Prologue: General Remarks on Computer Algebra Systems

Computer algebra algorithms allow us to compute in, and with, a multitude of mathematical structures. Accordingly, there is a large number of computer algebra systems suiting different needs, ranging from the general to the special purpose. Some well-known examples of the former are commercial, whereas many of the special purpose systems are open-source and can be downloaded from the internet for free. General purpose systems aim at providing basic functionality for a variety of different application areas. In addition to tools for symbolic computation, they usually offer tools for numerical computation and for visualization.

Example P.1 MAPLE is a commercial general purpose system. To show a few of its commands at work, we start with examples from calculus, namely definite and indefinite integration:

```
> int(sin(x), x = 0 .. Pi);
                          2
> int(x/(x^2-1), x);
             1/2 ln(x - 1) + 1/2 ln(x + 1)
```

For linear algebra applications, we first load the corresponding package. Then we demonstrate how to perform Gaussian elimination and to compute eigenvalues.

```
with(LinearAlgebra);
A := Matrix([[2, 1, 0], [1, 2, 1], [0, 1, 2]]);
```

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

```
GaussianElimination(A);
```

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 3/2 & 1 \\ 0 & 0 & 4/3 \end{bmatrix}$$

```
Eigenvalues(A);
```

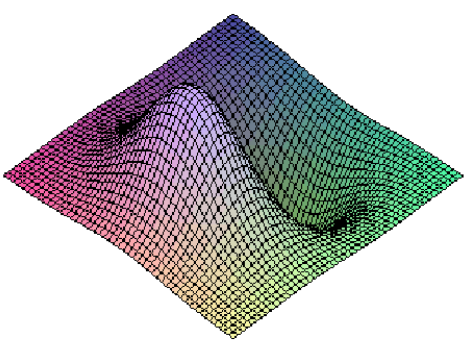
$$\begin{bmatrix} 2 \\ 2-\sqrt{2} \\ 2+\sqrt{2} \end{bmatrix}$$

Next, we give an example of solving numerically[1]:

```
> fsolve(2*x^5-11*x^4-7*x^3+12*x^2-4*x = 0);
                  -1.334383488, 0., 5.929222024
```

Finally, we show one of the graphic functions at work:

```
> plot3d(x*exp(-x^2-y^2),x = -2 .. 2,y = -2 .. 2,grid = [49, 49]);
```



For applications in research, general purpose systems are often not powerful enough: The implementation of the required basic algorithms may not be optimal with respect to speed and storage handling, and more advanced algorithms may not be implemented at all. Many special purpose systems were created by people working in a field other than computer algebra: they had a desperate need for computing power in the context of some of their research problems. A pioneering and prominent example is Veltman's SCHOONSHIP which helped to win a Nobel price in physics in 1999 (awarded to Veltman and t'Hooft 'for having placed particle physics theory on a firmer mathematical foundation').

[1] Note that only the real roots are computed.

Example P.2 GAP is a free open-source system for computational discrete algebra, with particular emphasis on Computational Group Theory. In the following GAP session, we define a subgroup G of the symmetric group S_{11} (the group of permutations of $\{1,\dots,11\}$) by giving two generators in cycle[2] notation. We check that G is simple (that is, its only normal subgroups are the trivial subgroup and the whole group itself). Then we compute the order $|G|$ of G, and factorize this number:

```
gap> G := Group([(1,2,3,4,5,6,7,8,9,10,11),(3,7,11,8)(4,10,5,6)]);
Group([(1,2,3,4,5,6,7,8,9,10,11), (3,7,11,8)(4,10,5,6)])
gap> IsSimple(G);
true
gap> size := Size(G);
7920
gap> Factors(size);
[ 2, 2, 2, 2, 3, 3, 5, 11 ]
```

From the factors, we see that G has a Sylow 2-subgroup[3] of order $2^4 = 16$. We use GAP to find such a group P:

```
gap> P := SylowSubgroup(G, 2);


Group([(2,8)(3,4)(5,6)(10,11), (3,5)(4,6)(7,9)(10,11),
     (2,4,8,3)(5,10,6,11)])
```

By making use of the *Small Groups Library* included in GAP, we can check that, up to isomorphism, there are 14 groups of order 16, and that P is the 8th group of order 16 listed in this library:

```
gap> SmallGroupsInformation(16);
 There are 14 groups of order 16.
 They are sorted by their ranks.
    1 is cyclic.
    2 - 9 have rank 2.
    10 - 13 have rank 3.
    14 is elementary abelian.
gap> IdGroup( P );
[ 16, 8 ]
```

Now, we determine what group P is. First, we check that P is neither Abelian nor the dihedral group of order 16 (the *dihedral group* of order $2n$ is the symmetry group of the regular n-gon):

[2] The cycle (4,10,5,6), for instance, maps 4 to 10, 10 to 5, 5 to 6, 6 to 4, and any other number to itself.

[3] If G is a finite group, and p is a prime divisor of its order $|G|$, then a subgroup U of G is called a *Sylow p-subgroup* if its order $|U|$ is the highest power of p dividing $|G|$.

```
gap> IsAbelian(P);
false
gap> IsDihedralGroup(P);
false
```

More information about P can be obtained by studying the subgroups of P of order 8. In fact, we consider the third such subgroup returned by GAP and name it H:

```
gap> H := SubgroupsOfIndexTwo(P)[3];
Group([(2,3,11,5,8,4,10,6)(7,9), (2,4,11,6,8,3,10,5)(7,9),
     (2,5,10,3,8,6,11,4)(7,9), (2,6,10,4,8,5,11,3)(7,9)])
gap> IdGroup(H);
[ 8, 1 ]
gap> IsCyclic(H);
true
```

Thus, H is the cyclic group C_8 of order 8 (*cyclic groups* are generated by just one element). Further checks show, in fact, that P is a semidirect product of C_8 and the cyclic group C_2. See Wild (2005) for the classification of groups of order 16.

Remark P.3 The group G studied in the previous example is known as the *Mathieu group* M_{11}. We should point out that researchers in group theory and representation theory have created quite a number of useful electronic libraries such as the *Small Groups Library* considered above.

Example P.4 MAGMA is a commercial system focusing on algebra, number theory, geometry and combinatorics. We use it to factorize the 8th Fermat number:

```
> Factorization(2^(2^8)+1);
[<1238926361552897,1>,
<93461639715357977769163558199606896584051237541638188580280321,1>]
```

Next, we meet our first example of an algebraic set: In *Weierstraß normal form*, an *elliptic curve* over a field K is a nonsingular[4] curve in the xy-plane defined by one polynomial equation of type

$$y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0,$$

with coefficients $a_i \in K$. In the following MAGMA session, we define an elliptic

[4] Informally, a curve is nonsingular if it admits a unique tangent line at each of its points. See, for instance, Silverman (2009) for a formal definition and for more information on elliptic curves.

curve E in Weierstraß normal form over the finite field F with 5^{90} elements by specifying the coefficients a_i. Then we count the number of points on E with coordinates in F.

```
F := FiniteField(5,90);
E := EllipticCurve([Zero(F),Zero(F),One(F),-One(F),Zero(F)]);
E;
Elliptic Curve defined by y^2 + y = x^3 + 4*x over GF(5^90)
#E;
807793566946316088741610050849537214477762546152780718396696352
```

The significance of elliptic curves stems from the fact that they carry an (additive) group law. Having specified a base point (the zero element of the group), the addition of points is defined by a geometric construction involving secant and tangent lines. For elliptic curves in Weierstraß normal form, it is convenient to choose the unique point at infinity of the curve as the base point (see Section 1.2.1 for points at infinity and Example P.6 below for a demonstration of the group law).

Remark P.5 Elliptic curves, most notably elliptic curves defined over $\mathbb{Q}$ respectively over a finite field, are of particular importance in number theory. They take center stage in the conjecture of Birch and Swinnerton-Dyer (1965) [5], they are key ingredients in the proof of Fermat's last theorem Wiles (1995), they are important for integer factorization Lenstra (1987), and they find applications in cryptography Koblitz (1987). As with many other awesome conjectures in number theory, the Birch and Swinnerton-Dyer conjecture is based on computer experiments.

Example P.6 SAGE is a free open-source mathematics software system which combines the power of many existing open-source packages into a common PYTHON-based interface. To show it at work, we start as in Example P.1 with computations from calculus. Then, we compute all prime numbers between two given numbers.

```
sage: limit(sin(x)/x, x=0)
1
sage: taylor(sqrt(x+1), x, 0, 5)
7/256*x^5 - 5/128*x^4 + 1/16*x^3 - 1/8*x^2 + 1/2*x + 1
sage: list(primes(10000000000, 10000000100))
[10000000019, 10000000033, 10000000061, 10000000069, 10000000097]
```

Finally, we define an elliptic curve E in Weierstraß normal form over $\mathbb{Q}$ and

[5] The Birch and Swinnerton-Dyer conjecture asserts, in particular, that an elliptic curve E over $\mathbb{Q}$ has an infinite number of points with rational coordinates iff its associated L-series satisfies $L(E,1) = 0$.

demonstrate the group law on this curve. The representation of the results takes infinity into account in the sense that the points are given by their homogeneous coordinates in the projective plane (see Section 1.2 for the projective setting). In particular, $(0 : 1 : 0)$ denotes the unique point at infinity of the curve which is chosen to be the zero element of the group.

```
sage: E = EllipticCurve([0,0,1,-1,0])
sage: E
Elliptic Curve defined by y^2 + y = x^3 - x over Rational Field
sage: P = E([0,0])
sage: P
(0 : 0 : 1)
sage: O = P - P
sage: O
(0 : 1 : 0)
sage: Q = E([-1,0])
sage: Q
(-1 : 0 : 1)
sage: Q + O
(-1 : 0 : 1)
sage: P + Q - (P+Q)
(0 : 1 : 0)
Q + (P + R) - ((Q + P) + R)
(0 : 1 : 0)
```

Among the systems combined by SAGE are MAXIMA, a general purpose system which is free and open-source, GAP, the system introduced in Example P.2, PARI/GP, a system for number theory, and SINGULAR, the system featured in these notes.

SINGULAR is a free open-source system for polynomial computations, with special emphasis on commutative and noncommutative algebra, algebraic geometry, and singularity theory. Like most other systems, SINGULAR consists of a precompiled kernel, written in C/C++, and additional packages, called libraries and written in the C-like SINGULAR user language. This language is interpreted on runtime. SINGULAR binaries are available for most common hardware and software platforms. Its release versions can be downloaded through `ftp` from

`ftp://www.mathematik.uni-kl.de/pub/Math/Singular/`

or via your favorite web browser from SINGULAR's webpage

`http://www.singular.uni-kl.de/` .

SINGULAR also provides an extensive online manual and help function. See its webpage or enter `help;` in a SINGULAR session.

Most algorithms implemented in SINGULAR rely on the basic task of computing Gröbner bases. Gröbner bases are special sets of generators for ideals in polynomial rings. Their definition and computation is subject to the choice of a monomial ordering such as the *lexicographical ordering* $>_{\mathrm{lp}}$ and the *degree reverse lexicographical ordering* $>_{\mathrm{dp}}$. We will treat Gröbner bases and their computation by Buchberger's algorithm in Chapter 2. SINGULAR examples, however, will already be presented beforehand.

SINGULAR Example P.7 We enter the polynomials of the system

$$\begin{aligned} x+y+z-1 &= 0\\ x^2+y^2+z^2-1 &= 0\\ x^3+y^3+z^3-1 &= 0 \end{aligned}$$

in a SINGULAR session. For this, we first have to define the corresponding polynomial ring which is named R and endowed with the lexicographical ordering. Note that the 0 in the definition of R refers to the prime field of characteristic zero, that is, to $\mathbb{Q}$.

```
> ring R = 0, (x,y,z), lp;
> poly f1 = x+y+z-1;
> poly f2 = x2+y2+z2-1;
> poly f3 = x3+y3+z3-1;
```

Next, we define the ideal generated by the polynomials and compute a Gröbner basis for this ideal (the system given by the Gröbner basis elements has the same solutions as the original system).

```
> ideal I = f1, f2, f3;
> ideal GI = groebner(I);  GI;
GI[1]=z3-z2
GI[2]=y2+yz-y+z2-z
GI[3]=x+y+z-1
```

In the first equation of the new system, the variables x and y are eliminated. In the second, x is eliminated. As a consequence, the solutions can now be directly read off:

$$(1,0,0),\ (0,1,0),\ (0,0,1).$$

The example indicates that $>_{\mathrm{lp}}$ is what we will call an *elimination ordering*. If such an ordering is chosen, Buchberger's algorithm generalizes Gaussian elimination. For most applications of the algorithm, however, the elimination property is not needed. It is, then, usually more efficient to choose the ordering $>_{\mathrm{dp}}$.

Multivariate polynomial factorization is another basic task on which some of the more advanced algorithms in SINGULAR rely. Starting with the first computer algebra systems in the 1960s, the design of algorithms for polynomial factorization has always been an active area of research. To keep the size of our notes within reasonable limits, we will not treat this here. We should point out, however, that algorithms for polynomial factorization do not depend on monomial orderings. Nevertheless, choosing such an ordering is always part of a ring definition in SINGULAR.

SINGULAR Example P.8 We factorize a polynomial in $\mathbb{Q}[x,y,z]$ using the SINGULAR command `factorize`. The resulting output is a list, with the factors as the first entry, and the corresponding multiplicities as a second.

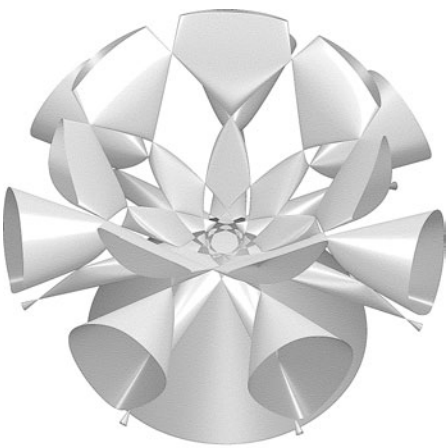
```
> ring R = 0, (x,y,z), dp;
> poly f = -x7y4+x6y5-3x5y6+3x4y7-3x3y8+3x2y9-xy10+y11-x10z
. +x8y2z+9x6y4z+11x4y6z+4x2y8z-3x5y4z2+3x4y5z2-6x3y6z2+6x2y7z2
. -3xy8z2+3y9z2-3x8z3+6x6y2z3+21x4y4z3+12x2y6z3-3x3y4z4+3x2y5z4
. -3xy6z4+3y7z4-3x6z5+9x4y2z5+12x2y4z5-xy4z6+y5z6-x4z7+4x2y2z7;
> factorize(f);
[1]:
   _[1]=-1
   _[2]=xy4-y5+x4z-4x2y2z
   _[3]=x2+y2+z2
[2]:
   1,1,3
```

Remark P.9 In recent years, quite a number of the more abstract concepts in algebraic geometry have been made constructive. They are, thus, not only easier to understand, but can be handled by computer algebra. A prominent example is the desingularization theorem of Hironaka (see Hironaka (1964)) for which Hironaka received the Fields Medal. In fact, Villamajor's constructive version of Hironaka's proof has led to an algorithm whose SINGULAR implementation allows us to resolve singularities in many cases of interest (see Bierstone and Milman (1997), Frühbis-Krüger and Pfister (2006), Bravo et al. (2005)).

When studying plane curves or surfaces in 3-space, it is often desirable to visualize the geometric objects under consideration. Excellent tools for this are

SURF and its descendants SURFEX[6] and SURFER[7]. Comparing these, we note that SURFEX has more features, whereas SURFER is easier to handle.

Example P.10 The following SURFER picture shows a surface in 3-space found by Oliver Labs using SINGULAR (see Labs (2006)):



SINGULAR Example P.11 We set up the equation of Labs' surface in SINGULAR. The equation is defined over a finite extension field of $\mathbb{Q}$ which we implement by entering its minimal polynomial:

```
> ring R = (0,a), (x,y,w,z), dp;
> minpoly = a^3 + a + 1/7;
> poly a(1) = -12/7*a^2 - 384/49*a - 8/7;
> poly a(2) = -32/7*a^2  + 24/49*a   - 4;
> poly a(3) =    -4*a^2  + 24/49*a   - 4;
> poly a(4) =  -8/7*a^2   + 8/49*a - 8/7;
> poly a(5) =    49*a^2      - 7*a  + 50;
> poly P = x*(x^6-3*7*x^4*y^2+5*7*x^2*y^4-7*y^6)
. +7*z*((x^2+y^2)^3-2^3*z^2*(x^2+y^2)^2
. +2^4*z^4*(x^2+y^2))-2^6*z^7;
> poly C = a(1)*z^3+a(2)*z^2*w+a(3)*z*w^2+a(4)*w^3+(z+w)*(x^2+y^2);
> poly S = P-(z+a(5)*w)*C^2;
> homog(S);  // returns 1 if poly is homogeneous
1
> deg(S);
7
```

We see that S is a homogeneous polynomial of degree 7. It defines Labs' sur-

[6] http://surf.sourceforge.net
[7] http://www.oliverlabs.net/welcome.php

face in *projective* 3-space. This surface is a 'world record' surface in that it has the maximal number of nodes known for a degree-7 surface in projective 3-space (a node constitutes the most simple type of a singularity). We use SINGULAR to confirm that there are precisely 99 nodes (and no other singularities).

First, we compute the dimension of the locus of singularities via the Jacobian criterion (see Decker and Schreyer (2013) for the criterion and Sections 1.1.8 and 2.3 for more on dimension):

```
> dim(groebner(jacob(S)))-1;
0
```

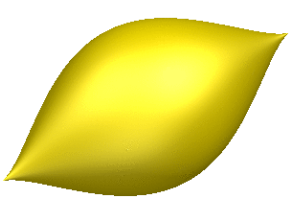
The result means that there are only finitely many singularities. By checking that the nonnodal locus is empty, we verify that all singularities are nodes. Then, we compute the number of nodes:

```
> dim(groebner(minor(jacob(jacob(S)),2))) - 1;
-1
> mult(groebner(jacob(S)));
99
```

SINGULAR Example P.12 When properly installed, SURF, SURFEX, and SURFER can be called from SINGULAR. To give an example, we use SURFER to plot a surface which, as it turns out, resembles a citrus fruit. To begin, we load the SINGULAR library connecting to SURF and SURFER.

```
> LIB "surf.lib";
> ring R = 0, (x,y,z), dp;
> ideal I = 6/5*y^2+6/5*z^2-5*(x+1/2)^3*(1/2-x)^3;
surfer(I);
```

The resulting picture will show in a pop-up window:



See `http://www.imaginary-exhibition.com` for more pictures.