The Combinatorial Classes of Parallel Manipulators[*]

J.C. Faugère[†] and D. Lazard[‡]
LITP, Institut Blaise Pascal
Case 168, 4, place Jussieu
F–75252 Paris Cedex 05

February 16, 1994

Abstract

The classes of parallel manipulators are listed, which are defined by the constraint that some legs have the same fixation on the platform or on the base. For each class, the generic number of assembly configurations is given. This number never changes if the constraint of planarity is added.

Gröbner basis computation are widely used for obtaining these results. It appears that this method is now an efficient tool for solving polynomial systems which arise in robot kinematics.

1 Introduction

A *parallel manipulator* is a body (*platform*), the spatial position of which is commanded by fixing the distance of 6 points of the platform to 6 fixed points of the space (the *base*).

There may be various practical ways to realize such a mechanism, the most usual one being to use six linear actuators connected by spherical joints to the platform and to the base.

When one wants to compute the position of the platform from its geometry and the lengths of the linear actuators, one is led to a system of algebraic equations which has several solutions. The number of these solutions strongly depends on the geometry of the manipulator. It has been proved recently that, if this number is finite, it is at most 40 [Laz93, RV].

If the geometry of the manipulator satisfies some constraints, this maximal number may decrease. If these constraints may be represented by polynomial equations, the maximal number of *complex* solutions is reached for almost all manipulators satisfying them. There are numerous papers whose aim is to determine this maximal number for some classes of manipulators.

It is clearly not possible to enumerate all possible constraints. Thus, we limit ourselves to two kind of constraints:

[*]Work supported by EEC projects POSSO and PROMOTION

[†]e-mail: jcf@posso.ibp.fr

[‡]e-mail: dl@posso.ibp.fr

- The *combinatorial* constraints which consist in asking that some of the spherical joints are the same for two or more of the linear actuators.
- The *planarity* constraints which prescribe that all the spherical joints on the platform or on the base are in the same plane.

In this paper, we call *combinatorial class* of parallel manipulators the set of all manipulators satisfying some given set of combinatorial constraints. We enumerate all these classes, and for each of them we give the maximal number of assembly positions.

Moreover, we show that this maximal number is never changed when planarity constraints are added.

We want to acknowledge Jean-Pierre Merlet for many helpful discussion and also in providing a very complete bibliography on the subject [Merl].

2 The combinatorial classes

Usually, the parallel manipulators are classified by the number of different spherical joints on the base and on the platform. For example the most general manipulator is the 6-6 one. This classification is not discriminating enough for our purpose.

Thus, we represent each combinatorial class by a graph, the edges of it being the actuators, the upper (resp. lower) vertices being the joints on the platform (resp. on the base). If the graph has several isomorphic components, they are represented once, with the number of their occurrences as exponent. This abstract description becomes clear when looking on figure 1.

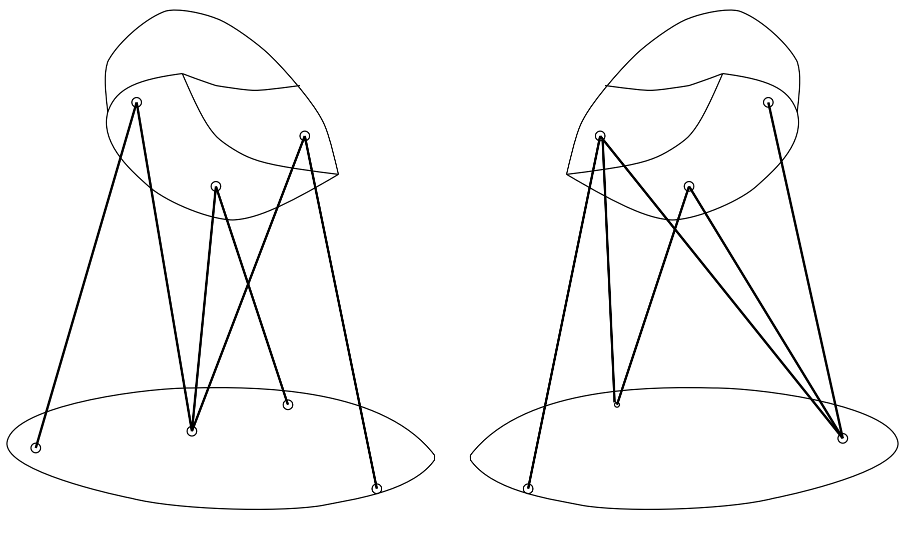

Figure 1: Platforms of classes (/\\|X\\) and (/|X|/)

In the list which follows, we have omitted the classes which have clearly an infinite number of assembly configurations, namely those

- which have only two joints on the platform or on the base;

- which have two actuators wit both their ends in common (the position of the platform depends on 5 conditions only, and one degree of freedom remains);
- which have 4 actuators which have one end in common (if 3 actuators have one end in common, this fixes the position of this end relatively to the base and to the platform; thus the fourth actuator is superfluous, and the position of the platform depends on 5 conditions only).

We may now list the combinatorial classes:

There is only one 6-6 combinatorial class: (|^6).

There is one 6-5 class: (/\ |^4).

There are two 6-4 classes : (/|\ |^3) and (/\$^2$ |^2).

There are two 6-3 classes: (/\$^3$)and (/|\ /\ |).

There are two 5-5 classes, both invariant by exchanging the platform and the base: (/\/ |^3) and (/\ \/ |^2).

There are five 5-4 classes: (/\/\ |^2), (/|\/ |^2), (/|\ \/ |), (/\/ /\ |) and (/\$^2$ \/).

There are four 5-3 classes: (/|\/\ |), (/\/\ /\), (/|\ /\/) and (/|\/ /\).

There are five 4-4 classes which are invariant by exchanging the platform and the base: (/\/\/ |), (/|\|/ |), (/|\ \|/), (/\/^{2}) and (|X| |^2).

There are three pairs of 4-4 classes which are exchanged by exchanging the platform and the base: (/\/\ \/), (\/|\/ |) and (/|\/ \/), and the ones obtained by symmetry with respect to a horizontal line.

There are seven 4-3 classes: (|X|\ |), (/|\/\/), (\/|\/\), (/\/\/\), (/|\|/\), (|X| /\) and (/\|X|\).

There are two 3-3 classes which are invariant by exchanging the platform and the base: (/|X|/) and (|X|X|).

There is one pair of 3-3 classes which are exchanged by exchanging the platform and the base: (|X|\/) and its symmetric (|X|/\) with respect to an horizontal line.

We can sum up this list in following result.

Proposition 1 *If the above listed classes with infinitely many assembly configurations are excluded, there are* 60 *combinatorial classes of parallel manipulators.*

If we identify two classes which are deduced one from the other by exchanging the platform and the base, the number of classes reduces to 35.

Remark 1 When we count the number of assembly positions, we count the number of relative positions of the platform and the base, and this number does not change if we exchange them. This is the motivation of the identification which appears in second part of Proposition 1. For this reason also, we have only listed the classes which do not have more joints on the platform than on the base.

3 Genericity and specialization

A *configuration* of a parallel manipulator is defined by fixing the parameters defining the geometry of the manipulator. These parameters may be

- the coordinates of the spherical joints on the base,
- the coordinates of the spherical joints on the platform, relatively to a frame linked to the platform,
- the 6 lengths of linear actuators

A configuration is said *generic* for a class if the values of the parameters do not satisfy any other algebraic relation than those which may be deduced from the definition of the class. As we are only considering classes of configurations which are defined by polynomial equations, it follows that almost all real choices for the parameters is generic.

However, we are not able to compute efficiently with irrational numbers, and generic configurations are not of practical use. Therefore we introduce the notion of *pseudo generic* configurations: a configuration is *pseudo generic* if the system of equations which has the assembly positions as solutions has the same number of *complex* solutions as for a generic configuration. The usefulness of these notions lies in the following result.

Proposition 2 *All the generic configurations of a class have the same number of assembly positions.*

If a configuration of a parallel manipulator has only a finite number of complex assembly positions, this number is not greater than the number of assembly positions of the generic configurations of this class.

The set of pseudo generic configurations for a class of parallel manipulators contains a dense open subset of the set of all configurations of the class.

Proof. The first assertion is a basic classical result of algebraic geometry.

For computing the assembly positions of a manipulator a set of equations may be written, which depends on some unknowns (variables defining the position of the platform) and of some parameters (variables defining the geometry of the configuration and the lengths of the legs). It appears that the number n of the equations is equal to the number of the unknowns (see [Laz92] for example, where $n = 9$). It follows that in the space of the unknowns *and* the parameters, these equations define an algebraic set V of co-dimension at most n and of dimension at least equal to the number of parameters.

If, for some point P in the space of the parameters, the number of solutions is finite, V cuts the linear space defined by P in a zero dimensional set. Above inequality on the dimension of V implies that the solutions at P remain existing in a whole neighbourhood of P.

This means that the number of solutions (assembly positions) may only increase in a neighbourhood of P. As any open set in the space of parameters contains generic configurations, second assertion follows.

The number of solutions may increase only if some multiple solution splits in several ones or if some solution at infinity becomes finite. As multiple solutions and solutions at infinity may be defined by algebraic conditions, the set of pseudo generic solutions contains the complement of an algebraic set, i.e. is a dense open set. ■

Corollary 1 *Almost all configurations are pseudo generic. This means that a random choice of the parameters has, with probability* 1, *a maximal number of complex assembly positions.*

Definition 1 *The* generic number of assembly positions *of a class of configurations is the number of assembly positions of any pseudo generic configuration in it.*

A class of manipulators is a specialization *of another one if it is obtained by adding some constraints. A combinatorial class is* maximal *(resp.* minimal*) if it is not the specialization of a class having (resp. if no specialization of it has) the same generic number of assembly positions. This is represented by an arrow from the less specialized class to the most specialized one.*

4 The generic numbers of assembly configurations

In this section, we list for all combinatorial classes the assembly number of configurations. When it is easy, we give a short proof of this number. Most of these proofs are not new; nevertheless we give them without references, because it is difficult to decide who gave them first.

In fact, we only give here an upper bound of these numbers, and we leave for next section the proof that these numbers are the exact values.

4.1 The 10 specializations of (/|\ /\ |)

Proposition 3 *The 10 specializations of* (/|\ /\ |)*, namely* (/|\ /\ |) *itself,* (/|\/\ |), (/|\/ /\), (/|\ /\/), (/|\/\/), (/|\|/\), (\/|\/\), (|X|\ |), (/|X|/) *and* (|X|\/) *have generically 8 assembly configurations.*

Clearly, the triple link on the platform may only have two positions. When one of them is chosen, the double link should be at a fixed distance from the triple one and from two points of the base, and there are two positions for it. Finally, the simple link should be at a fixed distance from the two other links of the platform and from one point of the base; again two choices. In the whole, we got 8 choices.

For proving that this bound is attained, it suffice to show it for the two minimal classes, namely (/|X|/) and (|X|\/).

4.2 The 6 remaining specializations of (/\$^3$)

Proposition 4 *Five specializations of* (/\$^3$) *have generically 16 assembly configurations, namely* (/\$^3$) $\rightarrow$ (/\/\ /\) $\rightarrow$ (/\/\/\) $\rightarrow$ (|X|X|) *and* (/\|X\)*.*

The generic number of assembly configurations is 8 for (|X| /\)*, as well as for its specialization* (|X|/\) *which is equivalent (by exchange of the platform and the base) to the above considered* (|X|\/)*.*

Each point of the platform has to be at a fixed distance from two points of the base and thus lies on a circle. Let us consider the parametrization of these three circles by the tangent of the half angle between a fixed and a variable radius. The distance between the links on the platform is fixed, and this gives three equations which (after removing denominators) are quadratic in each of two among the parameters and which do not depend on the third parameter. Thus, multi-homogeneous Bezout theorem [Shaf] shows the upper bound of 16 for the generic number of assembly positions of all these classes.

For bounding the number of assembly positions of (|X| /\), let us denote by M_1, M_2 and M_3 the three points of the platform, in the order in which they appear on the drawing representing the class, and F_1, F_2 and F_3 the corresponding points on the base. The lengths of the linear actuators fixed at M_3 assign it to lie on a circle; we will see that the distances from M_3 to M_1 and M_2 and the lengths of the four other links assign M_3 to lie also on one of two surfaces of degree 4 and of circularity 2. Each of these surfaces has 4 intersections at infinity which above circle and 4 other intersections, which gives a bound of 8 for the number of solutions of the problem.

The linear actuators fixed at M_1 and M_2 assign them to lie on two coaxial circles. If one fixes the position of M_1, the distance from it to M_2 assign it to lie on the intersection of a sphere and a circle. Let us choose one of these two intersection points. The distances from M_1 and M_2 to M_3 assign it to lie on a circle with M_1M_2 as axis. When M_1 varies, all the figure rotates around the axis F_1F_2; thus M_3 lies on a surface of revolution generated by a circle. Such a surface is a kind of torus, of degree 4 and of circularity 2, which may differ from the usual torus by the fact that its intersection with a plane passing through the axis is an irreducible curve of degree 4 instead of being two circles. All these facts may easily be proved by hand or by using any computer algebra system.

As for the previous set of classes, we have to prove that these upper bounds are exact; for this, only (|XX|) and (/\|X\) need to be considered, because they are minimal for this set of classes and because (|X| /\) has a specialization with is already known to have a generic number of assembly positions of 8.

4.3 The 7 remaining specializations of (/|\ |^3)

Proposition 5 *Among the 7 specialization of* (/|\ |^3) *which do not appear in preceding sections*

- *one has an infinite number of assembly positions:* (/|\ \|/),
- *three have generically 16 assembly positions:* (/|\ |^3) $\rightarrow$ (/|V |^2) $\rightarrow$ (V|V |),
- *three have generically 8 assembly positions:* (/|\ V |) $\rightarrow$ (/|V/ |) *and* (/|V V).

If one has an assembly configuration for (/|\ \|/), infinitely many of them may be obtained by rotating the figure around the line passing through the two triple joints. Thus we need to consider only the 6 remaining classes.

When the lengths of the linear actuators connected to the triple joint of the platform are fixed, the latter may have 2 localizations. When one is chosen, the 3 other joints of the platform have to be at a fixed distance from the triple joint and from one point of the base; thus we have a problem of the class (/\|X\) or (|X|/\), depending on the class under consideration. Among the solutions of this problem, some of them have the triple joint in the right place, and the others have this joint in the symmetric localization with respect to the plane defined by the 3 simple joints on the platform. Generically, the number of acceptable solutions is equal to the number of the not acceptable ones: replacing the platform by its mirror image exchanges the acceptable and the not acceptable solutions, and there are clearly pseudo generic instances of the problem such that the new problem defined by such a replacement is also pseudo generic.

Thus the problem in consideration has the same generic number of solutions as (/\|X|\) or as (|X|/\).

Here again, we have only proved an upper bound. But the classes considered here specialize to (/\|X|\) (after exchanging the platform and the base) or to (/|\/\/), which have the same generic number of assembly positions. Thus there is no need to provide here explicit examples.

4.4 The 12 remaining classes

Fact 1 *The generic number of assembly positions for the twelve remaining classes is:*

- *40 for* (|^6) → (/\ |^4) → (/\/ |^3)
- *32 for* (/\$^2$ |^2) → (/\/\ |^2)
- *24 for* (/\ \/ |^2) → (/\/ /\ |) → (/\/\/ |)
- *16 for* (/\$^2$ \/) → (/\/\ \/), (|X| |^2) *and* (/\/^{2})

As for the preceding classes we have to prove that the given value is an upper bound for the classes with no arrow on their left and to provide examples for the classes with no arrow on their right (except for the last class which specializes to (/\/\/\)).

For (|^6), the correct upper bound has been proved in [Laz93, RV, Mour]. For (/\ \/ |^2), this seems to have been done in [HP], but their proof applies also for obtaining a false value of 24 for (/\$^2$ |^2). For (/\/^{2}) a proof has been given in [LDG], but only when the platform and the base are planar; the generalization to the non planar case asserted in [HP] seems to be more difficult than claimed. For (/\$^2$ \/), the number of 16 appears in [HP] also. The two other maximal classes do not seem to have been studied successfully before us.

It follows from this short bibliographical review that, for these classes, the *mathematical* proofs are difficult, not yet available for all cases and may easily contain false assertions.

We have obtained these values by computing the number of assembly positions for randomly chosen configurations in the classes. The probability of obtaining a false value, i.e. a configuration which is not pseudo generic would be zero, if the random choices would be taken among all real numbers. As we only chose integers in the range $[-90..90]$ the probability of failure remains very low, but is not exactly 0: it is the probability that a random point in a space of dimension at least 18 lies on a specified hyper-surface. For speeding up the computation, we have sometimes added another cause of failure: We have done the computation modulo 31991; thus a false result could arise if some leading coefficient of the polynomials which appear during the computation would have 31991 as divisor.

We have done these modular computations at least twice for the maximal classes and once for the other classes.

Moreover, we have done the computation over the integers (i.e. without the risk of failure due to the modular computation) for all minimal classes, with planar platform and base.

For each of the 6 maximal classes for which no proof is given here, we have also computed, for one example, the Gröbner basis for the degree ordering.

The low probability of failure appears in the fact that all computations gave coherent values for the number of assembly configurations (except once where we did a mistake in specifying the class!).

Thus the values that we have given are not proved in the mathematical meaning of the word, but they are much more a certainty than if they were obtained as the result of a complicated mathematical proof.

5 Computations

For solving systems of algebraic equations, we use Gröbner bases. Let us recall very briefly what it is, leaving details and precise definitions to [Laz91, BCK].

Given a set of polynomials, a Gröbner basis is another set of polynomials which has the same common roots in a very strong sense (the multiplicities are the same, the generated ideal is the same). A Gröbner basis may be viewed as a compiled form for a system of equations in the sense that no information is lost and, on the opposite, many properties of the solutions, such as their number or their values may easily be deduced from the Gröbner basis.

A Gröbner basis is a canonical form for a system of equations which only depends on the input equations and on a total ordering on the monomials (power products). Two orderings are especially important, the degree–reverse–lexicographical one, which leads to rather easy computations and the purely lexicographical one for which the Gröbner basis is more difficult to compute, but for which the information is more accessible: For all computations that we have done for this paper, the Gröbner basis for the lexicographical ordering has the form

$$\begin{array}{ll} f_9(x_9) & \\ x_i - f_i(x_9) & \quad \text{for } i = 1, \ldots, 8 \end{array}$$

where f_9 is a polynomial with integer coefficients with the number of solutions as degree, and the f_i for $i < 9$ are polynomials of degree lower than the degree of f_9, with rational coefficients.

Fortunately an efficient algorithm is known for getting the Gröbner basis for the lexicographical ordering from another one [FGLM], and we have used it.

We have used Maple to generate the system of equations to be solved; our Maple program appears in Figure 2. The unknowns `x1, y1, z1, x2, y2,`

```
########################### spatial case NI3, 5-5 ##################
random:=rand(-90..90):
for i from 1 to 9 do d.i:= random()^2 od:
for i from 4 to 6 do e.i:=random(); f.i:=random(); g.i:=random() od:
for i to 6 do a.i:=random(); b.i:=random(); c.i:=random() od:
a1:=0: b1:=0: c1:=0: b2:=0: c2:=0: c3:=0:
vx:=(y2-y1)*(z3-z1)-(y3-y1)*(z2-z1):
vy:=(z2-z1)*(x3-x1)-(z3-z1)*(x2-x1):
vz:=(x2-x1)*(y3-y1)-(x3-x1)*(y2-y1):
e6:=0: f6:=0: g6:=0:                    # M6=M1 ###These 2 lines
a6:=a2: b6:=b2: c6:=c2:                 # A6=A2 ###specify the class
for i from 4 to 6 do
        x.i:= x1+e.i*(x2-x1)+f.i*(x3-x1)+g.i*vx;
        y.i:= y1+e.i*(y2-y1)+f.i*(y3-y1)+g.i*vy;
        z.i:= z1+e.i*(z2-z1)+f.i*(z3-z1)+g.i*vz
od:
for i from 1 to 6 do
        C.i:=(x.i-a.i)^2+(y.i-b.i)^2+(z.i-c.i)^2-d.i
od:
C7:=(x2-x1)^2+(y2-y1)^2+(z2-z1)^2-d7:
C8:=(x3-x1)^2+(y3-y1)^2+(z3-z1)^2-d8:
C9:=(x3-x1)*(x2-x1)+(y3-y1)*(y2-y1)+(z3-z1)*(z2-z1)-d9:
for i from 4 to 6 do
    C.i:=C.i-e.i^2*C7-f.i^2*C8-g.i^2*(C7*C8-C9^2)-e.i*f.i*(2*C9):
od:
C7:=-C7+C1+C2:   C8:=-C8+C1+C3:   C9:=2*C9+C7+C8-2*C1:
for i from 4 to 6 do
  C.i:= 2*C.i-e.i*(C7-2*C1)-f.i*(C8-2*C1):
  C.i:=C.i-2*g.i^2*d7*(C1+C3-C8)-2*g.i^2*d8*(C1+C2-C7)
       -2*g.i^2*d9*(C7-C9+C8-2*C1)+2*(e.i+f.i-1)*C1-C7*e.i-C8*f.i:
od:
L:=map(u->sort(expand(u)),[C1,C2,C3,C7,C8,C9,C4,C5,C6]);
lprint(L);
```

Figure 2: Program for generating equations

`z2, x3, y3, z3` are the coordinates of 3 points of the platform. The randomly chosen parameters are the square of the lengths of the linear actuators (`d1` to `d6`), the square of the distances from the first joint of the platform to the second and the third ones (`d7` and `d8`), the dot product of the vectors defined by these points (`d9`), the coordinates of the 3 last joints of the platform in the frame defined by the 3 first ones (`e.i, f.i, g.i`) and the coordinates of the points of the base (`a.i, b.i, c.i`). Some simplification of the equations are done by choosing the origin, the axis and by replacing some of the equations by linear combinations of them.

For the planar case, the `c.i` and the `g.i` are set to 0.

We have used the equations obtained by this Maple program in the Gröbner basis package developed by the first author and called Gb, which seems to be the most efficient Gröbner basis package presently available. This package may

```
vars:=[x1, x2, x3, y1, y2, y3, z1, z2, z3];
HD:=HDMP(vars,PF(31991));
l:List(HD):=[x1**2+y1**2+z1**2-576, x2**2+y2**2+z2**2-94*x2-4191, _
x3**2+y3**2+z3**2-30*x3+174*y3+4985, 2*x1*x2+2*y2*y1+2*z2*z1-94*x2 _
-4667, 2*x1*x3+2*y1*y3+2*z1*z3-30*x3+174*y3+4938, 2*x2*x3+2*y2*y3+2* _
z2*z3-94*x2-30*x3+174*y3+1261, 180*x1*y2*z3-180*x1*y3*z2-180*x2*y1* _
z3+180*x2*y3*z1-180*x3*y2*z1+180*x3*y1*z2-540*x1*y2+540*x1*y3-5400* _
x1*z2+5400*x1*z3+540*x2*y1-540*x2*y3+5400*x2*z1-5400*x2*z3+540*x3*y2 _
-540*x3*y1+5400*x3*z2-5400*x3*z1-8820*y2*z1+8820*y2*z3+8820*y1*z2- _
8820*y1*z3-8820*y3*z2+8820*y3*z1-13524*x1+16128*x2+7168*x3+5040*y2 _
-8280*y1-6384*y3-504*z2+828*z1-336*z3+189188716, -16*x1*y2*z3+16*x1 _
*y3*z2+16*x2*y1*z3-16*x2*y3*z1+16*x3*y2*z1-16*x3*y1*z2+320*x1*y2-320 _
*x1*y3-1168*x1*z2+1168*x1*z3-320*x2*y1+320*x2*y3+1168*x2*z1-1168*x2* _
z3-320*x3*y2+320*x3*y1+1168*x3*z2-1168*x3*z1+1392*y2*z1-1392*y2*z3- _
1392*y1*z2+1392*y1*z3+1392*y3*z2-1392*y3*z1+15312*x1+17152*x2-30600 _
*x3-9344*y2-12848*y1+48000*y3-2560*z2-3520*z1+6000*z3+8012714,  _
-188*x1-10272]
)time on
)axiom on
g:=groebner(l);
totolex(g);
```

Figure 3: Input for Gb

be obtained by anonymous ftp from `posso.ibp.fr`. It is fully interfaced with Axiom computer algebra system and uses its syntax. The Gb program which appears in Figure 3 computes the Gröbner basis for the degree–reverse–lexicographical ordering (function `groebner`) and then deduces from it the Gröbner basis for the lexicographical ordering function `totolex`). This is done with modular coefficients; for integer coefficients, it suffices to replace `PF(31991)` (which denotes the prime field with 31991 elements) by `INT`.

The output appears in Figure 4. One can see on this figure how a Gb session works. The lines which appear between `utilisation du serveur rapide` and the time needed for the computation are messages informing the user of the state of the computation. After the command `g:=groebner(l);` one can see an extension of the memory with the time of this extension. Then several lines of `*` and `+` appear; a `*` or a `+` means that a new polynomial has been computed which reduces to 0 (`*`) or not (`+`); numbers like `[2]` mean the degree in which the computation is done; the powers of variables shows that a polynomial with a leading monomial depending only on one variable has been found (the name of the variable is a dummy one because the variables names are not known during the computation); finally `|40|` means that enough information has been got in order to know that the number of solutions is finite and is at most 40. This last information appears at several places and not only in the final univariate polynomial.

As it is clearly impossible and useless to give all computational results, we only give the times of computation in Tables 1, 2 and 3. All computations have

```
dl@posso->Gb
Grobner Basis Computations (C++ version) , Version SCCS: 3.%I%  %W%  %G%(761)
Warning: this is not the definitive version !
Author:Jean-Charles Faugere ***  Universite Paris 6/LITP
Bug Report:    jcf@posso.ibp.fr
Connection established with X.exe on posso.ibp.fr
Working directory /users/dl/Articles/Stewart/Calculs2
GB> )r spatnN3I-mod.gb
; vars:=[x1, x2, x3, y1, y2, y3, z1, z2, z3];
; HD:=HDMP(vars,PF(31991));
;l:List(HD):=[x1**2+y1**2+z1**2-576, x2**2+y2**2+z2**2-94*x2-4191, _
                      ... 13 lines removed  ...
-188*x1-10272]
           2     2     2           2     2     2
       [x1  + y1  + z1  + 31415,x2  + y2  + z2  + 31897x2 + 27800
                        ... lines removed  ...
+ 16009y3 + 28471z1 + 29431z2 + 6000z3 + 14964,31803x1 + 21719]
        Type: List(GDMP([x1,x2,x3,y1,y2,y3,z1,z2,z3],PF(31991),DEGREVLEX(9)))
; )time on
; )axiom on
; g:=groebner(l);
Connection established with serveur__Dexp__GF on posso.ibp.fr
Utilisation du serveur rapide
*** Extend Memory (DMP__Dexp__GF) Tue Oct 19 10:17:04 1 alloc:1200000 octets
[2]+++|x3^2|[3]+++++++**[4]++++++++++++++++++++++**[5]++++++|x4^3|+|x6^3|+
+|x5^3|++++++++++++++++++++++|x4^2|+++*+|x5^2|****************************
*[6]++++++++*+|x7^3|*************************++++++++++++++++++************
*************************************[7]***********************************
***************************************************************+|x8^5||40|
******************************************[8]**************************[9]*
End of gbasis computation !
End of minimal gbasis computation !
                Serveur: 32.38 sec Eval: 0.28 sec Total: 32.67 sec
<base=[z3**5 + 17722*z3**4 + 30856*z1*z2*z3 + 25425*z2**2*z3 + 14694*x2*z3**2
                        ... many lines removed ...
2648*z2 + 28048*z3 + 4581,x1 + 29323],nb solutions= 40,
        serveur=|Serveur: serveur__Dexp__GF|>
; totolex(g);
Utilisation du serveur rapide
*** Extend Memory (DMP__Lexp__GF) Tue Oct 19 10:17:42 1 alloc:1200000 octets
(40)[8][7][6][5][4][3][2][1][0]
                Serveur: 2.00 sec Eval: 0.03 sec Total: 2.03 sec
[x1 + 29323,x2 + 29983*z3**39 + 12264*z3**38 + 2436*z3**37 + 22599*z3**36 +
                                ...
,x3 + 8428*z3**39 + 5118*z3**38 + 1506*z3**37 + 20086*z3**36 + 17489*z3**35
                                ...
,z3**40 + 218*z3**39 + 10912*z3**38 + 20790*z3**37 + 12075*z3**36 +
                                ...
+ 23862*z3**2 + 26508*z3 + 8241]
GB> )q
     Total Time: Central GB: 3.68 sec + Serveurs: 34.38 sec
Il reste 9611856 octets disponibles
dl@posso->
```

Figure 4: Output of Gb

	#roots	time groebner	time totolex		#roots	time groebner	time totolex
$(\vert^6)$	40	60.1	2	$(\diagup\!\diagdown\!\diagup\ \wedge\ \vert)$	24	6.7	0.6
$(\wedge\ \vert^4)$	40	35.2	2	$(\diagup\!\diagdown\!\diagup\!\diagdown\!\diagup\ \vert)$	24	5.7	0.6
$(\diagup\!\diagdown\!\diagup\ \vert^3)$	40	32.4	2	$(\wedge^2\ \vee)$	16	4.1	0.2
$(\wedge^2\ \vert^2)$	32	12.4	1.1	$(\diagup\!\diagdown\!\diagup\!\diagdown\ \vee)$	16	3.5	0.2
$(\diagup\!\diagdown\!\diagup\!\diagdown\ \vert^2)$	32	10.1	1.2	$(\diagup\!\diagdown\!\diagup^2)$	16	3.5	0.2
$(\wedge\ \vee\ \vert^2)$	24	23.7	0.5	$(\vert\!\times\!\vert\ \vert^2)$	16	2	0.2

Table 1: Times for modular computations (in seconds)

been done on a SUN station Sparc 2.

Table 1 corresponds to modular computations for the last 12 classes of preceding section; as the first Gröbner basis computation already gives the number of solutions, the change of ordering (function totolex) for getting the base for the lexicographical ordering is not really useful here.

Table 2 corresponds to computations over the integers, with planar base and platform; here the change of ordering is useful in order to get an univariate polynomial, which may be used for numerically computing the solutions or for computing the number of real solutions; we have used it for proving that we have chosen a proper parametrization of the problem: It could be the case that the solutions appear systematically as multiple roots of the system; as we have verified that the univariate polynomial is always irreducible, we know that its degree is the real number of complex solutions.

The entries of this table correspond to all minimal classes in the list. Thus, these computations are sufficient for proving that the generic numbers of assembly configurations is never smaller than the given values. They also prove that the constraints of planarity do not change these numbers.

We give also in this table some values called the output time and size; in fact, it is the total time of input output operations and the size of the image of the session; for the difficult cases, they are rather high because of the size

	#roots	time groebner	time totolex	output time	output size (k-bytes)
$(\diagup\!\diagdown\!\diagup\ \vert^3)$	40	32	31335	2581	603
$(\diagup\!\diagdown\!\diagup\!\diagdown\ \vert^2)$	32	12	7129	826	309
$(\diagup\!\diagdown\!\diagup\!\diagdown\!\diagup\ \vert)$	24	9	1107	227	143
$(\diagup\!\diagdown\!\diagup\!\diagdown\ \vee)$	16	8	228	87	75
$(\vert\!\times\!\vert\ \vert^2)$	16	4	69	30	42
$(\wedge\!\vert\!\times\!\diagdown)$	16	1.5	17.3	8.5	24
$(\vert\!\times\!\times\!\vert)$	16	2.5	29.13	12	28
$(\diagup\!\vert\!\times\!\vert\!\diagup)$	8	0.3	0.5	1.3	5.2
$(\vert\!\times\!\vert\!\vee)$	8	0.6	1.2	2	7.3

Table 2: Computation over the integers (planar case)

	#roots	time groebner	total time tgroebner	time of computation	time of verification
$(\|^6)$	40	> 10 days[1]	15h8'7"	14h43'20"	24'47"
$(\bigwedge^2 \|^2)$	32	6h1'15"	25'51"	17'43"	8'08"
$(\bigwedge \bigvee \|^2)$	24	47h	3h3'5"	2h51'1"	12'4"
$(\bigwedge^2 \bigvee)$	16	44'3"	7'52"	3'29"	4'23"
$(/\backslash/^2)$	16	47'6"	5'32"	2'51"	2'41"
$(\|\times\| \; \|^2)$	16	4'20"	1'7"	20"	47"

Table 3: Computation over the integers (non planar case)

of the integers which appear in the lexicographical Gröbner basis (result of `totolex`) and of the time needed to convert them from binary to decimal. For $(/\backslash/ \, |^3)$, some of these integers have more than 4000 decimal digits; the univariate polynomial of degree 40 of this example contains coefficients with more than 300 decimal digits.

Finally, Table 3 gives the time needed for computing the Gröbner basis over the integers (for the degree–reverse–lexicographical ordering) for the 6 maximal classes for which no mathematical proof are given here. These computations confirm the values found by modular computations. They show also that the non planar cases are much more difficult than the planar ones.

For each class appearing in this Table 3, several times are given. The first one corresponds to the first version of the Gröbner basis algorithm which was used; it appears that some times were very long and that most of them were spent in computing polynomials which reduce to 0. Thus we implemented another strategy called "tgroebner", which will become the default of Gb. It consists in using the trace (see [Trav]) of the modular computation in order to avoid to compute zero (with the hope that no integer appearing during the computation is a multiple of 31991), and then to verify the result and, if it is not correct, to complete it (but this never occurs in practice). Thus we give the total time needed by this new algorithm, and we split it into the computational time and the verification time; the latter may sometimes be avoided (If an unproved result is acceptable, or if it may be verified in another way, for example by computing the reverse kinematic).

We have given both old and new times in order to show that dramatic improvements are rather likely in Gröbner basis computation.

For changing the ordering (the most critical step), one of us has found recently a new algorithm which is dramatically faster than `totolex`. We hope that the fully optimized implementation (in progress) will lead to times of computation of the same order as for computing the first base.

If it will be the case, each planar case will be solved in a time lower than a minute, and the most difficult non planar cases will need a few hours, or a few tens of hours for the most general configuration.

[1]Estimation, the computation has been stopped after 110h.

6 What is a proof?

Some of the results in this paper are called "theorem" or "proposition", and one of them is called "fact". It appears that the latter is not proved in the usual mathematical meaning, but it is much more than a conjecture: it is more a certainty than many theorems, which have a complicated proof of several tens, hundreds or thousands of pages which makes questionable the reliability of the proof, and thus of the result.

For these reasons, a discussion on the kinds of proofs which arise in this subject may be useful.

Formally, a proof is a sequence of elementary deduction steps (syllogisms). Such proofs are so long and so tedious that they are never written. In fact, this is generally not possible by hand and it is the main objective of the most modern theorem provers. As far as we know, this has only be done for a few of rather simple mathematical theories.

As formal proofs are not available, mathematician usually consider as a proof a sequence of arguments which appears to be convertible into a formal proof. As this is not practically feasible, this convertibility is mainly a subjective matter, and a proof is viewed as correct when several experimented mathematicians have *read* it and assert that it is correct. Recent comments on the proof of Fermat last theorem enforces this sociological definition of the correctness of a proof.

In this context, computer aided proofs may make some problems: It is usually not possible to read (and verify) all programs which are involved in a computation, like compilers, operating systems and general systems like Maple. Thus, one has to trust the validation process of the software packages of general use (this validation process includes all the applications already done), and, if possible, to do again the computation with another machine and different software. It remains to verify the special purpose programs. It is for this reason that an example of each of the programs which have been written specially for this paper appears in the figures.

Under these conditions, a computer aided proof which uses *exact* computations is much more secure than a pure hand made proof, the highest probability of failure of the former lying in the hand made part. Nevertheless, it is a question of taste to make a choice between these two kinds of proofs. The two first proofs of the bound of 40 for the general parallel manipulator [RV, Laz93] is a good example of this discussion. The lower bounds for the generic numbers of assembly positions which appear in this paper are other examples of computer aided proofs.

On the other hand, the upper bounds which appear in "Fact 1" have not been proved in the above meanings. They are deduced from several random choices, and it is theoretically possible that all of them fail in giving the same false value. However, the probability of such a failure is so low that it is much lower than the probability of the existence of a hidden bug in the proof of any new theorem.

Thus our "Fact 1" is not proved but it is more a certainty than most "proved" theorem. We do not know any word for such a *known* mathematical fact which is not proved in the usual meaning of this word.

Such results which are not proved have already appeared in the theory of parallel manipulators, but they usually involve floating point computation, for which roundoff errors introduce a possibility of failure of a rather high probability, difficult to estimate. This is the case of Raghavan paper [Rag] which first gave the generic number of 40 assembly positions for the general parallel manipulator by solving several random examples. The justification of his method is mainly our Corollary 1. However he uses continuation method for solving systems of equations, and such a method is difficult to manage in order to be certain not to omit some solutions.

In [IPC] another, more hidden, numerical evidence appears. They obtain a univariate polynomial of degree 24 as the determinant of a matrix, but this determinant it too big to be expanded with all parameters kept symbolic. Thus, they know that the degree is 24 by choosing randomly some floating point values for the parameters, by evaluating the determinant and by remarking that all coefficients of degree higher than 24 are small enough to be considered as 0 (oral answer to a question in the Dagstuhl conference where this paper were presented; in fact, this answer was about [Inn], but it seems that the answer applies to both papers). It follows that, if exact values (instead of floating point ones) would be taken the status of their proof would be exactly the same as for our Fact 1.

7 Conclusion and open problems

We have solved the problem of determining the number of assembly positions for all combinatorial classes of parallel manipulators, and we have shown that solving the corresponding system is now possible with general purpose solvers.

However, for the less specialized classes, this system remains difficult for the most efficient solvers (several hours of computation). It should be remarked that efficiency of solving algorithms increases rapidly, and we hope dramatic improvement in next few years, especially for the base change algorithms (totolex).

Another way for improving the solving process is to choose other systems of equations to be solved: different modelizations may lead to dramatically different computational times. It is an open problem to determine, for each combinatorial class, the best system to be solved.

Another open problem is to mathematically prove Fact 1, that is to prove that the value given there are upper bounds for the generic number of assembly positions of $(\bigwedge^2 |^2)$, $(\bigwedge \bigvee |^2)$, $(\bigwedge^2 \bigvee)$, $(|\!\times\!| \, |^2)$ and $(/\!\backslash\!/^2)$. The interest of such proofs is, following the discussion of preceding section, a question of taste.

One may consider other classes of configurations which are defined by geometrical constraints such alignment of some points or parallelism of some lines. Some of them have been already studied [Nair, LM], but it seems difficult to make a complete list of all such interesting classes. Nevertheless, it is easy with existing software to know which is the generic number of assembly positions of any such class.

The most interesting open problem in this area is, in our opinion, to determine, for each class, the maximal number of *real* assembly position. The difficulty lies here in the fact that random experimentation never gives examples where this maximum is reached: the corresponding open set in the space of parameters is usually too small for hoping that a random choice falls into it.

The results of Section 3 may easily be extended in order to show that the number of simple real assembly positions do not decrease in a neighbourhood in the parameter space, i.e. that the maximal number of assembly positions do not increases when one specializes the class. From this, the following result may easily deduced from known examples.

Proposition 6 *For all combinatorial classes there are planar configurations with at least 8 real assembly positions.*

For all classes with a generic number of assembly positions not lower than 16, there are planar configurations with at least 16 real assembly positions, except, may be, for $(\bigwedge^2 \bigvee)$, $(\bigwedge\!\!\!\bigwedge \bigvee)$ *and* $(\|\!\!\times\!\!\|\,|^2)$.

Thus the main problem is to exhibit examples with more than 16 real assembly positions.

References

[BCK] Buchberger B., Collins G.E. and Kutzler B. Algebraic methods for geometric reasoning. *Ann. Rev. Comput. Sci.*, **3**, 1988, 85–119.

[Shaf] Shafarevich I.R. *Basic Algebraic Geometry*, Springer (Berlin), 1974.

[FGLM] Faugère J.C., Gianni P., Lazard D. and Mora T. Efficient Computation of Zero–dimensional Gröbner Bases by Change of Ordering. *To appear in J. Symb. Comp.* Technical Report LITP 89–52 (1989).

[Sugar] Giovini A., Mora T., Niesi G., Robbiano L., Traverso C. "One sugar cube, please"; or: Selection strategies in Buchberger algorithm. *Proc. ISSAC '91*, ACM, 1991, 49–54

[HP] Hunt K.H. and Primrose E.J.F. Assembly configurations of some in-parallel actuated manipulators. *Mechanism and Machine Theory*, **28**, 1993, 31–42,

[Inn] Innocenti C. Analytical determination of the intersection of two coupler-point curves generated by two four-bar linkages. *Computational Kinematics*, J. Angeles, P. Kovacs and G. Hommel (eds), Kluwer, 1993, 251–262.

[IPC] Innocenti C. and Parenti–Castelli V. Direct kinematics in analytical form of a general geometry 5–4 fully parallel manipulator. *Computational Kinematics*, J. Angeles, P. Kovacs and G. Hommel (eds), Kluwer, 1993, 141–152.

[Laz91] Lazard D. Systems of algebraic equations (algorithms and complexity). In *Proc. of Cortona conference (1991)*, D. Eisenbud et R. Robbiano (eds) Cambridge Univ. Press, 1993.

[Laz92] Lazard D. Stewart Platforms and Gröbner Basis. *3rd International Workshop on Advances in Robot Kinematic (3ARK, Ferrare, Italy)*, Parenti-Castelli V. and Lenarčič J. (eds), V. Parenti-Castelli (Ferrare), 1992, 136–142.

[Laz93] Lazard D. On the representation of rigid–body motions and its application to generalized platform manipulators. *Computational Kinematics*, J. Angeles, P. Kovacs and G. Hommel (eds), Kluwer, 1993, 175–181.

[LM] Lazard D. and Merlet J-P. The (true) Stewart platform has 12 configurations. *submitted to IEEE Int. Conf. on Robotics and Automation*, San Diego, 1994.

[LDG] Lin W., Duffy J., and Griffis M. Forward displacement analysis of the 4-4 Stewart platform. *Journal of Mechanical Design*, **114**, 1992, 444–450.

[Merl] Merlet J.P. *Les robots parallèles.* Mémoire d'Habilitation, Université de Nice, 1993.

[Mour] Mourrain B. The 40 generic position of a parallel robot. *Proceedings of ISSSAC'93 (Kiew)*, M. Bronstein ed., ACM Press, 173-182.

[Nair] Nair P. *On the kinematics geometry of parallel robot manipulators.* Master's thesis, University of Maryland, College Park, 1992.

[Rag] Raghavan M. The Stewart platform of general geometry has 40 configurations. *ASME Design and Automation Conf.*, **32-2**, 1991, 397–402.

[RV] Ronga F. and Vust Th. *Stewart platforms without computer?*, Preprint, Université de Genève, 1992.

[Trav] Traverso C. Gröbner trace algorithms. *Proceedings of ISSAC 88, Roma.* Lect. Notes in Comp. Sc. **358**, 1988, 125–138.