re-sorted X_1^*.

• *Initialize*

$$\overline{E_2} := E_2, \quad \overline{A_2} :=^* A_2, \quad \overline{F_2} := X_2^* \cup F_2, \quad \overline{U_2} := U_2.$$

Solve each individual equation $0 = z_{2i}^*, \quad z_{2i}^* \in Z_2^*$ *for one variable* $x_j \in X_1^*$ *to get its solution in the form* $x_j = \overline{R_{2j}}$. *Do this by checking all variables in* X_1^* *in the order specified above, to find one variable that*

∘ *occurs linearly in* z_{2i}^*, *and*

∘ *has a coefficient* C *which is non-zero according to* U_2, *and satisfies further* $A_1^*(C) \neq 0$, *and*

∘ *gives a regular substitution* $x_j = \overline{R_{2j}}$ *applied to* Z_2.

If no variable x_j *and consequently no such solution* $x_j = \overline{R_{2j}}$ *is found then the method stops without finding a merger. If for the equation* $0 = z_{2i}^*$ *under consideration a variable* x_j *is found, then do*

$$\begin{aligned}
Z_2 &:= \mathrm{num}((x_j = \overline{R_{2j}})(Z_2)) && (14)\\
Z_2^* &:= \mathrm{num}((x_j = \overline{R_{2j}})(Z_2^*)) && (15)\\
X_1^* &:= X_1^* \setminus \{x_j\} &&\\
Z_2^* &:= Z_2^* \cup \{A_1^*(x_j - \overline{R_{2j}})\} && (16)\\
A_1^* &:= A_1^* \setminus \{x_j = R_{1j}\} &&\\
\overline{E_2} &:= \mathrm{num}((x_j = \overline{R_{2j}})(\overline{E_2})) && (17)\\
\overline{A_2} &:= \overline{A_2} \cup \{x_j = \overline{R_{2j}}\} && (18)\\
\overline{F_2} &:= \overline{F_2} \setminus \{x_j\} && (19)\\
\overline{U_2} &:= \mathrm{num}((x_j = \overline{R_{2j}})(\overline{U_2})) && (20)
\end{aligned}$$

Comments:

∘ The substitution $x_j = \overline{R_{2j}}$ in Z_2 as shown in line (14) had already been computed when searching for an x_j giving a regular substitution. (14) is just assigning the computed value to Z_2.

∘ By doing substitutions in Z_2^* in line (15) it may happen that expressions become factorizable. In that case we drop all factors which do not contain the single x_k which was on the left hand side of the original assignment $x_k = R_{2k}$ in A_2^* which became the original $z_{2k}^* = x_k - R_{2k}$.

∘ In (16) x_j is substituted by the value it has in solution S_1. The new difference $R_{1j} - \overline{R_{2j}}$ is another expressions added to Z_2^* which must vanish modulo E_1 for merger.

∘ If finally all equations $0 = z_{2j}^*$ are solved then lines (17),(18),(19),(20) will provide the complete list of remaining unsolved equations $\overline{E_2}$, assignments $\overline{A_2}$, free or unsolved variables $\overline{F_2}$ and inequalities $\overline{U_2}$.

• If the execution reaches this point, solution S_2 has been successfully re-parametrized to $\overline{S_2}$. To include solution S_1 it has to check that the remaining substitutions A_1^* (which make S_1 more special than S_2) do not generate singularities: *Test whether substitutions* $Z_2 := A_1^*(Z_2)$ *are regular. Otherwise no merger is found.*

• *Final test: Merger is possible iff* $num(Z_2) \subset I(E_1)$. *In that case from* $\overline{U_2}$ *those expressions* u_2^* *are dropped for which* $num(A_1(u_2^*)) \subset I(E_1)$.

4. CONCLUSIONS

In this paper we discuss a method for merging different branches of solutions of polynomial algebraic systems. To reach this aim the method considers dropping inequalities that are associated with solutions, or to re-parametrize solutions, i.e. to make unknowns that are assigned a value to parameters and on the other hand to assign parameters a value. Although the procedure as it is described is not rigorous in finding under guarantee all mergers, it so far was fully successful in applications described in appendix A and in [2]. A modification to make the procedure complete is outlined. The method seems to be most useful for compactifying and simplifying general solutions that come in many branches, generated by a program that needs maximal flexibility in introducing case distinctions when solving large systems of polynomial equations.

5. ACKNOWLEDGMENTS

The author wants to thank Winfried Neun for many discussions. This research was supported by the Natural Sciences and Engineering Research Council of Canada, Grant 249783.

6. REFERENCES

[1] Sit, William, Y.: An Algorithm for Solving Parametric Linear Systems, J. Symb. Comput. (1992) **13**, 353–394.

[2] Wolf, T.: Applications of CRACK in the Classification of Integrable Systems, preprint arXiv nlin.SI/0301032, accepted for publlication in the CRM Proceedings, 15 pages (2002).

[3] Wolf, T., Efimovskaya, O. V.: Classification of integrable quadratic Hamiltonians on e(3), Regular and Chaotic Dynamics, **8**, no 2 (2003), p 155-162, and arXiv nlin.SI/0302001.

[4] Sokolov V.V.: A new integrable case for the Kirchhoff equation, *Teoret. and Mat. Fiz.*, (2001) **129**, no.**1**, 31–37, translation in *Theoret. and Math. Phys.*, (2001) **129**, no.**1**, 1335–1340.

APPENDIX

A. A NON-TRIVIAL APPLICATION

The package CRACK has been used for a variety of classifications of integrable systems (vector evolutionary partial differential equations (PDEs), 2+1 dimensional integro-differential PDE, Hamiltonian systems, non-linear 3$^{\text{rd}}$ order ODE, ... see [2]). The harder problems become the more the program must rely on introducing case distinctions and the more solutions have to be merged afterwards. In a recent investigation [3] the merging algorithm was especially useful, merging 90% of the solutions. The aim was to find quadratic Hamiltonians of the form

$$H = (M, AM) + (M, B\gamma) + (\gamma, C\gamma) + (P, M) + (Q, \gamma) \quad (21)$$

Table 1: An overview of the solved algebraic systems

Lie algebra:	$e(3)$	$e(3), J_2 = 0$	$e(3)$ quantum
# of unknowns (H,I,total)	17,200,217	17,176,193	17,200,217
# of equations	451	396	451
total # of terms	5469	5243	9681
average # of terms/equ.	12.1	13.2	21.5
time to solve	18h 53min	$\approx$ 15h	11h 43min
solutions before merging	61	68	53
solutions after merging	6	6	6

where $M = (M_1, M_2, M_3)$, $\gamma = (\gamma_1, \gamma_2, \gamma_3)$ are the dynamic variables, A, C are constant symmetric matrices, B is a general constant matrix and P, Q are constant vectors, such that this Hamiltonian admits polynomial first integrals of 3^{rd} or 4^{th} degree. The equations of motion are given by

$$\frac{dM_i}{dt} = \{H, M_i\}, \quad \frac{d\gamma_i}{dt} = \{H, \gamma_i\}$$

where Poisson brackets for any two functions F, G of the variables $Y_i = (M_1, M_2, M_3, \gamma_1, \gamma_2, \gamma_3)$ are defined by

$$\{F, G\} = \sum_{i,j} \left(\frac{\partial F}{\partial Y_i}\frac{\partial G}{\partial Y_j} - \frac{\partial F}{\partial Y_j}\frac{\partial G}{\partial Y_i} \right) \{Y_i, Y_j\}, \qquad (22)$$

$$\begin{aligned} \{M_i, M_j\} &= \sum_k \varepsilon_{ijk}\, M_k, \\ \{M_i, \gamma_j\} &= \sum_k \varepsilon_{ijk}\, \gamma_k, \qquad (23) \\ \{\gamma_i, \gamma_j\} &= 0 \end{aligned}$$

with ε_{ijk} being the totally anti-symmetric tensor. These linear Poisson brackets are related to the Lie algebra $e(3)$. Hamiltonians (21) with Poisson brackets (23) describe the motion of a rigid body around a fixed point under gravity (Euler-Poinsot model) and the motion of a rigid body in ideal fluid (Kirchhoff model). For the first integral I (not to be confused with the ideal I from previous sections) an ansatz for a general 4^{th} degree polynomial in the dynamical variables M_i, γ_j with constant unknown coefficients is made. For I to be a first integral the Poisson bracket $\{H, I\}$, as defined above, has to vanish identically in the M_i and γ_j. The set of resulting algebraic conditions is linear in the components of A, B, C, P, Q in (21) and linear in the 200 coefficients of the ansatz for I. But as each term of the conditions has two factors, one from each set, the problem is non-linear (bi-linear). In our approach we assumed without loss of generality the matrix A to be diagonal

$$A = \text{diag}(a_1, a_2, a_3)$$

and the matrix B to be low-triangular, i.e.

$$b_{12} = b_{13} = b_{23} = 0.$$

The two first integrals (so-called Casimirs)

$$J_1 = \gamma_1^2 + \gamma_2^2 + \gamma_3^2, \quad J_2 = \gamma_1 M_1 + \gamma_2 M_2 + \gamma_3 M_3 \qquad (24)$$

are used to simplify H and I. As a restriction we assumed A to have two identical eigenvalues, i.e. $a_1 = a_2$ which allows in addition to gauge $b_{21} = 0$.

In a second investigation (column 2 of table 1) we used the same ansatz but restricted the dynamical variables to obey $J_2 = 0$ with J_2 from (24) (a special motion in a plane of symmetry leading to *less* restrictive and therefore harder algebraic systems) and finally we run a quantum version (column 3) where M_i, γ_j are non-commuting operators and the Poisson bracket (22) is replaced by the commutator $[F, G] = FG - GF$ with commutator rules for M_i, γ_j like in (23).

Our complete classification revealed new (complex) Hamiltonians and a generalization of known classes of Hamiltonians of Kowalewski and Sokolov [4]. Table 1 gives an indication of the size of computations. Times are measured on a 1.7GHz Pentium 4 running in a 120 MByte REDUCE session under Linux. The first and third case had been solved automatically whereas the second case needed some human interaction. On the web page
`http://lie.math.brocku.ca/twolf/bl/e3/over.html`
the conditions, the unknowns and assumed inequalities are shown as well as the solutions before and after merging. The task to merge these solutions could not be done by hand as each solution contains expressions for about 200 unknowns, different solutions may be parametrized differently and with typically 60 solutions in each of the 3 investigations hundreds of merging attempts had to be done to obtain finally only 6 different solutions in each case. Merging the solutions took on average 9min in each case.