`groebner` — library for Gröbner bases

Table of contents

Introduction

The `groebner` package contains some functions dealing with ideals of multivariate polynomial rings over a field. In particular, Gröbner bases of such ideals can be computed.

An ideal is given by a list of generators. They must all be polynomials of the same type, i.e., for all of them, the coefficient ring (third operand) and list of unknowns (second operand) must be the same. The generators may also be expressions (all of them must be, if any of them is); they are understood as polynomials over the rationals in this case.

Gröbner bases and related notions depend on the monomial ordering (also called term ordering) under consideration. MuPAD knows the following orderings:

- the lexicographical ordering, denoted by the identifier `LexOrder`.
- the ordering by total degree, with the lexicographical ordering used as a tie-break; it is denoted by the identifier `DegreeOrder`.
- the ordering by total degree, with the opposite of the lexicographical ordering for the reverse order of unknowns used as a tie-break (i.e., the monomial that is lexicographically smaller if the order of variables is reversed, is considered the bigger one); this one is denoted by `DegInvLexOrder`.
- user-defined orderings. They constitute a domain `Dom::MonomOrdering` of their own.

Orderings always refer to the order of the unknowns of the polynomial; e.g., x is lexicographically bigger than y in $F[x, y]$, but smaller than y when regarded as an element of $F[y, x]$.

`groebner::dimension` – the dimension of the affine variety generated by polynomials

`groebner::dimension(polys)` computes the dimension of the affine variety generated by the polynomials in the set or list `polys`.

Call(s):

- `groebner::dimension(polys <, order>)`

Parameters:

`polys` — a list or set of polynomials of the same type. Alternatively, a list or set of polynomial expressions with rational coefficients.

`order` — one of the identifiers *`DegInvLexOrder`*, *`DegreeOrder`*, and *`LexOrder`*, or a user-defined term ordering of type `Dom::MonomOrdering`. The default ordering is *`DegInvLexOrder`*.

Return Value: a nonnegative integer

Related Functions: `groebner::gbasis`, `poly`

Details:

- The rules laid down in the introduction to the `groebner` package concerning the polynomial types and the ordering apply.
- The polynomials in the list `polys` must all be of the same type. In particular, do not mix polynomials created via `poly` and polynomial expressions!

Example 1. An example from the book of Cox, Little and O'Shea (see below):

```
>> groebner::dimension([y^2*z^3, x^5*z^4, x^2*y*z^2])
                                    2
```

Background:

- First, the Gröbner basis of the given polynomials with respect to the given monomial ordering is computed using `groebner::gbasis`. This Gröbner basis is then used to compute the dimension of the affine variety generated by the polynomials.
- The implemented algorithm is described in Cox, Little, O'Shea: "Ideals, Varieties and Algorithms", Springer, 1992, Chapter 9.

Changes:

- Further term orderings from `Dom::MonomOrdering` are handled. The special term orderings from the Gröbner package were moved there.

`groebner::gbasis` – computation of a reduced Gröbner basis

`groebner::gbasis(polys)` computes a reduced Gröbner basis of the ideal generated by the polynomials in the list `polys`.

Call(s):

- `groebner::gbasis(polys <, order> <, Reorder>)`

Parameters:

`polys` — a list of polynomials of the same type. Alternatively, a list of polynomial expressions with rational coefficients.

`order` — one of the identifiers *`DegInvLexOrder`*, *`DegreeOrder`*, and *`LexOrder`*, or a user-defined term ordering of type `Dom::MonomOrdering`. The default ordering is *`DegInvLexOrder`*.

Options:

`Reorder` — With this option `groebner::gbasis` internally may change the lexicographical ordering of variables to decrease running time.

Return Value: a list of polynomials. The output polynomials have the same type as the polynomials of the input list.

Related Functions: `poly`

Details:

- The rules laid down in the introduction to the `groebner` package concerning the polynomial types and the ordering apply.
- The polynomials in the list `polys` must all be of the same type. In particular, do not mix polynomials created via `poly` and polynomial expressions!
- The ordering strategy indicated by *`Reorder`* is used automatically when polynomial expressions are used.

Option <*Reorder*>:

- With this option the variables are sorted internally such that they have a "heuristic optimal" ordering. Consequently, the ordering of the variables in the output polynomials may differ from their ordering in the input polynomials. For details on the ordering strategy, see W. Boege, R. Gebauer und H. Kredel: "Some Examples for Solving Systems of Algebraic Equations by Calculating Groebner Bases" im J. Symbolic Comp. (1986) Vol. 1, 83-98.
- Re-ordering is always applied when polynomial expressions are used for input.

Example 1. We demonstrate the effect of various input formats. First, we use polynomial expressions to define the polynomial ideal. The Gröbner basis is returned as a list of polynomial expressions:

```
>> groebner::gbasis([x^2 - y^2, x^2 + y], LexOrder)

                          4    2       2
                        [x  - x , y + x ]
```

Next, the same polynomials are defined via `poly`. Note that `poly` fixes the ordering of the variables.

```
>> groebner::gbasis([poly(x^2 - y^2, [x, y]),
                     poly(x^2 + y, [x, y])], LexOrder)

                 2                      2
          [poly(x  + y, [x, y]), poly(y  + y, [x, y])]
```

Changing the ordering of the variables in `poly` changes the lexicographical ordering. This results in a different basis:

```
>> groebner::gbasis([poly(x^2 - y^2, [y, x]),
                     poly(x^2 + y, [y, x])], LexOrder)

                     2                  4    2
          [poly(y + x , [y, x]), poly(x  - x , [y, x])]
```

With *Reorder* the ordering of the variables may be changed internally:

```
>> groebner::gbasis([poly(x^2 - y^2, [x, y]),
                     poly(x^2 + y, [x, y])], LexOrder, Reorder)

                     2                  4    2
          [poly(y + x , [y, x]), poly(x  - x , [y, x])]
```

Background:

- The basis is computed via the Buchberger algorithm with the "shugar" selection strategy being used. For general informations, see T. Becker and V. Weispfenning: "Gröbner Bases", Springer (1993). For details on the shugar selection strategy, see A. Giovini, T. Mora, G. Niesi, L. Robbiano, C. Traverso: "One sugar cube, please — or Selection strategies in the Buchberger algorithm", Proc. ISSAC '91, Bonn, 49-54 (1991).

Changes:

- Further term orderings from `Dom::MonomOrdering` are handled. The special term orderings from the Gröbner package were moved there.

`groebner::normalf` – complete reduction modulo a polynomial ideal

`groebner::normalf(p, polys)` computes a normal form of the polynomial `p` by complete reduction modulo all polynomials in the list `polys`.

Call(s):

- `groebner::normalf(p, polys <, order>)`

Parameters:

`p`	—	a polynomial or a polynomial expression
`polys`	—	a list of polynomials of the same type as `p`. In particular, `polys` must be a list of polynomial expressions with rational coefficients, if `p` is a polynomial expression.
`order`	—	one of the identifiers *`DegInvLexOrder`*, *`DegreeOrder`*, and *`LexOrder`*, or a user-defined term ordering of type `Dom::MonomOrdering`. The default ordering is *`DegInvLexOrder`*.

Return Value: a polynomial of the same type as the input polynomials. If polynomial expressions are used as input, then a polynomial expression is returned.

Related Functions: `groebner::gbasis`, `poly`

Details:

- The rules laid down in the introduction to the `groebner` package concerning the polynomial types and the ordering apply.

- The polynomials in the list `polys` must all be of the same type as `p`. In particular, do not mix polynomials created via `poly` and polynomial expressions!

Example 1. We consider the ideal generated by the following polynomials:

```
>> p1 := poly(x^2 - x + 2*y^2, [x,y]):
   p2 := poly(x + 2*y - 1, [x,y]):
```

We compute the normal form of the following polynomial `p` modulo the ideal generated by `p1`, `p2` with respect to lexicographical ordering:

```
>> p :=  poly(x^2*y - 2*x*y + 1, [x,y]):
   groebner::normalf(p, [p1, p2], LexOrder);
                          3      2
                poly(- 2 y  + 2 y  - y + 1, [x, y])
```

Note that `p1`, `p2` do not form a Gröbner basis. The corresponding Gröbner basis leads to a different normal form of `p`:

```
>> groebner::normalf(p, groebner::gbasis([p1, p2]), LexOrder)
                      poly(- 5/9 y + 1, [x, y])
>> delete p1, p2, p:
```

Background:

- A polynomial g is a reduced form of a polynomial p modulo a list of polynomials $p_1, \ldots, p_n$, if $g \equiv p$ and none of the leading terms of the p_i divides the leading term of p, or if — for some i — g is a reduced form of $p - q\,p_i$, where q is the quotient of the leading monomial of p and the leading monomial of p_i. A reduced form always exists, but need not be unique. It is unique, if the p_i form a Gröbner basis.
- In the implementation of `groebner::normalf`, reduction modulo some p_i of largest possible total degree is preferred, if reduction modulo several p_i is possible.

Changes:

- Further term orderings from `Dom::MonomOrdering` are handled. The special term orderings from the Gröbner package were moved there.

`groebner::spoly` – the S-polynomial of two polynomials

`groebner::spoly(p1, p2)` computes the S-polynomial of the polynomials `p1` and `p2`.

Call(s):

- `groebner::spoly(p1, p2 <, order>)`

Parameters:

`p1, p2` — polynomials of the same type or polynomial expressions with rational coefficients

`order` — one of the identifiers *`DegInvLexOrder`*, *`DegreeOrder`*, and *`LexOrder`*, or a user-defined term ordering of type `Dom::MonomOrdering`. The default ordering is *`DegInvLexOrder`*.

Return Value: a polynomial of the same type as the input polynomials. If polynomial expressions are used as input, then a polynomial expression is returned.

Related Functions: `poly`

Details:

- The rules laid down in the introduction to the `groebner` package concerning the polynomial types and the ordering apply.
- The polynomials must be of the same type. In particular, do not mix polynomials created via `poly` and polynomial expressions!

Example 1. The polynomials

```
>> p1 := poly(x^2 - x + 2*y^2, [x, y]):
   p2 := poly(x + 2*y - 1, [x, y]):
```

generate the following S-polynomial with respect to lexicographical ordering:

```
>> groebner::spoly(p1, p2, LexOrder)
                                    2
                  poly(- 2 x y + 2 y , [x, y])
```

```
>> delete p1, p2:
```

Background:

- The S-polynomial of two polynomials p_1, p_2 is defined to be

$$\frac{\mathrm{lcm}(\mathrm{lterm}(p_1), \mathrm{lterm}(p_2))}{\mathrm{lmonomial}(p_1)} p_1 - \frac{\mathrm{lcm}(\mathrm{lterm}(p_1), \mathrm{lterm}(p_2))}{\mathrm{lmonomial}(p_2)} p_2,$$

  where `lterm` and `lmonomial` are used in the same sense as the MuPAD functions of the same name. This formula is constructed such that the leading terms of the two summands cancel.

Changes:

- Further term orderings from `Dom::MonomOrdering` are handled. The special term orderings from the Gröbner package were moved there.