# Ideals, Varieties and *Macaulay 2*

Bernd Sturmfels*

This chapter introduces *Macaulay 2* commands for some elementary computations in algebraic geometry. Familiarity with Gröbner bases is assumed.

Many students and researchers alike have their first encounter with Gröbner bases through the delightful text books [1] and [2] by David Cox, John Little and Donal O'Shea. This chapter illustrates the use of *Macaulay 2* for some computations discussed in these books. It can be used as a supplement for an advanced undergraduate course or first-year graduate course in computational algebraic geometry. The mathematically advanced reader will find this chapter a useful summary of some basic *Macaulay 2* commands.

## 1 A Curve in Affine Three-Space

Our first example concerns geometric objects in (complex) affine 3-space. We start by setting up the ring of polynomial functions with rational coefficients.

```
i1 : R = QQ[x,y,z]

o1 = R

o1 : PolynomialRing
```

Various monomial orderings are available in *Macaulay 2*; since we did not specify one explicitly, the monomials in the ring `R` will be sorted in graded reverse lexicographic order [1, §I.2, Definition 6]. We define an ideal generated by two polynomials in this ring and assign it to the variable named `curve`.

```
i2 : curve = ideal( x^4-y^5, x^3-y^7 )

                    5    4     7    3
o2 = ideal (- y  + x , - y  + x )

o2 : Ideal of R
```

We compute the reduced Gröbner basis of our ideal:

```
i3 : gb curve

o3 = | y5-x4 x4y2-x3 x8-x3y3 |

o3 : GroebnerBasis
```

By inspecting leading terms (and using [1, §9.3, Theorem 8]), we see that our ideal `curve` does indeed define a one-dimensional affine variety. This can be tested directly with the following commands in *Macaulay 2*:

```
i4 : dim curve

o4 = 1
```

* Partially supported by the National Science Foundation (DMS-9970254).

```
i5 : codim curve

o5 = 2
```

The *degree* of a curve in complex affine 3-space is the number of intersection points with a general plane. It coincides with the degree [2, §6.4] of the projective closure [1, §8.4] of our curve, which we compute as follows:

```
i6 : degree curve

o6 = 28
```

The Gröbner basis in `o3` contains two polynomials which are not irreducible: they contain a factor of $x^3$. This shows that our curve is not irreducible over $\mathbf{Q}$. We first extract the components which are transverse to the plane $x = 0$:

```
i7 : curve1 = saturate(curve,ideal(x))

                2       5    4   5    3
o7 = ideal (x*y  - 1, y  - x , x  - y )

o7 : Ideal of R
```

And next we extract the component which lies in the plane $x = 0$:

```
i8 : curve2 = saturate(curve,curve1)

              3   5
o8 = ideal (x , y )

o8 : Ideal of R
```

The second component is a multiple line. Hence our input ideal was not radical. To test equality of ideals we use the command `==` .

```
i9 : curve == radical curve

o9 = false
```

We now replace our curve by its first component:

```
i10 : curve = curve1

                 2       5    4   5    3
o10 = ideal (x*y  - 1, y  - x , x  - y )

o10 : Ideal of R

i11 : degree curve

o11 = 13
```

The ideal of this curve is radical:

```
i12 : curve == radical curve

o12 = true
```

Notice that the variable **z** does not appear among the generators of the ideal. Our curve consists of 13 straight lines (over $\mathbf{C}$) parallel to the `z`-axis.

## 2 Intersecting Our Curve With a Surface

In this section we explore basic operations on ideals, starting with those described in [1, §4.3]. Consider the following surface in affine 3-space:

```
i13 : surface = ideal( x^5 + y^5 + z^5 - 1)

                5    5    5
o13 = ideal(x  + y  + z  - 1)

o13 : Ideal of R
```

The union of the curve and the surface is represented by the intersection of their ideals:

```
i14 : theirunion = intersect(curve,surface)

                 6 2      7      2 5    5    5    5      2      5 5    1 ···
o14 = ideal (x y  + x*y  + x*y z  - x  - y  - z  - x*y  + 1, x y  + y  ···

o14 : Ideal of R
```

In this example this coincides with the product of the two ideals:

```
i15 : curve*surface == theirunion

o15 = true
```

The intersection of the curve and the surface is represented by the sum of their ideals. We get a finite set of points:

```
i16 : ourpoints = curve + surface

                   2       5    4   5    3   5    5    5
o16 = ideal (x*y  - 1, y  - x , x  - y , x  + y  + z  - 1)

o16 : Ideal of R

i17 : dim ourpoints

o17 = 0
```

The number of points is sixty five:

```
i18 : degree ourpoints

o18 = 65
```

Each of the points is multiplicity-free:

```
i19 : degree radical ourpoints

o19 = 65
```

The number of points coincides with the number of monomials not in the initial ideal [2, §2.2]. These are called the *standard monomials*.

```
i20 : staircase = ideal leadTerm ourpoints

                   2   5   5   5
o20 = ideal (x*y , z , y , x )

o20 : Ideal of R
```

The `basis` command can be used to list all the standard monomials

```
i21 : T = R/staircase;

i22 : basis T

o22 = | 1 x x2 x3 x4 x4y x4yz x4yz2 x4yz3 x4yz4 x4z x4z2 x4z3 x4z4 x3y ···

                1       65
o22 : Matrix T  <--- T
```

The assignment of the quotient ring to the global variable `T` had a side effect: the variables `x`, `y`, and `z` now have values in that ring. To bring the variables of `R` to the fore again, we must say:

```
i23 : use R;
```

Every polynomial function on our 65 points can be written uniquely as a linear combination of these standard monomials. This representation can be computed using the normal form command `%`.

```
i24 : anyOldPolynomial = y^5*x^5-x^9-y^8+y^3*x^5

       5 5    9    5 3    8
o24 = x y  - x  + x y  - y

o24 : R

i25 : anyOldPolynomial % ourpoints

       4     3
o25 = x y - x y

o25 : R
```

Clearly, the normal form is zero if and only the polynomial is in the ideal.

```
i26 : anotherPolynomial = y^5*x^5-x^9-y^8+y^3*x^4

       5 5    9    8    4 3
o26 = x y  - x  - y  + x y

o26 : R

i27 : anotherPolynomial % ourpoints

o27 = 0

o27 : R
```

## 3 Changing the Ambient Polynomial Ring

During a *Macaulay 2* session it sometimes becomes necessary to change the ambient ring in which the computations takes place. Our original ring, defined in `i1`, is the polynomial ring in three variables over the field $\mathbf{Q}$ of rational numbers with the graded reverse lexicographic order. In this section two modifications are made: first we replace the field of coefficients by a finite field, and later we replace the monomial order by an elimination order.

An important operation in algebraic geometry is the decomposition of algebraic varieties into irreducible components [1, §4.6]. Algebraic algorithms for this purpose are based on the *primary decomposition* of ideals [1, §4.7]. A future version of *Macaulay 2* will have an implementation of primary decomposition over any polynomial ring. The current version of *Macaulay 2* has a command `decompose` for finding all the minimal primes of an ideal, but, as it stands, this works only over a finite field.

Let us change our coefficient field to the field with 101 elements:

```
i28 : R' = ZZ/101[x,y,z];
```

We next move our ideal from the previous section into the new ring (fortunately, none of the coefficients of its generators have 101 in the denominator):

```
i29 : ourpoints' = substitute(ourpoints,R')

                 2       5    4    5    3    5    5    5
o29 = ideal (x*y  - 1, y  - x , x  - y , x  + y  + z  - 1)

o29 : Ideal of R'

i30 : decompose ourpoints'

                                                                     ...
o30 = {ideal (z + 36, y - 1, x - 1), ideal (z + 1, y - 1, x - 1), idea ...

o30 : List
```

Oops, that didn't fit on the display, so let's print them out one per line.

```
i31 : oo / print @@ print;
ideal (z + 36, y - 1, x - 1)

ideal (z + 1, y - 1, x - 1)

ideal (z - 6, y - 1, x - 1)

ideal (z - 14, y - 1, x - 1)

ideal (z - 17, y - 1, x - 1)

        3      2              2                        3    2     2      ...
ideal (x  - 46x  + 28x*y - 27y  + 46x + y + 27, - 16x  + x y + x  - 15 ...

          2                                           2                ...
ideal (- 32x  - 16x*y + x*z - 16x - 27y - 30z - 14, - 34x  - 14x*y + y ...

          2                                         2          2       ...
ideal (44x  + 22x*y + x*z + 22x - 26y - 30z - 6, 18x  + 12x*y + y  + 1 ...

          2                                            2             2 ...
ideal (- 41x  + 30x*y + x*z + 30x + 38y - 30z + 1, - 26x  - 10x*y + y  ...

          2                                             2            2 ...
ideal (39x  - 31x*y + x*z - 31x - 46y - 30z + 36, - 32x  - 13x*y + y  ...

          2                                             2            2 ...
ideal (- 10x  - 5x*y + x*z - 5x - 40y - 30z - 17, - 37x  + 35x*y + y  ...
```

If we just want to see the degrees of the irreducible components, then we say:

```
i32 : ooo / degree

o32 = {1, 1, 1, 1, 1, 30, 6, 6, 6, 6, 6}

o32 : List
```

Note that the expressions `oo` and `ooo` refer to the previous and prior-to-previous output lines respectively.

Suppose we wish to compute the $x$-coordinates of our sixty five points. Then we must use an elimination order, for instance, the one described in [1, §3.2, Exercise 6.a]. We define a new polynomial ring with the elimination order for $\{y,z\} > \{x\}$ as follows:

```
i33 : S = QQ[z,y,x, MonomialOrder => Eliminate 2]

o33 = S

o33 : PolynomialRing
```

We move our ideal into the new ring,

```
i34 : ourpoints'' = substitute(ourpoints,S)

               2       5    4     3    5   5    5    5
o34 = ideal (y x - 1, y  - x , - y  + x , z  + y  + x  - 1)

o34 : Ideal of S
```

and we compute the reduced Gröbner basis in this new order:

```
i35 : G = gens gb ourpoints''

o35 = | x13-1 y-x6 z5+x5+x4-1 |

              1       3
o35 : Matrix S  <--- S
```

To compute the elimination ideal we use the following command:

```
i36 : ideal selectInSubring(1,G)

             13
o36 = ideal(x   - 1)

o36 : Ideal of S
```

## 4 Monomials Under the Staircase

Invariants of an algebraic variety, such as its dimension and degree, are computed from an initial monomial ideal. This computation amounts to the combinatorial task of analyzing the collection of standard monomials, that is, the monomials under the staircase [1, Chapter 9]. In this section we demonstrate some basic operations on monomial ideals in *Macaulay 2*.

Let us create a non-trivial staircase in three dimensions by taking the third power of the initial monomial from line `i20`.

```
i37 : M = staircase^3

              3 6   2 4 5   2 9   7 4     2 10     7 5   6 2 5     12  ···
o37 = ideal (x y , x y z , x y , x y , x*y z  , x*y z , x y z , x*y  , ···

o37 : Ideal of R
```

The number of current generators of this ideal equals

```
i38 : numgens M

o38 = 20
```

To see all generators we can transpose the matrix of minimal generators:

```
i39 : transpose gens M
```

```
o39 = {-9}  | x3y6    |
      {-11} | x2y4z5  |
      {-11} | x2y9    |
      {-11} | x7y4    |
      {-13} | xy2z10  |
      {-13} | xy7z5   |
      {-13} | x6y2z5  |
      {-13} | xy12    |
      {-13} | x6y7    |
      {-13} | x11y2   |
      {-15} | z15     |
      {-15} | y5z10   |
      {-15} | x5z10   |
      {-15} | y10z5   |
      {-15} | x5y5z5  |
      {-15} | x10z5   |
      {-15} | y15     |
      {-15} | x5y10   |
      {-15} | x10y5   |
      {-15} | x15     |

              20       1
o39 : Matrix R   <--- R
```

Note that this generating set is not minimal; see o48 below. The number of standard monomials equals

```
i40 : degree M

o40 = 690
```

To list all the standard monomials we first create the residue ring

```
i41 : S = R/M

o41 = S

o41 : QuotientRing
```

and then we ask for a vector space basis of the residue ring:

```
i42 : basis S

o42 = | 1 x x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x14y x14yz x14 ···

              1       690
o42 : Matrix S  <--- S
```

Let us count how many standard monomials there are of a given degree. The following table represents the Hilbert function of the residue ring.

```
i43 : tally apply(flatten entries basis(S),degree)

o43 = Tally{{0} => 1  }
            {1} => 3
            {10} => 63
            {11} => 69
            {12} => 73
            {13} => 71
            {14} => 66
            {15} => 53
            {16} => 38
            {17} => 23
            {18} => 12
```

```
             {19} => 3
             {2} => 6
             {3} => 10
             {4} => 15
             {5} => 21
             {6} => 28
             {7} => 36
             {8} => 45
             {9} => 54

o43 : Tally
```

Thus the largest degree of a standard monomial is nineteen, and there are three standard monomials of that degree:

```
i44 : basis(19,S)

o44 = | x14yz4 x9yz9 x4yz14 |

              1       3
o44 : Matrix S  <--- S
```

The most recently defined ring involving `x`, `y`, and `z` was `S`, so all computations involving those variables are done in the residue ring `S`. For instance, we can also obtain the standard monomials of degree nineteen as follows:

```
i45 : (x+y+z)^19

            14   4           9   9          4   14
o45 = 58140x  y*z  + 923780x y*z  + 58140x y*z

o45 : S
```

An operation on ideals which will occur frequently throughout this book is the computation of minimal free resolutions. This is done as follows:

```
i46 : C = res M

       1      16      27      12
o46 = R  <-- R   <-- R   <-- R   <-- 0

      0      1       2       3       4

o46 : ChainComplex
```

This shows that our ideal `M` has sixteen minimal generators. They are the entries in the leftmost matrix of the chain complex `C`:

```
i47 : C.dd_1

o47 = | x3y6 x7y4 x2y9 x2y4z5 x11y2 xy12 x6y2z5 xy7z5 xy2z10 x15 y15 x ···

              1      16
o47 : Matrix R  <--- R
```

This means that four of the twenty generators in `o39` were redundant. We construct the set consisting of the four redundant generators as follows:

```
i48 : set flatten entries gens M - set flatten entries C.dd_1

            6 7   10 5   5 10   5 5 5
o48 = Set {x y , x  y , x y  , x y z }

o48 : Set
```

Here `flatten entries` turns the matrix `M` into a single list. The command `set` turns that list into a set, to which we can apply the difference operation for sets.

Let us now take a look at the first syzygies (or *minimal S-pairs* [1, §2.9]) among the sixteen minimal generators. They correspond to the columns of the second matrix in our resolution `C`:

```
i49 : C.dd_2

o49 = {9}  | -y3 -x4 0   -z5 0   0   0   0   0   0   0   0   0   0   0 ···
      {11} | 0   y2  0   0   0   -x4 0   0   -z5 0   0   0   0   0   0 ···
      {11} | x   0   -y3 0   0   0   0   0   0   -z5 0   0   0   0   0 ···
      {11} | 0   0   0   xy2 -y3 0   -x4 0   x5  y5  0   -z5 0   0   0 ···
      {13} | 0   0   0   0   0   y2  0   0   0   0   0   0   0   -x4 0 ···
      {13} | 0   0   x   0   0   0   0   -y3 0   0   0   0   0   0   0 ···
      {13} | 0   0   0   0   0   0   y2  0   0   0   0   0   0   0   - ···
      {13} | 0   0   0   0   x   0   0   0   0   0   -y3 0   0   0   0 ···
      {13} | 0   0   0   0   0   0   0   0   0   0   0   xy2 -y3 0   0 ···
      {15} | 0   0   0   0   0   0   0   0   0   0   0   0   0   y2  0 ···
      {15} | 0   0   0   0   0   0   0   x   0   0   0   0   0   0   0 ···
      {15} | 0   0   0   0   0   0   0   0   0   0   0   0   0   0   y ···
      {15} | 0   0   0   0   0   0   0   0   0   0   x   0   0   0   0 ···
      {15} | 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 ···
      {15} | 0   0   0   0   0   0   0   0   0   0   0   0   x   0   0 ···
      {15} | 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 ···

               16       27
o49 : Matrix R   <--- R
```

The first column represents the S-pair between the first generator $x^3y^6$ and the third generator $x^2y^9$. It is natural to form the *S-pair graph* with 16 vertices and 27 edges represented by this matrix. According to the general theory described in [3], this is a planar graph with 12 regions. The regions correspond to the 12 second syzygies, that is, to the columns of the matrix

```
i50 : C.dd_3

o50 = {12} | z5  0   0   0   0   0   0   0   0   0   0   0   |
      {13} | 0   z5  0   0   0   0   0   0   0   0   0   0   |
      {14} | 0   0   z5  0   0   0   0   0   0   0   0   0   |
      {14} | -y3 -x4 0   0   0   0   0   0   0   0   0   0   |
      {14} | 0   0   -y5 z5  0   0   0   0   0   0   0   0   |
      {15} | 0   0   0   0   z5  0   0   0   0   0   0   0   |
      {15} | 0   0   0   0   -x5 z5  0   0   0   0   0   0   |
      {16} | 0   0   0   0   0   0   z5  0   0   0   0   0   |
      {16} | 0   y2  0   0   -x4 0   0   0   0   0   0   0   |
      {16} | x   0   -y3 0   0   0   0   0   0   0   0   0   |
      {16} | 0   0   0   0   0   0   -y5 z5  0   0   0   0   |
      {16} | 0   0   0   -y3 0   -x4 0   0   0   0   0   0   |
      {16} | 0   0   0   0   0   0   0   -y5 z5  0   0   0   |
      {17} | 0   0   0   0   0   0   0   0   0   z5  0   0   |
      {17} | 0   0   0   0   0   0   0   0   0   -x5 z5  0   |
      {17} | 0   0   0   0   0   0   0   0   0   0   -x5 z5  |
      {18} | 0   0   0   0   y2  0   0   0   0   -x4 0   0   |
      {18} | 0   0   x   0   0   0   -y3 0   0   0   0   0   |
      {18} | 0   0   0   0   0   y2  0   0   0   0   -x4 0   |
      {18} | 0   0   0   x   0   0   0   -y3 0   0   0   0   |
      {18} | 0   0   0   0   0   0   0   0   -y3 0   0   -x4 |
      {20} | 0   0   0   0   0   0   0   0   0   y2  0   0   |
```

```
      {20} | 0   0   0   0   0   0   x   0   0   0   0   0   |
      {20} | 0   0   0   0   0   0   0   0   0   0   y2  0   |
      {20} | 0   0   0   0   0   0   0   x   0   0   0   0   |
      {20} | 0   0   0   0   0   0   0   0   0   0   0   y2  |
      {20} | 0   0   0   0   0   0   0   0   x   0   0   0   |

                 27        12
o50 : Matrix R   <--- R
```

But we are getting ahead of ourselves. Homological algebra and resolutions will be covered in the next chapter, and monomial ideals will appear in the chapter of Hoşten and Smith. Let us return to Cox, Little and O'Shea [2].

## 5 Pennies, Nickels, Dimes and Quarters

We now come to an application of Gröbner bases which appears in [2, Section 8.1]: *Integer Programming*. This is the problem of minimizing a linear objective function over the set of non-negative integer solutions of a system of linear equations. We demonstrate some techniques for doing this in *Macaulay 2*. Along the way, we learn about multigraded polynomial rings and how to compute Gröbner bases with respect to monomial orders defined by weights. Our running example is the linear system defined by the matrix:

```
i51 : A = {{1, 1, 1, 1},
           {1, 5,10,25}}

o51 = {{1, 1, 1, 1}, {1, 5, 10, 25}}

o51 : List
```

For the algebraic study of integer programming problems, a good starting point is to work in a multigraded polynomial ring, here in four variables:

```
i52 : R = QQ[p,n,d,q, Degrees => transpose A]

o52 = R

o52 : PolynomialRing
```

The degree of each variable is the corresponding column vector of the matrix Each variable represents one of the four coins in the U.S. currency system:

```
i53 : degree d

o53 = {1, 10}

o53 : List

i54 : degree q

o54 = {1, 25}

o54 : List
```

Each monomial represents a collection of coins. For instance, suppose you own four pennies, eight nickels, ten dimes, and three quarters:

```
i55 : degree(p^4*n^8*d^10*q^3)

o55 = {25, 219}

o55 : List
```

Then you have a total of 25 coins worth two dollars and nineteen cents. There are nine other possible ways of having 25 coins of the same value:

```
i56 : h = basis({25,219}, R)

o56 = | p14n2d2q7 p9n8d2q6 p9n5d6q5 p9n2d10q4 p4n14d2q5 p4n11d6q4 p4n8 ···

                1       9
o56 : Matrix R  <--- R
```

For just counting the number of columns of this matrix we can use the command

```
i57 : rank source h

o57 = 9
```

How many ways can you make change for ten dollars using 100 coins?

```
i58 : rank source basis({100,1000}, R)

o58 = 182
```

A typical integer programming problem is this: among all 182 ways of expressing ten dollars using 100 coins, which one uses the fewest dimes? We set up the Conti-Traverso algorithm [2, §8.1] for answering this question. We use the following ring with the lexicographic order and with the variable order: dimes (d) before pennies (p) before nickels (n) before quarters (q).

```
i59 : S = QQ[x, y, d, p, n, q,
          MonomialOrder => Lex, MonomialSize => 16]

o59 = S

o59 : PolynomialRing
```

The option `MonomialSize` advises *Macaulay 2* to use more space to store the exponents of monomials, thereby avoiding a potential overflow.

We define an ideal with one generator for each column of the matrix A.

```
i60 : I = ideal( p - x*y, n - x*y^5, d - x*y^10, q - x*y^25)

                             5            10             25
o60 = ideal (- x*y + p, - x*y  + n, - x*y   + d, - x*y   + q)

o60 : Ideal of S
```

The integer program is solved by normal form reduction with respect to the following Gröbner basis consisting of binomials.

```
i61 : transpose gens gb I

o61 = {-6}  | p5q-n6      |
      {-4}  | d4-n3q      |
      {-3}  | yn2-dp      |
      {-6}  | yp4q-dn4    |
      {-4}  | yd3-pnq     |
      {-6}  | y2p3q-d2n2  |
      {-5}  | y2d2n-p2q   |
      {-7}  | y2d2p3-n5   |
      {-6}  | y3p2q-d3    |
      {-6}  | y3dp2-n3    |
      {-5}  | y4p-n       |
      {-6}  | y5n-d       |
```

```
         {-8}  | y6d2-pq    |
         {-16} | y15d-q     |
         {-7}  | xq-y5d2    |
         {-5}  | xn-y3p2    |
         {-2}  | xd-n2      |
         {-2}  | xy-p       |

                 18       1
o61 : Matrix S   <--- S
```

We fix the quotient ring, so the reduction to normal form will happen automatically.

```
i62 : S' = S/I

o62 = S'

o62 : QuotientRing
```

You need at least two dimes to express one dollar with ten coins.

```
i63 : x^10 * y^100

       2 6 2
o63 = d n q

o63 : S'
```

But you can express ten dollars with a hundred coins none of which is a dime.

```
i64 : x^100 * y^1000

       75 25
o64 = n  q

o64 : S'
```

The integer program is infeasible if and only if the normal form still contains the variable $x$ or the variable $y$. For instance, you cannot express ten dollars with less than forty coins:

```
i65 : x^39 * y^1000

       25 39
o65 = y  q

o65 : S'
```

We now introduce a new term order on the polynomial ring, defined by assigning a weight to each variable. Specifically, we assign weights for each of the coins. For instance, let pennies have weight 5, nickels weight 7, dimes weight 13 and quarters weight 17.

```
i66 : weight = (5,7,13,17)

o66 = (5, 7, 13, 17)

o66 : Sequence
```

We set up a new ring with the resulting weight term order, and work modulo the same ideal as before in this new ring.

```
i67 : T = QQ[x, y, p, n, d, q,
                Weights => {{1,1,0,0,0,0},{0,0,weight}},
                MonomialSize => 16]/
            (p - x*y, n - x*y^5, d - x*y^10, q - x*y^25);
```

One dollar with ten coins:

```
i68 : x^10 * y^100

       5 2 3
o68 = p d q

o68 : T
```

Ten dollars with one hundred coins:

```
i69 : x^100 * y^1000

       60 3 37
o69 = p  n q

o69 : T
```

Here is an optimal solution which involves all four types of coins:

```
i70 : x^234 * y^5677

       2 4 3 225
o70 = p n d q

o70 : T
```

## References


1. David Cox, John Little, and Donal O'Shea: *Ideals, varieties, and algorithms.* Springer-Verlag, New York, second edition, 1997. An introduction to computational algebraic geometry and commutative algebra.
2. David Cox, John Little, and Donal O'Shea: *Using algebraic geometry.* Springer-Verlag, New York, 1998.
3. Ezra Miller and Bernd Sturmfels: Monomial ideals and planar graphs. In S. Lin M. Fossorier, H. Imai and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 1719 of *Springer Lecture Notes in Computer Science*, pages 19–28, 1999.

# Index