MAT 401 - Seminar in Mathematics

(Elementary) Algebraic Geometry from an algorithmic point of view

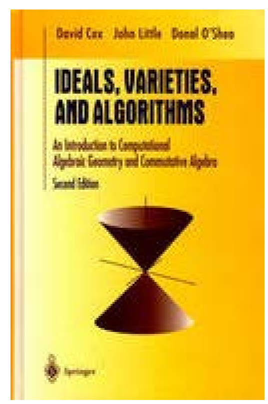
Instructor Sorin Popescu (office: Math 4-119, tel. 632-8358, e-mail sorin@math.sunysb.edu)
Location MAT S-235, TuTh 12:50-2:10

Prerequisites

A good foundation in linear algebra and some knowledge of abstract algebra (such as MAT 313 or MAT 311 or MAT 312). However, I will try to keep the amount of required previous knowledge to a minimum. The seminar will require active student participation and will encourage student discoveries of known (and perhaps unknown) mathematics.

Textbook(s)

Ideals, Varieties, and Algorithms (An Introduction to Computational Algebraic Geometry and Commutative Algebra), by D. Cox, J. Little, D. O'Shea, Springer Undergraduate Texts, Second Edition 1997



This book is a gentle introduction to computational algebraic geometry and commutative algebra at the undergraduate level. It discusses systems of polynomial equations ("ideals"), their solutions ("varieties"), and how these objects can be manipulated ("algorithms"). The Table of Contents may give you a more detailed picture of the topics covered in the book.

Other recommended texts:

- *Undergraduate Algebraic Geometry*, Miles Reid, LMS Student Texts **12**, Cambridge University Press 1989
- *Undergraduate Commutative Algebra*, Miles Reid, LMS Student Texts **29**, Cambridge University Press 1996

I have placed the above books on reserve.

Seminar plan

The seminar will address the following topics:

- Algebra and geometry:

- Affine varieties (particularly curves in $\mathbf{R}^2$ and curves and surfaces in $\mathbf{R}^3$), parametrizations, ideals.
  - Monomial orderings, division algorithm, Gröbner bases and basic properties. The Buchberger algorithm.
  - Applications of Gröbner bases: elimination theory, singular points, envelope of a family of curves, etc.
  - Some theory on varieties and ideals (an algebra-geometry dictionary)
  - Robotics. Integer programming.
  - Introduction to projective algebraic geometry.
- Use of computer packages:

  Specialized computer algebra systems as well as vizualization tools can help tremendously in the study of certain explicitly given (i.e. by equations) algebraic varieties. We will make use of the following software:
  - Maple and Mathematica for the visualization of implicitly defined curves, surfaces, etc.
  - Macaulay 2 and Macaulay for Gröbner basis calculations, and A local version of the Macaulay 2 manual is available here.

Homework

I will assign problems in each lecture, ranging in difficulty from routine to more challenging. Course grades will be based on these problems and any other class participation; solving at least 2/3 of them will be considered a perfect score. **Late homework** will be accepted until the second class meeting after the due date, but **will not** be accepted after that time. Each student will also be required to deliver a 20-30 minute (depending on class size) presentation and hand in one-two papers or computer projects (the first is due by November 1; the second by December 1)

Software documentation, tutorials, and computer projects

We will use the math computer lab in **S-235** of the math tower; this lab contains 30 Sun workstations running Unix, as well as a number of PCs running Windows NT. We will be using the Unix machines in class; however, much of the work can be done on other systems. We will rely heavily on Macaulay 2 (a software system devoted to supporting computations in algebraic geometry and commutative algebra) and Maple (a program that can do algebra, calculus, graphics, etc.), although if other tools are better suited to the task, we may make use of them. No previous experience with computers is needed.

Macaulay 2 and Maple are available for most platforms (Unix, Macintosh, Windows, etc); Macaulay 2 can be freely downloaded from the following location, while a student version of Maple can be purchased from Waterloo Maple for $99. You can also use the campus modem pool to dial-in to the mathlab computers.

Here are some important things to read:

- To access the documentation for Macaulay 2 you will need to use a web browser (which you must already be doing if you are reading this). I like the Netscape Navigator, (or its other version, Mozilla, but any other browser such as the Microsoft Internet Explorer is OK if you want. You can get Netscape and MSIE from Instructional Computing, although more recent versions are available from Netscape and Micro$oft.

- We may sometimes have to access documents on the web which are in the Adobe PDF form (Portable Document Format). To read these you will need to have a copy of the freely available Adobe Acrobat Reader on your machine. You can get this from Adobe. For PostScript documents, a free viewer can be downloaded from the GhostScript site.
- Since we will be doing most of our work (in class at least) on a Unix system, you should look over Using Unix. Also useful are the UNIXhelp tutorial, and sections of UNIX is a four-letter word.
- We will use Macaulay 2 in Emacs mode, but we will essentially make no other use of Emacs fancy features. Here is a short Emacs tutorial and here is an Emacs Quick Reference file.
- The simplest way to start Macaulay 2 is to type **M2** at the shell prompt. This will run a Macaulay 2 session in the current window. However the recommended way is to run Macaulay 2 as an emacs subprocess, one nice feature of the emacs' Macaulay 2 mode being command completion. Click here for a brief tutorial introduction to the use of emacs with Macaulay 2.

On pages 510 and 512 of the second edition of *Ideals, Varieties, and Algorithms*, Appendix C mentions computer packages for Maple and Mathematica. These were written mainly for teaching purposes and tend to be rather slow in comparison with a dedicated system as Macaulay 2. You may browse/download these packages from David A. Cox's web site. For Maple V, Release 5(1) you may also download the package, a tutorial and a reference worksheet from the following location:

- Package for Release 5.1 of Maple V
- Tutorial Worksheet for the Package
- Reference Worksheet for the Package

- Maple Package for Maple 6
- Tutorial Worksheet for the Package (Maple 6)
- Reference Worksheet for the Package (Maple 6)

Macaulay 2 is a software system devoted to supporting research in algebraic geometry and commutative algebra, developed by Daniel R. Grayson and Michael E. Stillman. Here are two elementary tutorials:

- A short tutorial introducing a number of basic operations using Groebner bases, and at the same time a range of useful Macaulay 2 constructs.
- A chapter (click here for PDF) by Bernd Sturmfels on elementary computations in Algebraic Geometry from a forthcoming book on Macaulay 2. It illustrates also the use of Macaulay 2 for some of the computations in the textbook by Cox, Little and O'Shea.

Worksheets, handouts, and other class related materials:

Download class related materials from the following links:

- August 30, 2001
- September 18, 2001
- September 25, 2001
- October 2, 2001
- October 9, 2001
- October 11, 2001
- October 16, 2001
- November 1, 2001

Links

Here are some links related to (elementary) uses of Gröbner bases:

- At the Geometry Center at the University of Minnesota, there is a lab on the nephroid (a kidney-shaped curve with some interesting mathematical properties) You may browse the entire lab, starting at the introduction, or read just the part which makes use of Gröbner bases (see also Projections, Profiles, and Envelopes and Singular Sets of Algebraic Curves and Surfaces).
- The Computer Algebra Information Network (based in Europe) has a lot of interesting material on Gröbner bases. For instance take a look at the various tutorials.

Special needs

If you have a physical, psychiatric, medical or learning disability that may impact on your ability to carry out assigned course work, you may contact the Disabled Student Services (DSS) office (Humanities 133, 632-6748/TDD). DSS will review your concerns and determine, with you, what accommodations may be necessary and appropriate. I will take their findings into account in deciding what alterations in course work you require. All information on and documentation of a disability condition should be supplied to me in writing at the earliest possible time AND is strictly confidential. Please act early, since I will not be able to make any retroactive course changes.

Sorin Popescu

2001-08-17

Sorin Popescu

Department of Mathematics
Stony Brook University
Stony Brook, NY 11794-3651

email: sorin@math.sunysb.edu
Office: Math 3-109
Phone: (631)-632-8255
Fax: (631)-632-7631

Research Interests: Algebraic Geometry, Commutative Algebra, Combinatorics and Computational methods

Teaching:

Spring 2006	MAT 311 Number Theory	MAT 614 Topics in Algebraic Geometry
Previous years	Teaching Archive	

Algebra, Geometry and Physics seminar: Spring 2006

Publications & E-Prints: Unless otherwise indicated, the files below are DVI files (), PostScript files (), PDF files (), or tar gziped DVI and PostScript files (). Files marked as () or () are hyperlinked PDF or Macromedia Flash files formated for screen viewing. Other formats (source, PS using Type I fonts) can be obtained via the UC Davis Front to the Mathematics ArXiv. Click on () or () for related *Macaulay2*, or *Macaulay* code.

Syzygies:

- *Gale Duality and Free Resolutions of Ideals of Points* [], [] [] [] [], *Invent math* **136** (1999) 2, 419-449
  David Eisenbud and Sorin Popescu
- *The Projective Geometry of the Gale Transform* [], [] [] [], *J. Algebra* **230** (2000), no. 1, 127-173

  David Eisenbud and Sorin Popescu
  (in the D. Buchsbaum anniversary volume of *J. Algebra*)
- *Syzygy Ideals for Determinantal Ideals and the Syzygetic Castelnuovo Lemma* [] [], [MathSci],

Springer 1999
David Eisenbud and Sorin Popescu

- *Extremal Betti Numbers and Applications to Monomial Ideals* [ ] [ ] [ ] [ ], *J. Algebra* **221** (1999), no. 2, 497-512
  Dave Bayer, Hara Charalambous and Sorin Popescu
- *Lagrangian Subbundles and Codimension 3 Subcanonical Subschemes* [ ], [ ] [ ] [ ], *Duke Math. J.* **107** (2001), no. 3, 427-467
  David Eisenbud, Sorin Popescu and Charles Walter
- *Enriques Surfaces and other Nonpfaffian Codimension 3 Subcanonical Subschemes* [ ] [ ] [ ] [ ], *Comm. Algebra* **28** (2000), 5629-5653
  David Eisenbud, Sorin Popescu and Charles Walter
  (in the Hartshorne anniversary volume of *Comm. Algebra*)
- *Syzygies of Unimodular Lawrence Ideals* [ ] [ ] [ ] [ ], *J. Reine Angew. Math* **534** (2001), 169-186
  Dave Bayer, Sorin Popescu and Bernd Sturmfels
- *Hyperplane Arrangement Cohomology and Monomials in the Exterior Algebra* [ ] [ ] [ ] [ ] [ ], Trans. AMS. **355** (2003), 4365-4383
  David Eisenbud, Sorin Popescu and Sergey Yuzvinsky
- *Exterior algebra methods for the Minimal Resolution Conjecture* [ ] [ ] [ ] [ ], *Duke Math. J.* **112** (2002), no. 2, 379-395
  David Eisenbud, Frank-Olaf Schreyer, Sorin Popescu and Charles Walter
- *Symmetric resolutions of coherent sheaves* [ ] [ ] [ ]
  David Eisenbud, Sorin Popescu and Charles Walter
- *A note on the Intersection of Veronese Surfaces* [ ] [ ] [ ] [ ] [ ]
  David Eisenbud, Klaus Hulek and Sorin Popescu
- *Restricting linear syzygies: algebra and geometry* [ ] [ ] [ ] [ ] [ ], *Compositio Math.* **141** (2005), no.6, 1460-1478
  David Eisenbud, Mark Green, Klaus Hulek and Sorin Popescu
- *Small schemes and varieties of minimal degree* [ ] [ ] [ ] [ ] [ ], *Amer. J of Math* (2005), to appear
  David Eisenbud, Mark Green, Klaus Hulek and Sorin Popescu

Abelian varieties, modular varieties and equations:

- *Equations of (1,d)-polarized abelian surfaces* [ ] [ ] [ ], *Math. Ann.* **310** (1998), no. 2, 333-377
  Mark Gross and Sorin Popescu
- *The moduli space of (1,11)-polarized abelian surfaces is unirational* [ ] [ ] [ ], *Compositio Math.* **126** (2001), no. 1, 1-24
  Mark Gross and Sorin Popescu
- *Calabi-Yau threefolds and moduli of abelian surfaces I* [ ] [ ] [ ], *Compositio Math.* **127**, no. 2, (2001), 169-228
  Mark Gross and Sorin Popescu

Calabi-Yau threefolds and moduli of abelian surfaces II [ ] [ ] [ ]
Mark Gross and Sorin Popescu

- *Elliptic functions and equations of modular curves* [ ] [ ] [ ] [ ], *Math. Ann.* **321** (2001), no. 3, 553-568
  Lev A. Borisov, Paul Gunnells, and Sorin Popescu

Surfaces in P^4 and threefolds in P^5:

- *The Geometry of Bielliptic Surfaces in P^4* [ ], [ ] [ ], *Internat. J. Math.* **4** (1993), no. 6, 873-902
  A. Aure, W. Decker, K. Hulek, S. Popescu and K. Ranestad
- *On Surfaces in P^4 and Threefolds in P^5* [ ] [ ] [ ], [MathSci], LMSLN **208**, 69--100
  W. Decker and S. Popescu
- *Surfaces of degree 10 in P^4 via linear systems and linkage* [ ] [ ] [ ] [ ] [ ], *J. Algebraic Geom.* **5** (1996), no. 1, 13-76
  S. Popescu and K. Ranestad
- *Syzygies of Abelian and Bielliptic Surfaces in P^4* [ ] [ ] [ ], *Internat. J. Math.* **8** (1997), no. 7, 849-919
  A. Aure, W. Decker, K. Hulek, S. Popescu and K. Ranestad
- *Examples of smooth non general type surfaces in P^4* [ ] [ ] [ ] [ ] [ ], *Proc. London Math. Soc.* (3) **76** (1998), no. 2, 257-275
  S. Popescu
- *Surfaces of degree >= 11 in the Projective Fourspace* [ ] [ ] [ ]+ *Appendix* [ ] [ ] [ ]
  S. Popescu

PRAGMATIC 1997: A summer school in Catania, Sicily

- *Research Problems for the summer school* [ ], [ ] [ ], [MathSci], *Matematiche* (Catania) **53** (1998), 1-14
  David Eisenbud and Sorin Popescu

Algorithmic Algebra and Geometry: Summer Graduate Program (1998) at MSRI:

- Poster [ ] [ ], lecture slides and streaming video , CD ROM,
  Dave Bayer and Sorin Popescu

Linear algebra notes

- *On circulant matrices* [ ], [ ] [ ] [ ] [ ],
  Daryl Geller, Irwin Kra, Sorin Popescu and Santiago Simanca

---

Upcoming conferences:

- DARPA FunBio Mathematics-Biology Kick-off meeting, Princeton, September 21-23, 2005
- MAGIC 05: Midwest Algebra, Geometry and their Interactions Conference, University of Notre Dame, Notre Dame, October 7-11, 2005
- AMS Special Session on Resolutions, Eugene, OR, November 12-13, 2005
- Clay Workshop on Algebraic Statistics and Computational Biology, Clay Mathematics Institute, November 12-14, 2005
- CIMPA School on Commutative Algebra, December 26, 2005 - January 6, 2006, Hanoi, Vietnam
- AMS Special Session on Syzygies in Commutative Algebra and Geometry, San Antonio, TX, January 12-15, 2006
- KAIST Workshop on Projective Algebraic Geometry, January 23-25, 2006, Korean Advanced Institute of Science and Technology, Daejeon
- AMS Special Session on the Geometry of Groebner bases, San Francisco, CA, April 29-30, 2006
- Castenuovo-Mumford regularity and related topics, Workshop at CIRM, Luminy, France, May 9-13, 2006
- Commutative Algebra and its Interaction with Algebraic Geometry, Workshop at CIRM, Luminy, France, May 22-26, 2006
- Syzygies and Hilbert Functions, Banff International Research Meeting, Canada, October 14-19, 2006

Past conferences:

- A conference on alegbraic geometry to celebrate Robin Hartshorne's 60th birthday, Berkeley, August 28-30, 1998
- Western Algebraic Geometry Seminar, MSRI, Berkeley, December 5-6, 1998
- Conference on Groebner Bases, Guanajato, Mexico, February 8-12, 1999
- The Pacific Northwest Geometry Seminar
- Computational Commutative Algebra and Combinatorics, Osaka, July 21-30, 1999. 
- Kommutative Algebra und Algebraische Geometrie, Oberwolfach, August 8-14, 1999. 
- AMS Western Section Meeting Salt Lake City, UT, September 25-26, 1999.
- Algebra and Geometry of Points in Projective Space, Napoli, February 9-12, 2000.
- AMS Spring Eastern Sectional Meeting Lowell, MA, April 1-2, 2000.
- Algèbre commutative et ses interactions avec la géométrie algébrique, Centre International de Rencontres Mathématiques, June 5-9, 2000.
- Topics in Classical Algebraic Geometry, Oberwolfach, June 18-24, 2000 
- AMS Fall Central Section Meeting Toronto, Ontario Canada, September 22-24, 2000
- AMS Fall Eastern Section Meeting, New York, Columbia U. in New York, November 4-5, 2000
- Exterior algebra methods and other new directions in Algebraic Geometry, Commutative Algebra and Combinatorics, 8-15 September 2001, Ettore Majorana Centre, Erice, Sicily, Italy. Photos from the conference.
- Classical Algebraic Geometry, Oberwolfach, May 26 - June 1, 2002 
- Current trends in Commutative Algebra, Levico, Trento, June 17-21, 2002
- Birational and Projective Geometry of Algebraic Varieties, Ferrara, September 2-8, 2002
- Commutative Algebra, Singularities and Computer Algebra, Sinaia, September 17-22, 2002. Photos from the conference.
- James H. Simons Conference on Quantum and Reversible Computation , Stony Brook, May 25-31, 2003

- Conference on Commutative Algebra, Lisbon, June 23-27 2003. Photos from the conference. Also photos from Belém.
- Commutative Algebra and Interactions with Algebraic Geometry and Combinatorics, ICTP, Trieste, June 6-11
- III Iberoamerican Congress on Geometry, Salamanca, June 7-12
- Projective Varieties: A Conference in honour of the 150th anniversary of the birth of G. Veronese, Siena, June 8-12 , 2004. Photos from the conference.
- Algebraic Geometry: conference in honour of Joseph Le Potier & Christian Peskine, Paris, June 15-18, 2004
- Classical Algebraic Geometry, Oberwolfach, June 27-July 3, 2004
- Combinatorial Commutative Algebra, Oberwolfach, July 4-10th, 2004

Next: Introduction

USING UNIX

Phil Boyland

Fall, 1992 (revised Fall, 1998)

Emacs Tutorial

NOTE: It is suggested that you start up Emacs before continuing. To bring up this tutorial in Emacs, hit the ESC key following by "x" then type help-with-tutorial followed by hitting the RETURN key.

Emacs commands generally involve the CONTROL key (sometimes labelled CTRL or CTL) or the META key (sometimes labelled EDIT). Rather than write out META or CONTROL each time we want you to prefix a character, we'll use the following abbreviations:

C-<chr> means hold the CONTROL key while typing the character <chr> Thus, C-f would be: hold the CONTROL key and type f.

M-<chr> means hold the META or EDIT key down while typing <chr>. If there is no META or EDIT key, type <ESC>, release it, then type the character <chr>. "<ESC>" stands for the key labelled "ALT" or "ESC".

Important note: to end the Emacs session, type C-x C-c. (Two characters.)

The characters ">>" at the left margin indicate directions for you to try using a command. For instance:

>> Now type C-v (View next screen) to move to the next screen. (go ahead, do it by depressing the control key and v together). From now on, you'll be expected to do this whenever you finish reading the screen. Note that there is an overlap when going from screen to screen; this provides some continuity when moving through the file.

The first thing that you need to know is how to move around from place to place in the file. You already know how to move forward a screen, with C-v. To move backwards a screen, type M-v (depress the META key and type v, or type v if you don't have a META or EDIT key). >> Try typing M-v and then C-v to move back and forth a few times.

Summary

The following commands are useful for viewing screenfuls:

```
        C-v     Move forward one screenful
        M-v     Move backward one screenful
        C-l     Clear screen and redisplay everything
                 putting the text near the cursor at the center.
                 (That's control-L, not control-1.)
```

>> Find the cursor and remember what text is near it. Then type a C-l. Find the cursor again and see what text is near it now.

Basic Cursor Control

Getting from screenful to screenful is useful, but how do you reposition yourself within a given screen to a specific place? There are several ways you can do this. One way (not the best, but the most basic) is to use the commands previous, backward, forward and next. As you can imagine these commands (which are given to Emacs as C-p, C-b, C-f, and C-n respectively) move the cursor from where it currently is to a new place in the given direction. Here, in a more graphical form are the commands:

```
                              Previous line, C-p
                                        :
                                        :
   Backward, C-b .... Current cursor position .... Forward, C-f
                                        :
                                        :
                              Next line, C-n
```

>> Move the cursor to the line in the middle of that diagram and type C-l to see the whole diagram centered in the screen.

You'll probably find it easy to think of these by letter. P for previous, N for next, B for backward and F for forward. These are the basic cursor positioning commands and you'll be using them ALL the time so it would be of great benefit if you learn them now.

>> Do a few C-n's to bring the cursor down to this line.

>> Move into the line with C-f's and then up with C-p's. See what C-p does when the cursor is in the middle of the line.

Lines are separated by Newline characters. For most applications there should normally be a Newline character at the end of the text, as well, but it is up to you to make sure of this. A file can validly exist without a Newline at the end.

>> Try to C-b at the beginning of a line. Do a few more C-b's. Then do C-f's back to the end of the line and beyond.

When you go off the top or bottom of the screen, the text beyond the edge is shifted onto the screen so that your instructions can be carried out while keeping the cursor on the screen.

>> Try to move the cursor off the bottom of the screen with C-n and see what happens.

If moving by characters is too slow, you can move by words. M-f (Meta-f) moves forward a word and M-b moves back a word.

>> Type a few M-f's and M-b's. Intersperse them with C-f's and C-b's.

Notice the parallel between C-f and C-b on the one hand, and M-f and M-b on the other hand. Very often Meta characters are used for operations related to English text whereas Control characters operate on the basic textual units that are independent of what you are editing (characters, lines, etc). There is a similar parallel between lines and sentences: C-a and C-e move to the beginning or end of a line, and M-a and M-e move to the beginning or end of a sentence.

>> Try a couple of C-a's, and then a couple of C-e's.
Try a couple of M-a's, and then a couple of M-e's.

See how repeated C-a's do nothing, but repeated M-a's keep moving farther. Do you think that this is right?

Two other simple cursor motion commands are M-< (Meta Less-than), which moves to the beginning of the file, and M-> (Meta Greater-than), which moves to the end of the file. You probably don't need to try them, since finding this spot again will be boring. On most terminals the "<" is above the comma and you must use the shift key to type it. On these terminals you must use the shift key to type M-< also; without the shift key, you would be typing M-comma. The location of the cursor in the text is also called "point". To paraphrase, the cursor shows on the screen where point is located in the text. Here is a summary of simple moving operations including the word and sentence moving commands:

```
C-f     Move forward a character
C-b     Move backward a character

M-f     Move forward a word
M-b     Move backward a word

C-n     Move to next line
C-p     Move to previous line

C-a     Move to beginning of line
C-e     Move to end of line

M-a     Move back to beginning of sentence
M-e     Move forward to end of sentence

M-<     Go to beginning of file
M->     Go to end of file
```

Like all other commands in Emacs, these commands can be given arguments which cause them to be executed repeatedly. The way you give a command a repeat count is by typing C-u and then the digits before you type the command. If you have a META or EDIT key, you can omit the C-u if you hold down the META or EDIT key while you type the digits. This is easier, but we recommend the C-u method because it works on any terminal.

For instance, C-u 8 C-f moves forward eight characters.

>> Try giving a suitable argument to C-n or C-p to come as close as you can to this line in one jump.

The only apparent exception to this is the screen moving commands, C-v and M-v. When given an argument, they scroll the screen up or down by that many lines, rather than screenfuls. This proves to be much more useful.

>> Try typing C-u 8 C-v now.

Did it scroll the screen up by 8 lines? If you would like to scroll it down you can give an argument to M-v.

If you are using X Windows, there is probably a rectangular area called a scroll bar at the right hand side of the Emacs window. You can scroll the text by clicking the mouse in the scroll bar.

>> Try pressing the middle button at the top of the highlighted area within the scroll bar, then moving the mouse while holding that button down.

>> Move the mouse to a point in the scroll bar about three lines from the top, and click the left button a couple of times. Then try the right button a couple of times.

When Emacs is hung

If Emacs gets into an infinite (or simply very long) computation which you don't want to finish, you can stop it safely by typing C-g. You can also use C-g to discard a numeric argument or the beginning of a command that you don't want to finish.

>> Type C-u 100 to make a numeric arg of 100, then type C-g. Now type C-f. How many characters does it move? If you have typed an <ESC> by mistake, you can get rid of it with a C-g.

If you type <ESC> <ESC>, you get a new window appearing on the screen, telling you that M-ESC is a "disabled command" and asking whether you really want to execute it. The command M-ESC is marked as disabled because you probably don't want to use it until you know more about Emacs, and we expect it would confuse you if it were allowed to go ahead and run. If you really want to try the M-ESC command, you could type a Space in answer to the question and M-ESC would go ahead. Normally, if you do not want to execute M-ESC, you would type "n" to answer the question.

>> Type <ESC> <ESC>, then type n.

Windows

Emacs can have several windows, each displaying its own text. At this stage it is better not to go into the techniques of using multiple windows. But you do need to know how to get rid of extra windows that may appear to display help or output from certain commands. It is simple:

C-x 1 One window (i.e., kill all other windows).

That is Control-x followed by the digit 1. C-x 1 makes the window which the cursor is in become the full screen, by getting rid of any other windows.

>> Move the cursor to this line and type C-u 0 C-l.
>> Type Control-h k Control-f. See how this window shrinks, while a new one appears to display documentation on the Control-f command.

>> Type C-x 1 and see the documentation listing window disappear.

Inserting and Deleting

If you want to insert text, just type it. Characters which you can see, such as A, 7, *, etc. are taken by Emacs as text and inserted immediately. Type <Return> (the carriage-return key) to insert a Newline character.

You can delete the last character you typed by typing <Rubout>. <Rubout> is a key on the keyboard, which might be labelled "Delete" instead of "Rubout" on some terminals. More generally, <Rubout> deletes the character immediately before the current cursor position.

>> Do this now, type a few characters and then delete them by typing <Rubout> a few times. Don't worry about this file being changed; you won't affect the master tutorial. This is just a copy of it.

>> Now start typing text until you reach the right margin, and keep typing. When a line of text gets too big for one line on the screen, the line of text is "continued" onto a second screen line. The backslash at the right margin indicates a line which has been continued.

>> Use <Rubout>s to delete the text until the line fits on one screen line again. The continuation line goes away.

>> Move the cursor to the beginning of a line and type <Rubout>. This deletes the newline before the line and merges the line onto the previous line. The resulting line may be too long to fit, in which case it has a continuation line.

>> Type <Return> to reinsert the Newline you deleted.

Remember that most Emacs commands can be given a repeat count; this includes characters which insert themselves.

>> Try that now -- type C-u 8 * and see what happens.

You've now learned the most basic way of typing something in Emacs and correcting errors. You can delete by words or lines as well. Here is a summary of the delete operations:

```
<Rubout>     delete the character just before the cursor
C-d          delete the next character after the cursor

M-<Rubout>   kill the word immediately before the cursor
M-d          kill the next word after the cursor

C-k          kill from the cursor position to end of line
M-k          kill to the end of the current sentence
```

Notice that <Rubout> and C-d vs M-<Rubout> and M-d extend the parallel started by C-f and M-f (well, <Rubout> isn't really a control character, but let's not worry about that). C-k and M-k are like C-e and M-e, sort of, in that lines are opposite sentences.

Now suppose you kill something, and then you decide that you want to get it back? Well, whenever you kill something bigger than a character, Emacs saves it for you. To yank it back, use C-y. You can kill text in one place, move elsewhere, and then do C-y; this is a good way to move text around. Note that the difference between "Killing" and "Deleting" something is that "Killed" things can be yanked back, and "Deleted" things cannot. Generally, the commands that can destroy a lot of text save it, while the ones that attack only one character, or nothing but blank lines and spaces, do not save.

For instance, type C-n a couple times to position the cursor at some line on this screen.

>> Do this now, move the cursor and kill that line with C-k.

Note that a single C-k kills the contents of the line, and a second C-k kills the line itself, and make all the other lines move up. If you give C-k a repeat count, it kills that many lines AND their contents.

The text that has just disappeared is saved so that you can retrieve it. To retrieve the last killed text and put it where the cursor currently is, type C-y.

>> Try it; type C-y to yank the text back.

Think of C-y as if you were yanking something back that someone took away from you. Notice that if you do several C-k's in a row the text that is killed is all saved together so that one C-y will yank all of the lines.

>> Do this now, type C-k several times.

Now to retrieve that killed text:

>> Type C-y. Then move the cursor down a few lines and type C-y again. You now see how to copy some text.

What do you do if you have some text you want to yank back, and then you kill something else? C-y would yank the more recent kill. But the previous text is not lost. You can get back to it using the M-y command. After you have done C-y to get the most recent kill, typing M-Y replaces that yanked text with the previous kill. Typing M-y again and again brings in earlier and earlier kills. When you have reached the text you are looking for, you can just go away and leave it there. If you M-y enough times, you come back to the starting point (the most recent kill).

>> Kill a line, move around, kill another line. Then do C-y to get back the second killed line. Then do M-y and it will be replaced by the first killed line. Do more M-y's and see what you get. Keep doing them until the second kill line comes back, and then a few more. If you like, you can try giving M-y positive and negative arguments.

Undo

Any time you make a change to the text and wish you had not done so, you can undo the change (return the text to its previous state) with the undo command, C-x u. Normally, C-x u undoes one command's worth of changes; if you repeat the C-x u several times in a row, each time undoes one more command. There are two exceptions: commands that made no change (just moved the cursor) do not count, and self-inserting characters are often lumped together in groups of up to 20. This is to reduce the number of C-x u's you have to type.

C-_ is another command for undoing; it is just the same as C-x u but easier to type several times in a row. The problem with C-_ is that on some keyboards it is not obvious how to type it. That is why C-x u is provided as well. On some DEC terminals, you can type C-_ by typing / while holding down CTRL. Illogical, but what can you expect from DEC?

Giving a numeric argument to C-_ or C-x u is equivalent to repeating it as many times as the argument says.

Files

In order to make the text you edit permanent, you must put it in a file. Otherwise, it will go away when your invocation of Emacs goes away. You put your editing in a file by "finding" the file. What finding means is that you see the contents of the file in your Emacs; and, loosely speaking, what you are editing is the file itself. However, the changes still don't become permanent until you "save" the file. This is so you can have control to avoid leaving a half-changed file around when you don't want to. Even then, Emacs leaves the original file under a changed name in case your changes turn out to be a mistake.

If you look near the bottom of the screen you will see a line that begins and ends with dashes, and contains the string "Emacs: TUTORIAL". Your copy of the Emacs tutorial is called "TUTORIAL". Whatever file you find, that file's name will appear in that precise spot.

The commands for finding and saving files are unlike the other commands you have learned in that they consist of two characters. They both start with the character Control-x. There is a whole series of commands that start with Control-x; many of them have to do with files, buffers, and related things, and all of them consist of Control-x followed by some other character.

Another thing about the command for finding a file is that you have to say what file name you want. We say the command "reads an argument from the terminal" (in this case, the argument is the name of the file). After you type the command

C-x C-f Find a file

Emacs asks you to type the file name. It echoes on the bottom line of the screen. You are using the minibuffer now! this is what the minibuffer is for. When you type <Return> to end the file name, the minibuffer is no longer needed, so it disappears.

>> Type C-x C-f, then type C-g. This cancels the minibuffer, and also cancels the C-x C-f command that was using the minibuffer. So you do not find any file.

In a little while the file contents appear on the screen. You can edit the contents. When you wish to make the changes permanent, issue the command

C-x C-s Save the file

The contents of Emacs are written into the file. The first time you do this, the original file is renamed to a new name so that it is not lost. The new name is made by appending "~" to the end of the original file's name.

When saving is finished, Emacs prints the name of the file written. You should save fairly often, so that you will not lose very much work if the system should crash.

>> Type C-x C-s, saving your copy of the tutorial. This should print "Wrote .../TUTORIAL" at the bottom of the screen. On VMS it will print "Wrote ...[...]TUTORIAL."

To make a new file, just find it "as if" it already existed. Then start typing in the text. When you ask to "save" the file, Emacs will really create the file with the text that you have inserted. From then on, you can consider yourself to be editing an already existing file.

Buffers

If you find a second file with C-x C-f, the first file remains inside Emacs. You can switch back to it by finding it again with C-x C-f. This way you can get quite a number of files inside Emacs.

The object inside Emacs which holds the text read from one file is called a "buffer." Finding a file makes a new buffer inside Emacs. To see a list of the buffers that exist in Emacs, type

C-x C-b List buffers

>> Try C-x C-b now.

See how each buffer has a name, and it may also have a file name for the file whose contents it holds. Some buffers do not correspond to files. For example, the buffer named "*Buffer List*" does not have any file. It is the buffer which contains the buffer list that was made by C-x C-b. ANY text you see in an Emacs window has to be in some buffer.

>> Type C-x 1 to get rid of the buffer list.

If you make changes to the text of one file, then find another file, this does not save the first file. Its changes remain inside Emacs, in that file's buffer. The creation or editing of the second file's buffer has no effect on the first file's buffer. This is very useful, but it also means that you need a convenient way to save the first file's buffer. It would be a nuisance to have to switch back to it with C-x C-f in order to save it with C-x C-s. So we have

C-x s Save some buffers

C-x s goes through the list of all the buffers you have and finds the ones that contain files you have changed. For each such buffer, C-x s asks you whether to save it.

Extending the Command Set

There are many, many more Emacs commands than could possibly be put on all the control and meta characters. Emacs gets around this with the X (eXtend) command. This comes in two flavors:

```
        C-x     Character eXtend.  Followed by one character.
        M-x     Named command eXtend.  Followed by a long name.
```

These are commands that are generally useful but used less than the commands you have already learned about. You have already seen two of them: the file commands C-x C-f to Find and C-x C-s to Save. Another example is the command to tell Emacs that you'd like to stop editing and get rid of Emacs. The command to do this is C-x C-c. (Don't worry; it offers to save each changed file before it kills the Emacs.)

C-z is the usual way to exit Emacs, because it is always better not to kill the Emacs if you are going to do any more editing. On systems which allow it, C-z exits from Emacs to the shell but does not destroy the Emacs; if you use the C shell, you can resume Emacs with the `fg' command (or, more generally, with `%emacs', which works even if your most recent job was some other). On systems where suspending is not possible, C-z creates a subshell running under Emacs to give you the chance to run other programs and return to Emacs afterward, but it does not truly "exit" from Emacs. In this case, the shell command `exit' is the usual way to get back to Emacs from the subshell.

You would use C-x C-c if you were about to log out. You would also use it to exit an Emacs invoked under mail handling programs and other random utilities, since they may not believe you have really finished using the Emacs if it continues to exist.

There are many C-x commands. The ones you know are:

```
        C-x C-f         Find file.
        C-x C-s         Save file.
        C-x C-b         List buffers.
        C-x C-c         Quit Emacs.
        C-x u           Undo.
```

Named eXtended commands are commands which are used even less frequently, or commands which are used only in certain modes. These commands are usually called "functions". An example is the function replace-string, which globally replaces one string with another. When you type M-x, Emacs prompts you at the bottom of the screen with M-x and you should type the name of the function you wish to call; in this case, "replace-string". Just type "repl s<TAB>" and Emacs will complete the name. End the command name with <Return>. Then type the two "arguments"--the string to be replaced, and the string to replace it with--each one ended with a Return.

>> Move the cursor to the blank line two lines below this one. Then type M-x repl s<Return>changed<Return>altered<Return>.

Notice how this line has changed: you've replaced the word c-h-a-n-g-e-d with "altered" wherever it occurred after the cursor.

Mode Line

If Emacs sees that you are typing commands slowly it shows them to you at the bottom of the screen in an area called the "echo area." The echo area contains the bottom line of the screen. The line immediately above it is called the MODE LINE. The mode line says something like

```
--**-Emacs: TUTORIAL          (Fundamental)--58%----------------------
```

This is a very useful "information" line.

You already know what the filename means--it is the file you have found. What the --NN%-- means is that NN percent of the file is above the top of the screen. If the top of the file is on the screen, it will say --Top-- instead of --00%--. If the bottom of the file is on the screen, it will say --Bot--. If you are looking at a file so small it all fits on the screen, it says --All--.

The stars near the front mean that you have made changes to the text. Right after you visit or save a file, there are no stars, just dashes.

The part of the mode line inside the parentheses is to tell you what modes you are in. The default mode is Fundamental which is what you are in now. It is an example of a "major mode". There are several major modes in Emacs for editing different languages and text, such as Lisp mode, Text mode, etc. At any time one and only one major mode is active, and its name can always be found in the mode line just where "Fundamental" is now. Each major mode makes a few commands behave differently. For example, there are commands for creating comments in a program, and since each programming language has a different idea of what a comment should look like, each major mode has to insert comments differently. Each major mode is the name of an extended command, which is how you get into the mode. For example, M-x fundamental-mode is how to get into Fundamental mode.

If you are going to be editing English text, such as this file, you should probably use Text Mode.

>> Type M-x text-mode<Return>.

Don't worry, none of the commands you have learned changes Emacs in any great way. But you can observe that apostrophes are now part of words when you do M-f or M-b. Major modes are usually like that: commands don't change into completely unrelated things, but they work a little bit differently.

To get documentation on your current major mode, type C-h m.

>> Use C-u C-v once or more to bring this line near the top of screen.
>> Type C-h m, to see how Text mode differs from Fundamental mode.
>> Type C-x 1 to remove the documentation from the screen.

Major modes are called major because there are also minor modes. They are called minor because they aren't alternatives to the major modes, just minor modifications of them. Each minor mode can be turned on or off by itself, regardless of what major mode you are in, and regardless of the other minor modes. So you can use no minor modes, or one minor mode, or any combination of several minor modes.

One minor mode which is very useful, especially for editing English text, is Auto Fill mode.

When this mode is on, Emacs breaks the line in between words automatically whenever the line gets too long. You can turn this mode on by doing M-x auto-fill-mode<Return>. When the mode is on, you can turn it off by doing M-x auto-fill-mode<Return>. If the mode is off, this function turns it on, and if the mode is on, this function turns it off. This is called "toggling".

>> Type M-x auto-fill-mode<Return> now. Then insert a line of "asdf " over again until you see it divide into two lines. You must put in spaces between them because Auto Fill breaks lines only at spaces.

The margin is usually set at 70 characters, but you can change it with the C-x f command. You should give the margin setting you want as a numeric argument.

>> Type C-x f with an argument of 20. (C-u 2 0 C-x f). Then type in some text and see Emacs fill lines of 20 characters with it. Then set the margin back to 70 using C-x f again.

If you makes changes in the middle of a paragraph, Auto Fill mode does not re-fill it for you. To re-fill the paragraph, type M-q (Meta-q) with the cursor inside that paragraph.

>> Move the cursor into the previous paragraph and type M-q.

Searching

Emacs can do searches for strings (these are groups of contiguous characters or words) either forward through the file or backward through it. To search for the string means that you are trying to locate it somewhere in the file and have Emacs show you where the occurrences of the string exist. This type of search is somewhat different from what you may be familiar with. It is a search that is performed as you type in the thing to search for. The command to initiate a search is C-s for forward search, and C-r for reverse search. BUT WAIT! Don't do them now. When you type C-s you'll notice that the string "I-search" appears as a prompt in the echo area. This tells you that Emacs is in what is called an incremental search waiting for you to type the thing that you want to search for. <RET> terminates a search.

>> Now type C-s to start a search. SLOWLY, one letter at a time, type the word 'cursor', pausing after you type each character to notice what happens to the cursor.
>> Type C-s to find the next occurrence of "cursor".
>> Now type <Rubout> four times and see how the cursor moves.
>> Type <RET> to terminate the search.

Did you see what happened? Emacs, in an incremental search, tries to go to the occurrence of the string that you've typed out so far. To go to the next occurrence of 'cursor' just type C-s again. If no such occurrence exists Emacs beeps and tells you that it is a failing search. C-g would also terminate the search.

If you are in the middle of an incremental search and type <Rubout>, you'll notice that the last character in the search string is erased and the search backs up to the last place of the search. For instance, suppose you currently have typed 'cu' and you see that your cursor is at the first occurrence of 'cu'. If you now type <Rubout>, the 'u' on the search line is erased and you'll be repositioned in the text to the occurrence of 'c' where the search took you before you typed the 'u'. This provides a useful means for backing up while you are searching.

If you are in the middle of a search and type a control or meta character (with a few exceptions--characters that are special in a search, such as C-s and C-r), the search is terminated.

The C-s starts a search that looks for any occurrence of the search string AFTER the current cursor position. But what if you want to search for something earlier in the text? To do this, type C-r for Reverse search. Everything that applies to C-s applies to C-r except that the direction of the search is reversed.

Multiple Windows

One of the nice features of Emacs is that you can display more than one window on the screen at the same time.

>> Move the cursor to this line and type C-u 0 C-l.

>> Now type C-x 2 which splits the screen into two windows. Both windows display this tutorial. The cursor stays in the top window.

>> Type C-M-v to scroll the bottom window.

>> Type C-x o ("o" for "other") to move the cursor to the bottom window.

>> Use C-v and M-v in the bottom window to scroll it. Keep reading these directions in the top window.

>> Type C-x o again to move the cursor back to the top window. The cursor is still just where it was in the top window before.

You can keep using C-x o to switch between the windows. Each window has its own cursor position, but only one window actually shows the cursor. All the ordinary editing commands apply to the window that the cursor is in.

The command C-M-v is very useful when you are editing text in one window and using the other window just for reference. You can keep the cursor always in the window where you are editing, and edit there as you advance through the other window.

>> Type C-x 1 (in the top window) to get rid of the bottom window.

(If you had typed C-x 1 in the bottom window, that would get rid of the top one. Think of this command as "Keep just one window--the window I am already in.")

You don't have to display the same buffer in both windows. If you use C-x C-f to find a file in one window, the other window doesn't change. You can pick a file in each window independently.

Here is another way to use two windows to display two different things:

>> Type C-x 4 C-f followed by the name of one of your files. End with <RETURN>. See the specified file appear in the bottom window. The cursor goes there, too.

>> Type C-x o to go back to the top window, and C-x 1 to delete the bottom window.

Recursive Editing Levels

Sometimes you will get into what is called a "recursive editing level". This is indicated by square brackets in the mode line, surrounding the parentheses around the major mode name. For example, you might see [(Fundamental)] instead of (Fundamental).

To get out of the recursive editing level, type

M-x top-level<Return>.

>> Try that now; it should display "Back to top level" at the bottom of the screen.

In fact, you were ALREADY at top level (not inside a recursive editing level) if you have obeyed instructions. M-x top-level does not care; it gets out of any number of recursive editing levels, perhaps zero, to get back to top level.

You can't use C-g to get out of a recursive editing level because C-g is used for discarding numeric arguments and partially typed commands WITHIN the recursive editing level.

Getting More Help

In this tutorial we have tried to supply just enough information to get you started using Emacs. There is so much available in Emacs that it would be impossible to explain it all here. However, you may want to learn more about Emacs since it has numerous desirable features that you don't know about yet. Emacs has a great deal of internal documentation. All of these commands can be accessed through the character Control-h, which we call "the Help character" because of the function it serves.

To use the HELP features, type the C-h character, and then a character saying what kind of help you want. If you are REALLY lost, type C-h ? and Emacs will tell you what kinds of help it can give. If you have typed C-h and decide you don't want any help, just type C-g to cancel it.

The most basic HELP feature is C-h c. Type C-h, a c, and a command character or sequence, and Emacs displays a very brief description of the command.

>> Type C-h c Control-p. The message should be something like C-p runs the command previous-line

This tells you the "name of the function". That is important in writing Lisp code to extend

Emacs; it also is enough to remind you of what the command does if you have seen it before but did not remember.

Multi-character commands such as C-x C-s and (if you have no META or EDIT key) <ESC>v are also allowed after C-h c.

To get more information on the command, use C-h k instead of C-h c.

>> Type C-h k Control-p.

This displays the documentation of the function, as well as its name, in an Emacs window. When you are finished reading the output, type C-x 1 to get rid of the help text. You do not have to do this right away. You can do some editing while referring to the help text and then type C-x 1.

Here are some other useful C-h options:

```
   C-h f        Describe a function.  You type in the name of the
                function.


   C-h a        Command Apropos.  Type in a keyword and Emacs will list
                all the commands whose names contain that keyword.
                These commands can all be invoked with Meta-x.
                For some commands, Command Apropos will also list a one
                or two character sequence which has the same effect.
```

>> Type C-h a file<Return>.

This displays in another window a list of all M-x commands with "file" in their names. You will also see commands like C-x C-f and C-x C-w, listed beside the command names find-file and write-file.

Conclusion

Remember, to exit Emacs permanently use C-x C-c. To exit to a shell temporarily, so that you can come back in, use C-z.

This tutorial is meant to be understandable to all new users, so if you found something unclear, don't sit and blame yourself - complain!

Copying

This tutorial descends from a long line of Emacs tutorials starting with the one written by Stuart Cracraft for the original Emacs.

This version of the tutorial, like GNU Emacs, is copyrighted, and comes with permission to distribute copies on certain conditions:

The conditions for copying Emacs itself are slightly different but in the same spirit. Please read the file COPYING and then do give copies of GNU Emacs to your friends. Help stamp out software obstructionism ("ownership") by using, writing, and sharing free software!

Press here to return to the Editors Menu

Emacs Quick Reference

NOTES: This file is not inclusive.

C followed by a - and a letter means hold down the <CONTROL> key while pressing the appropriate letter.

M followed by a - and a letter means to depress and release the <ESC> key followed by the appropriate letter.

```
C-\ t                   Tutorial suggested for new emacs users.
C-x C-c                 exit emacs

--Files--
C-x C-f                 read a file into emacs
C-x C-s                 save a file back to disk
C-x i                   insert contents of another file into
                        this buffer
C-x C-v                 replace this file with the contents of
                        file you want
C-x C-w                 write buffer to specified file


--Error Recovery--
C-g                     abort partially typed or executing
                        command
M-x recover-file        recover a file lost by a system crash
C-x C-_                 undo an unwanted change
M-x revert-buffer       restore a buffer to its original
                        contents
C-l                     redraw garbled screen

--Getting Help--
C-\ t                   Tutorial recommended for new emacs users
C-x 1                   get rid of help window
M C-v                   scroll help window

--Incremental Search--
M-x goto-line           go to line number
C-s                     search forward (can repeat for next,
                        use M to exit)
C-r                     search backward (can repeat for next,
                        use M to exit)
C-r                     search backward
C-M-s                   regular expression search
M                       escape incremental search

--Cursor Movement--
C-f                     move forward one character
C-b                     move backward one character
C-n                     move to next line
C-p                     move to previous line
C-a                     move to beginning of line
C-e                     move to end of line
M-f                     move forward one word
M-b                     move backword one word
C-v                     move forward one screen
M-v                     move backward one screen
M-<                     go to beginning of file
M->                     go to end of file

--Killing and Deleting--
<DEL>                   delete backward one character
C-d                     delete forward one character
M-<DEL>                 delete backward one word
M-d                     delete forward one word
M-z {char}              kill to next occurrenc of character
C-k                     kill to end of line
```

```
C-y                     yank back last thing killed

--Marking--
C-@ or C-<SPC>          set mark here
C-x C-x                 exchange point and mark
M-h                     mark paragraph
C-x h                   mark entire buffer (alternately M-< M->)

--Replacing--
M-%                     interactively replace a test string
M-x replace-string      replace string with string
                        mark to end of file

--Multiple Windows--
C-x 1                   delete all other windows
C-x 0                   delete this window
C-x 2                   split this window in half vertically
C-x 5                   split this window in half horizontally
C-M-v                   scroll other window
C-x o                   switch cursor to another window

--Buffers--
C-x b                   select another buffer
C-x C-b                 list all buffers
C-x k                   kill a buffer

--Spelling Check--
M-$                     check spelling of current word
M-x spell-region        check spelling of marked region
M-x spell-buffer        check spelling of entire buffer

--Shells--
M-!                     execute a shell command
M-|                     run a shell command on region
C-u M-|                 filter a region through a shell command
```

Additional help is available in **man emacs**.

Press here to return to the Editors Menu

Running Macaulay 2 in Emacs

Because some answers can be very wide, it is a good idea to run Macaulay 2 in a window which does not wrap output lines and allows the user to scroll horizontally to see the rest of the output. We provide a package for "emacs" which implements this, in 'emacs/M2.el'. It also provides for dynamic completion of symbols in the language.

If you are a newcomer to emacs, start up emacs with the command 'emacs' and then start up the emacs tutorial with the keystrokes 'C-H t'. (The notation 'C-H' indicates that you should type 'Control-H', by holding down the control key, and pressing 'H'.) The emacs tutorial will introduce you to the basic keystrokes useful with emacs. After running through that you will want to examine the online emacs manual which can be read with 'info' mode; you may enter or re-enter that mode with the keystrokes 'C-H i'. You may also want to purchase (or print out) the emacs manual. It is cheap, comprehensive and informative. Once you have spent an hour with the emacs tutorial and manual, come back and continue from this point.

If you are reading this file with emacs, then use the keystrokes 'C-x 2' to divide the buffer containing this file into two windows. Then press the 'F12' function key to start up Macaulay 2 in a buffer named '*M2*'.

If this doesn't start up Macaulay 2, one reason may be that your function keys are not operable. In that case press 'C-C m' instead. (The notation 'C-C' is standard emacs notation for Control-C.) Another reason may be that you have not installed Macaulay 2 properly - the startup script ('M2' or 'M2.bat') should be on your path.
A third reason may be that you are in Windows-98 and are using anti-virus software such as 'Dr. Solomon's', which can interfere with emacs when it tries to run a subprocess.

You may use 'C-x o' freely to switch from one window to the other.
Verify that Macaulay 2 is running by entering a command such as '2+2'.
Now paste the following text into a buffer, unless you have the ASCII version of this documentation in an emacs buffer already, position the cursor on the first line of code, and press the 'F11' function key (or 'C-C s') repeatedly to present each line to Macaulay 2.

```
i1 : R = ZZ/101[x,y,z]
```

```
i2 : f = symmetricPower(2,vars R)
i3 : M = cokernel f
i4 : C = resolution M
i5 : betti C
```

Notice that the input prompts are not submitted to Macaulay 2.

There is a way to send a region of text to Macaulay 2: simply select a region of text, making sure the mark is active (as described above) and press 'F11'. Try that on the list below; put it into an emacs buffer, move your cursor to the start of the list, press 'M-C-@' or 'M-C-space' to mark the list, and then press 'F11' to send it to Macaulay 2. (The notation 'M-C-@' means: while holding down the Meta key and the Control key press the '@' key, for which you'll also need the shift key.)

```
{a,b,c,d,e,f,
g,h,i,j,k,l,
m,n}
```

Now let's see how we can handle wide and tall Macaulay 2 output. Execute the following line of code.

```
random(R^20,R^{6:-2})
```

Notice that the long lines in the Macaulay 2 window, instead of being wrapped around to the next line, simply disappear off the right side of the screen, as indicated by the dollar signs in the rightmost column. Switch to the other window and practice scrolling up and down with 'M-v' and 'C-v', and scrolling left and right with the function key 'F3' (or 'C-C <') and the function key 'F4' (or 'C-C >'). Notice how the use of 'C-E' to go to the end of the line sends the cursor to the dollar sign at the right hand side of the screen; that's where the cursor will appear whenever you go to a position off the screen to the right. Then use the 'F2' function key (or 'C-C .') to scroll the text so the cursor appears at the center of the screen. Use 'C-A' to move to the beginning of the line and then the 'F2' function key (or 'C-C .') to bring the left margin back into view.

You may use the 'F5' function key or (or 'C-C ?') to toggle whether long lines are truncated or wrapped; initially they are truncated.

Now go to the very end of the '*M2*' buffer with 'M->' and experiment with keyword completion. Type 'reso' and then press the 'TAB' key. Notice how the word is completed to 'resolution' for you. Delete the word with 'M-DEL', type 'res' and then press the 'TAB' key. The possible completions are displayed in a window. Switch to it with the 'F8' key, move to the desired completion, select it with the 'RETURN' key, and then return to the

'*M2*' buffer with 'C-X o'. Alternatively, if you have a
mouse, use the middle button to select the desired completion.

Experiment with command line history in the '*M2*' buffer. Position
your cursor at the end of the buffer, and then use 'M-p' and 'M-n'
to move to the previous and next line of input remembered in the history. When you
get to one you'd like to run again, simply press return to do so. Or edit it
slightly to change it before pressing return.

Assuming you have installed the "w3" emacs web browser,
you may explore the documentation by positioning the cursor near a documented
word such as 'List' and pressing 'C-C d'. Alternatively,
when the prompt appears, you can type the key whose documentation should be
found.

```
##########################################################################
##########################################################################
#
#           IVA GROBNER PACKAGE FOR MAPLE V, RELEASE 5
#
#             by Albert Lin and Philippe Loustaunau,
#               George Mason University
#
#             Minor modifications and corrections by David Cox,
#               Amherst College (04/21/99)
#
#             Modifications to work under Maple V.4 and V.5
#               by C. D. Wensley, University of Wales, Bangor
#               (02/27/98 and 03/03/99)
#
#             Further modifications by William Gryc,
#               Amherst College (08/13/99)
#
#   All comments, suggestions, and bug reports should be sent to:
#
#               David A. Cox, Amherst College
#
#               email:  dac@cs.amherst.edu
#
##########################################################################
##########################################################################
#
#                    HOW TO START
#
#     Suppose that the file containing this Maple package is called
#     gbr5.mpl.  Then, if you start Maple from the directory
#     containing this file, you can load the file into Maple using
#     the command
#
#         > read(`gbr5.mpl`):
#
#     If the file resides in another directory, then the appropriate
#     path needs to be included in the read statement.
#
##########################################################################
##########################################################################
#
#                  THE MAJOR COMMANDS
#
#     This package has several major commands:
#
#     1) ring, which sets the term order and variables used by the
```

```
#         package.
#
#      2) div_alg (for "division algorithm"), which computes the
#         remainders AND quotients for the division algorithm.
#
#      3) slowbasis_gb, altbasis_gb and quickbasis_gb, which are
#         three versions of the basic Grobner basis algorithm.
#
#      4) min_gb and red_gb, which take a Grobner basis produced by
#         the three *_gb commands from 3) and make them minimal and
#         reduced respectively.
#
#      5) quot_mx (for "matrix of quotients"), which computes a matrix
#         telling you how to transform the Grobner basis into the
#         original basis.
#
#      6) mxgb (for "matrix grobner"), which computes a reduced
#         basis together with a matrix telling you how to transform
#         the original basis into the grobner basis.
#
#         mxgb is the union of three separate commands:
#           (i) quickbasis_mxgb  which computes a grobner basis which in
#                            general is neither minimal nor reduced;
#          (ii)       min_mxgb  which converts a basis to a minimal basis;
#         (iii)       red_mxgb  which reduces a minimal basis.
#
#      Commands 2) - 6) use a monomial order which is set by the
#      user through the command ring().  The names of the predefined Maple
#      term orders (plex and tdeg) should not be used.  The orders lex,
#      grevlex, grlex, [k,n] (elimination order), and [v1,...,vn] (matrix
#      order) are valid monomial orders (see ring() described below).
#
#      These commands are described in more detail below.
#
######################################################################
######################################################################
#                 THE ring COMMAND
#
#      The INPUT is:
#
#            > ring(torder, varlist);
#
#      where
#            torder is the order on the polynomials.  The valid values
#                  for torder are lex, grlex, grevlex, [k,n], and
#                  [v1,...,vn].  [k,n] defines the kth elimination order
#                  on n variables.  Note that this means the number of
#                  variables must equal n.  [v1,...,vn] is a list of
```

```
#                   lists, where vi is a n-length list used as a row
#                   vector, and [v1,...,vn] defines a matrix order on n
#                   variables.  Note that the vi's must be linearly
#                   independant and there must be n variables in varlist.
#
#             varlist is the list of variables in the ring.
#
#       The OUTPUT is:
#
#             The term-order defined by the Grobner package.  The output
#             is not terribly important in this command.  The command's
#             main purpose is to set the monomial order for all other
#             commands which need a termorder.
#
######################################################################
######################################################################
#
#               THE grlex AND elimination COMMANDS
#
#       Since they can be represented by matrix orders and are not
#       built into Maple, ring() sets grlex and elimination orders
#       using matrices.  ring() uses the commands grlex() and
#       elimination() to get these matrices for grlex and elimination
#       orders.
#
#       These commands can be used as part of the input of the
#       ring() command directly, although this is hardly necessary.
#
#       The INPUT is:
#
#             > grlex(n)
#
#       where
#             n is the number of variables.
#
#       The OUTPUT is:
#
#             the vector list that specifies graded lexicographic
#             order for n variables.
#
#       The INPUT is:
#
#             > elimination(k,n)
#
#       where
#             n is the number of variables,
#
#             k is the number of variables to be eliminated.
```

```
#
#     The OUTPUT is:
#
#           the vector list that specifies the kth elimination order on
#           n variables.
#
#####################################################################
#
#           EXAMPLES OF THE grlex AND elimination COMMANDS
#
#     If the input is:
#
#           > grlex(4);
#
#     then the output is the vector list that specifies lex order
#     for 4 variables:
#
#           [[1,1,1,1],[1,0,0,0],[0,1,0,0],[0,0,1,0]]
#
#     If the input is:
#
#           > elimination(2,4);
#
#     then the output is the vector list that specifies elimination
#     order that eliminates the first 2 of 4 variables:
#
#           [[1,1,0,0],[0,0,1,1],[0,0,0,-1],[0,-1,0,0]]
#
#####################################################################
#####################################################################
#
#                THE div_alg COMMAND
#
#     The INPUT is:
#
#           > div_alg(f,[f1,...,fs]);
#
#     where
#           f is the polynomial to be divided
#
#           [f1,...,fs] is the list of input polynomials.
#
#     The OUTPUT is:
#
#           the remainder r and the quotients a1,...,as of the
#           division of f by f1,...,fs, with respect to the term
#           order set by ring().  In other words,
#
```

```
#                 f = a1 f1 + ... + as fs + r.
#
######################################################################
#
#               EXAMPLE OF THE div_alg COMMAND
#
#       If the input is:
#
#             > ring(lex, [x,y,z]);
#             > Polys := [x^2+y^2+z^2-4,x^2+2*y^2-5,x*z-1];
#             > div_alg(x^3+2*y*z,Polys);
#
#       then the output is the remainder r and quotients [a1,a2,a3]
#       of the division of x^3 + 2*y*z on division by Polys:
#
#             [-x*y^2 + 4*x + 2*y*z - z,[x,0,-z]]
#
#       So the remainder is -x*y^2 + 4*x + 2*y*z - z
#       and the quotients are  x, 0 and -z.
#
######################################################################
######################################################################
#
#       THE slowbasis_gb, altbasis_gb, AND quickbasis_gb COMMANDS
#
#       All these commands have the same format and do the same thing:
#       They all calculate a Grobner basis that is generally not
#       minimal nor reduced.  They also print out the steps of the
#       Grobner basis calculation.  If this is not desired, there is
#       an optional second argument, nosteps, that will supress this
#       printing.  The difference between the three is how they use
#       Buchberger's algorithm.  slowbasis_gb uses a naive version of
#       the algorithm, altbasis_gb uses a slightly more insightful version,
#       and quickbasis_gb uses a more advanced version.  Since the input
#       and output are the same for all three commands, we will just use
#       quickbasis_gb as an example.
#
#       The INPUT is:
#
#             > quickbasis_gb([f1,...,fs]);
#       OR
#             > quickbasis_gb([f1,...,fs], nosteps);
#
#       where
#             [f1,...,fs] is the list of input polynomials.
#
#             nosteps is just the string "nosteps" (minus the quotes).
#
```

```
#       The OUTPUT is:
#
#               [g1,...,gt], a list of polynomial that form a Grobner
#               basis under the termorder set by ring() for <f1,...,fs>.
#               If the second argument "nosteps" is not entered, the
#               steps of the calculation with be printed.
#
#############################################################################
#############################################################################
#
#                       THE min_gb AND red_gb COMMANDS
#
#       Three commands slowbasis_gb, altbasis_gb and quickbasis_gb
#       compute Grobner bases which in general are neither minimal nor
#       reduced.  Hence there are companion commands min_gb and red_gb
#       to minimize and reduce them.  These commands are described
#       below.
#
#############################################################################
#############################################################################
#
#                       THE min_gb COMMAND
#
#       The INPUT is:
#
#               > min_mxgb([f1,...,fs]);
#
#       where
#               [f1,...,fs] is the list of input polynomials which are
#               a Grobner basis in the termorder set by ring.
#
#       The OUTPUT is:
#
#               [[g1,...,gt]]
#
#       where
#               [g1,...,gt] is a minimal Grobner basis of <f1,...,fs>,
#               which in general is not reduced.
#
#############################################################################
#############################################################################
#
#                       THE red_gb COMMAND
#
#       The INPUT is:
#
#               > red_gb([f1,...,fs]);
#
```

```
#       where
#               [f1,...,fs] is the list of input polynomials which are
#               a minimal Grobner basis with respect to the termorder
#               set by ring().
#
#       The OUTPUT is:
#
#               [g1,...,gt]
#
#       where
#               [g1,...,gt] is a reduced Grobner basis of <f1,...,fs>.
#
#######################################################################
#######################################################################
#
#                       THE quot_mx COMMAND
#
#       The INPUT is:
#
#               > quot_mx([f1,...,fs],[g1,...,gt]);
#
#       where
#               [f1,...,fs] is a list of polynomials
#
#               [g1,...,gt] is a Grobner basis for the ideal <f1,...,fs>
#               with respect to the term order set by ring().
#
#       The OUTPUT is:
#
#               A matrix Q such that Q * [g1,...,gt]^T =[f1,...,fs]^T.
#               In other Q, the ith row of Q gives how fi is expressed
#               in terms of [g1,...,gt].
#
#######################################################################
#
#                       EXAMPLE OF THE quot_mx COMMAND
#
#       If the input is:
#
#               > ring(grevlex, [x,y]);
#               > quot_mx([x^2*y - 1,x*y^2 - x],[-y + x^2, y^2-1]);
#
#       then the output is a 2 x 2 matrix which tells how to express
#       the original basis in terms of the Grobner basis:
#
#                       [ y  1 ]
#                       [      ]
#                       [ 0  x ]
```

```
#
#	Thus
#		x^2*y - 1 = y*(-y+x^2) + 1*(y^2-1)
#	and
#		x*y^2 - x = 0*(-y+x^2) + x*(y^2-1)
#
######################################################################
######################################################################
#
#			THE mxgb COMMAND
#
#	The INPUT is:
#
#		> mxgb([f1,...,fs]);
#
#	where
#		[f1,...,fs] is the list of input polynomials, and steps
#		of the Grobner basis calculation will be printed.
#
#	OR
#
#		> mxgb([f1,...,fs], nosteps);
#
#	where
#		[f1,...,fs] is the list of input polynomials, and steps of
#		the Grobner basis calculation will no be printed.
#
#
#	The OUTPUT is:
#
#		a list
#
#			[[g1,...,gt], M]
#
#		where [g1,...,gt] is the reduced Grobner basis of
#		<f1,...,fs> with respect to the termorder set by
#		ring() and M is a t x s matrix giving the Grobner
#		basis as a combination of the fi's; i.e.,
#
#				[g1 ]     [f1 ]
#				[g2 ]     [f2 ]
#				[...] = M [...]
#				[...]     [...]
#				[gt ]     [fs ]
#
######################################################################
#
#			EXAMPLE OF THE mxgb COMMAND
```

```
#
#      If the input is:
#
#            > ring(lex, [x,y,z]);
#            > Polys := [x^2+y^2+z^2-4, x^2+2*y^2-5, x*z-1];
#            > gb := mxgb(Polys);
#
#      then the output is the reduced Grobner basis G of Polys
#      with respect to the lexicographic ordering with x > y > z,
#      and the matrix M of transformation such that G=M*Polys:
#
#      [
#      [
#      [[y^2 - z^2 - 1, x + 2z^3 - 3z, 1/2 + z^4 - 3/2 z^2],
#      [
#      [
#
#                [   -1        1                0        ]]
#                [                                      ]]
#                [   2z       - z             - x       ]]
#                [                                      ]]
#                [  z^2   - 1/2 z^2    - 1/2 - 1/2 x *z  ]]
#
#      Also, the intermediate steps of the computation (the "unminimized"
#      basis and matrix step, and the "unreduced" basis and matrix step)
#      are also printed during computation, unless the optional second
#      argument "nosteps" is included.  The command line for this option
#      would look like:
#
#            > mxgb(Polys, nosteps);
#
#########################################################################
#########################################################################
#
#          THE quickbasis_mxgb, min_mxgb AND red_mxgb COMMANDS
#
#      As explained above, the command mxgb operates by calling the
#      three commands:
#      1) quickbasis_mxgb, which computes a Grobner basis (and associated
#          matrix) which is usually neither minimal nor reduced.
#      2) min_mxgb, which takes a Grobner basis (and matrix) and
#         produces a minimal Grobner basis (and matrix).
#      3) red_mxgb, which takes a minimal Grobner basis (and matrix)
#         and produces a reduced Grobner basis (and matrix).
#      Below, we give a brief description of these commands (though
#      here we don't include examples).
#
#########################################################################
```

```
######################################################################
#
#                  THE quickbasis_mxgb COMMAND
#
#       The INPUT is:
#
#             > quickbasis_mxgb([f1,...,fs]);
#
#       where
#             [f1,...,fs] is the list of input polynomials.
#
#       The OUTPUT is:
#
#             a 3-element list:
#
#                   ["isgbasis",[g1,...,gt],coeff_mx]
#
#       where
#             [g1,...,gt] is a Grobner basis of <f1,...,fs> with
#             respect to the termorder set by ring(); the Grobner
#             basis is in general is neither minimal nor reduced,
#
#             coeff_mx is the corresponding transformation matrix.
#
######################################################################
######################################################################
#
#                  THE min_mxgb COMMAND
#
#       The INPUT is:
#
#             > min_mxgb("isgbasis", [f1,...,fs],coeff_mx);
#
#       where
#            "isgbasis" is just a string
#
#            [f1,...,fs] is the list of input polynomials which are
#             a grobner basis.
#
#             coeff_mx is the corresponding transformation matrix for the
#            Grobner basis
#
#       The OUTPUT is:
#
#            a 3-element list:
#
#                  ["isminimal",[g1,...,gt],coeff_mx]
#
```

```
#       where
#             [g1,...,gt] is a minimal Grobner basis of <f1,...,fs>,
#              which in general is not reduced,
#
#              coeff_mx is the corresponding transformation matrix.
#
########################################################################
########################################################################
#
#                    THE red_mxgb COMMAND
#
#       The INPUT is:
#
#              > red_mxgb("isminimal",[f1,...,fs],coeff_mx);
#
#       where
#             [f1,...,fs] is the list of input polynomials which are
#              a minimal Grobner basis with respect to the termorder
#             set by ring().
#
#       The OUTPUT is:
#
#             a 2-element list
#
#                    [[g1,...,gt],coeff_mx]
#
#       where
#             [g1,...,gt] is a reduced Grobner basis of <f1,...,fs>,
#
#              coeff_mx is the corresponding transformation matrix.
#
########################################################################
########################################################################
#
#                    END OF DOCUMENTATION
#
########################################################################


########################################################################
#
#      HERE ARE STANDARD MAPLE PACKAGES USED BY gbr5.
#
########################################################################

with(linalg);
with(Groebner);
with(Ore_algebra);
```

```
#######################################################################
#
#             HERE IS THE CODE FOR mxgb.
#
#######################################################################

mxgb := proc(F::list)
   local basislist, minlist, redlist, temp1, temp2;
   basislist := quickbasis_mxgb(F, nosteps);
   if nargs = 1 then
     temp1 := basislist[2];
     print(`Unminimized basis: `, temp1);
     temp2 := op(basislist[3]);
     print(`Unminimized matrix: `, temp2);
   fi;
   minlist := min_mxgb( basislist );
   if nargs = 1 then
     temp1 := minlist[2];
     print(`Minimized basis: `, minlist[2]);
     temp2 := op(minlist[3]);
     print(`Minimized matrix: `, temp2);
   fi;
   redlist := red_mxgb( minlist );
   redlist[3] := op(redlist[3]);
   redlist[2..3];
end;

#######################################################################
#
#    HERE IS THE CODE FOR _leadingterm.  THIS FINDS LEADING TERMS
#    AND IS USED BY ALL OF THE GROBNER BASIS ROUTINES.
#
#######################################################################

_leadingterm := proc(f)
   local f1, mono;
   global morder;
   if type(op(morder), name) then
      ERROR(`Monomial Order not set.  Use ring() to set it.`) fi;
   f1 := expand(f);
   mono := leadmon(f1, morder);
   mono;
end;

#######################################################################
#
#      HERE IS THE CODE FOR div_alg.
```

```
#
########################################################################

div_alg := proc( g, Set )
    local  h, i, a, v, f, lmf, lmg, b, c, d;
    a := array(1..nops(Set));
    for b to nops(Set) do
        f := Set[b];
        a[b] := 0;
        lmf[b] := _leadingterm( f );
    od;
    v := 0;
    h := g;
    while (h<>0) do
        lmg := _leadingterm( h );
        for i to nops(Set) do
            f := Set[i];
            if (f <> 0) then
                d := degree( denom( simplify( lmg[2]/lmf[i][2] ) ) );
                if d=0 then
                    a[i] := simplify( a[i]+lmg[1]*lmg[2]/(lmf[i][1]*lmf[i][2]) );
                    h := simplify( h-f*lmg[1]*lmg[2]/(lmf[i][1]*lmf[i][2]) );
                    if h<>0 then
                        i:=0
                    fi;
                    lmg := _leadingterm( h );
                fi;
            else
                a[i]:=0;
            fi;
        od;
        v := simplify( v+lmg[1]*lmg[2] );
        h := simplify( h-lmg[1]*lmg[2] );
    od;
    a := convert( a, list );
    c := [v,a];
end;

########################################################################
#
#      HERE IS THE CODE FOR quot_mx.
#
########################################################################

quot_mx := proc( pols, bas )
    local  np, nb, index1, index2, d, Q;
    np := nops( pols );
    nb := nops( bas );
```

```
   Q := matrix( np, nb, 0 );
   for index1  to np do
      d := div_alg( pols[index1], bas );
      if ( d[1]<>0 ) then
         ERROR( `non-zero remainder` );
      fi;
      for index2 to nb do
         Q[index1,index2] := d[2][index2];
      od;
   od;
   op(Q);
end;

######################################################################
#
#     HERE IS THE CODE FOR min_mxgb.  THIS IS USED BY mxgb.
#
######################################################################

min_mxgb := proc( gbrec )
   local y, i, j, n, H, G, coeff_mx, numcol, minrec,lt;
   if not ( ( nargs = 1 ) and type( gbrec, list )
          and ( nops( gbrec ) = 3 ) and type( gbrec[1], string )
          and ( gbrec[1] = "isgbasis" ) ) then
      ERROR(`input not a matrixgrobner record` );
   fi;
   G := gbrec[2];
   coeff_mx := gbrec[3];
   numcol := coldim( coeff_mx );
   n:=nops(G);
   H:=G;
   i:=1;
   while i<=n do
      lt := _leadingterm( H[i] )[2];
      j:=1;
      while j<=n do
         if i<>j then
            if divide(lt,_leadingterm( H[j] )[2]) then
               if (i=1) then
                  H := H[2..n];
               elif (i=n) then
                  H := H[1..(n-1)];
               else
                  H := [ op( H[1..(i-1)] ), op( H[(i+1)..n] ) ];
               fi;
               for y from i to n-1 do
                  coeff_mx:=swaprow( coeff_mx, y, y+1 );
               od;
```

```
          coeff_mx:=submatrix( coeff_mx, 1..n-1, 1..numcol );
          n:=n-1;
          j:=n;
          i := i-1;
        fi;
      fi;
      j:=j+1;
    od;
    i := i+1;
  od;
  minrec := [ "isminimal", H, coeff_mx ];
end;

########################################################################
#
#    HERE IS THE CODE FOR min_gb.
#
########################################################################

min_gb := proc( gb )
  local i, j, n, H, G, minrec,lt;
  if not isgbasis(gb) then
   ERROR(`input not a Grobner basis`);
  fi;
  G := gb;
  n:=nops(G);
  H:=G;
  i:=1;
  while i<=n do
    lt := _leadingterm( H[i] )[2];
    j:=1;
    while j<=n do
      if i<>j then
        if divide(lt,_leadingterm( H[j] )[2]) then
          if (i=1) then
            H := H[2..n];
          elif (i=n) then
            H := H[1..(n-1)];
          else
            H := [ op( H[1..(i-1)] ), op( H[(i+1)..n] ) ];
          fi;
          n:=n-1;
          j:=n;
          i := i-1;
        fi;
      fi;
      j:=j+1;
    od;
```

```
      i := i+1;
   od;
   minrec := H ;
end;

######################################################################
#
#  HERE IS THE CODE FOR is_min_gbasis.  THIS TESTS FOR MINIMALITY
#  AND IS USED BY red_gb.
#
######################################################################

is_min_gbasis := proc(G::list)
   local i, j, n, lt;
   if not isgbasis(G) then
     RETURN(false);
   fi;
   n := nops(G);
   for i to n do
     for j to n do
       lt := _leadingterm(G[i])[2];
       if i <> j then
          if divide(lt, _leadingterm(G[j])[2]) then
            RETURN(false);
          fi;
       fi;
     od;
   od;
   true;
end;


######################################################################
#
#      HERE IS THE CODE FOR isgbasis.  THIS TESTS FOR BEING A GROBNER
#      BASIS AND IS USED BY min_gb AND is_min_gbasis.
#
######################################################################

isgbasis := proc(G::list)
   local i,j,n,r;
   global morder;
   n := nops(G);
   for i to n do
     for j to n do
       if i <> j then
         r := normalf(spoly(G[i], G[j], morder), G, morder);
         if r <> 0 then
```

```
          RETURN(false);
        fi;
      fi;
    od;
  od;
  true;
end;

######################################################################
#
#     HERE IS THE CODE FOR quickbasis_mxgb.  THIS IS USED BY mxgb.
#
######################################################################

quickbasis_mxgb := proc(H::list)
  local x, y, setofpairs, z, p, l, F, r, i, n, lmf, lmg, ernie,
     u, T, Q, index1, index2, index3, index4, index5, subscript,
     coeff_tab, coeff_mx, numcol, gbrec, divs;
  divs := 0;
  F:=[];
  coeff_tab := table(sparse);
  for i to nops(H) do
    coeff_tab[i,i]:=1;
  od;
  F:=H;
  if nargs = 1 then
    print(`Current Basis: `.F)
  fi;
  setofpairs:=[];
  for index1 to nops(H)-1 do
    for index2 from index1+1 to nops(H) do
      setofpairs:=[ op(setofpairs), [index1, index2] ];
    od;
  od;
  while nops(setofpairs) <>0 do
    numcol := nops(H);
    subscript := setofpairs[1];
    setofpairs := [ op(2..nops(setofpairs), setofpairs) ];
    lmf := _leadingterm( F[subscript[1]] );
    lmg := _leadingterm( F[subscript[2]] );
    if ( gcd( lmf[2], lmg[2] ) <> 1 ) then
      ernie := 0;
      for u to subscript[1]-1 do
        if divide(lcm(lmf[2],lmg[2]),_leadingterm(F[u])[2])
          then ernie:=1;
              break;
        fi;
      od;
```

```
            if ernie = 0 then
                p := lcm(lmf,lmg);
                T := p*F[subscript[1]]/(lmf[1]*lmf[2])
                     - p*F[subscript[2]]/(lmg[1]*lmg[2]);
                Q := div_alg(simplify(T),F);
                r := Q[1];
                if ( r <> 0 ) then
                    for index3 to nops(F) do
                        setofpairs:=[op(setofpairs),[index3,nops(F)+1]];
                    od;
                    F := [op(F),r];
                    if ( nargs = 1 ) then
                        print(`Added: `.r)
                    fi;
                    divs := divs + 1;
                    n := nops(F);
                    for l to n-1 do
                        coeff_tab[n,l] := - op(l,Q[2]);
                    od;
                    coeff_tab[n,subscript[1]] :=
                     simplify( coeff_tab[n,subscript[1]] + p/(lmf[1]*lmf[2]) );
                    coeff_tab[n,subscript[2]] :=
                     simplify( coeff_tab[n,subscript[2]] - p/(lmg[1]*lmg[2]) );
                fi;
            fi;
        fi;
    od;
    if ( nops(F) > nops(H) ) then
        for x from numcol+2 to n do
            for y to numcol do
                for z from numcol+1 to x-1 do
                    coeff_tab[x,y] := simplify( coeff_tab[x,y]
                                  + coeff_tab[x,z]*coeff_tab[z,y] );
                od;
            od;
        od;
    fi;
    coeff_mx := array( 1..nops(F), 1..numcol );
    for index4 to nops(F) do
    for index5 to numcol do
        coeff_mx[ index4, index5 ] := coeff_tab[ index4, index5 ];
        od;
    od;
    gbrec := [ "isgbasis", F, coeff_mx ];
end;

#######################################################################
#
```

```
#      HERE IS THE CODE FOR quickbasis_gb.
#
###########################################################################

quickbasis_gb := proc(H::list)
    local x, y, setofpairs, z, p, l, F, r, i, n, lmf, lmg, ernie, numcol,
          u, T, Q, index1, index2, index3, subscript, gbrec, divs,localdivs;
    global morder;
    F:=[];
    F:=H;
    divs := 0;
    localdivs := 0;
    if nargs = 1 then
        print(`Current Basis: `.F)
    fi;
    setofpairs:=[];
    for index1 to nops(H)-1 do
        for index2 from index1+1 to nops(H) do
            setofpairs:=[ op(setofpairs), [index1, index2] ];
        od;
    od;
    while nops(setofpairs) <>0 do
        numcol := nops(H);
        subscript := setofpairs[1];
        setofpairs := [ op(2..nops(setofpairs), setofpairs) ];
        lmf := _leadingterm( F[subscript[1]] );
        lmg := _leadingterm( F[subscript[2]] );
        if ( gcd( lmf[2], lmg[2] ) <> 1 ) then
            ernie := 0;
            for u to subscript[1]-1 do
               if divide(lcm(lmf[2],lmg[2]),_leadingterm(F[u])[2])
                  then ernie:=1;
                       break;
               fi;
            od;
            if ernie = 0 then
                p := lcm(lmf,lmg);
                T := p*F[subscript[1]]/(lmf[1]*lmf[2])
                     - p*F[subscript[2]]/(lmg[1]*lmg[2]);
                Q := normalf(simplify(T),F,morder);
                divs := divs + 1;
                localdivs := localdivs + 1;
                r := Q;
                if ( r <> 0 ) then
                    for index3 to nops(F) do
                        setofpairs:=[op(setofpairs),[index3,nops(F)+1]];
                    od;
                    F := [op(F),r];
```

```
          if ( nargs = 1 ) then
            print(`Added: `.r);
            print(`Local divisions: `.localdivs);
          fi;
          localdivs := 0;
        fi;
      fi;
    fi;
  od;
  if (nargs = 1) then
    print(`Total divisions performed: `. divs);
  fi;
  gbrec := F;
end;

######################################################################
#
#     HERE IS THE CODE FOR slowbasis_gb.
#
######################################################################

slowbasis_gb := proc(F::list)
  local G, Gprime, S, noneadded, p, q, printable, steps, divs,i,j,length,
      oldlength,localdivs;
  global morder;
  divs := 0;
  if nargs = 1 then
   print(`Current basis: `. F)
  fi;
  G := F;
  steps = 0;
  noneadded := false;
  oldlength := 1;
  while not noneadded do
   localdivs := 0;
   steps := steps + 1;
   printable := [];
   noneadded := true;
   Gprime := G;
   length := nops(Gprime);
   for i from 1 to length do
    p := Gprime[i];
    for j from max(i+1, oldlength + 1) to length do
     q := Gprime[j];
     S := normalf(spoly(p,q,morder), Gprime, morder);
     divs := divs + 1;
     localdivs := localdivs + 1;
     if S <> 0 then
```

```
            G := [op(G), S];
            printable := [op(printable), S];
            noneadded := false
          fi;
        od;
      od;
      oldlength := length;
      if nargs = 1 then
        print(`Added: `. printable);
        print(`Local divisions: `. localdivs);
      fi;
    od;
    if (nargs = 1) then
        print(`Total divisions performed: `. divs);
    fi;
    G;
end;

######################################################################
#
#      HERE IS THE CODE FOR altbasis_gb.
#
######################################################################

altbasis_gb := proc(F::list)
    local G, Gprime, S, noneadded, p, q, printable, steps, divs,i,j,length,
          oldlength,localdivs;
    global morder;
    divs := 0;
    if nargs = 1 then
      print(`Current basis: `.F)
    fi;
    G := F;
    steps = 0;
    noneadded := false;
    oldlength := 1;
    while not noneadded do
      localdivs := 0;
      steps := steps + 1;
      printable := [];
      noneadded := true;
      Gprime := G;
      length := nops(Gprime);
      for i from 1 to length do
        p := Gprime[i];
        for j from max(i+1, oldlength + 1) to length do
          q := Gprime[j];
          if is(p = q) then next fi;
```

```
          S := normalf(spoly(p,q,morder), G, morder);
          divs := divs + 1;
          localdivs := localdivs + 1;
          if S <> 0 then
            G := [op(G), S];
            printable := [op(printable), S];
            noneadded := false
          fi;
        od;
      od;
      oldlength := length;
      if nargs = 1 then
        print(`Added: `. printable);
        print(`Local divisions: `. localdivs)
      fi;
    od;
    if (nargs = 1) then
        print(`Total divisions performed: `. divs);
    fi;
    G;
end;

######################################################################
#
#      HERE IS THE CODE FOR red_mxgb.  THIS IS USED BY mxgb.
#
######################################################################

red_mxgb := proc( minrec )
    local  x, Ca, t, i, j, k, l, m, n, a, o, Da, Db, r, J, J0, J1,
           coeff_mx, index1, index2, redmx, grobmx, grob1;
    if not ( ( nargs = 1 ) and type( minrec[1], string )
                        and ( minrec[1] = "isminimal" ) ) then
        ERROR(`input not a minimal matrixgrobner record` );
    fi;
    J := minrec[2];
    grobmx := minrec[3];
    n := nops(J);
    m := coldim( grobmx );
    grob1 := submatrix( grobmx, 1..n, 1..m );
    i:=1;
    Ca:=table(sparse);
    if n<>1 then
        while (i <= n) do
          # Da:=[op(1..i-1,J)];
          # Db:=[op(i+1..n,J)];
                if (i=1) then
                     Da := [ ];
```

```
                Db := J[2..n];
            elif (i=n) then
                Da := J[1..(n-1)];
                Db := [ ];
            else
                Da := J[1..(i-1)];
                Db := J[(i+1)..n];
            fi;
        r := div_alg( J[i], [ op(Da), op(Db) ] );
        J := [ op(Da), r[1], op(Db) ];
        for j to n do
            for k to i-1 do
                Ca[i,j]:=simplify(Ca[i,j]-r[2][k]*Ca[k,j]);
            od;
        od;
        for l from i+1 to n do
            Ca[i,l]:=simplify(Ca[i,l]-r[2][l-1]);
        od;
        Ca[i,i]:=simplify(Ca[i,i]+1);
        i:=i+1;
    od;
else
    Ca[1,1]:=1;
fi;
redmx := matrix( n, n, 0 );
for index1 to n do
    for index2 to n do
        redmx[ index1, index2 ] := Ca[ index1, index2 ];
    od;
od;
J0 := [ ];
J1 := [ ];
for a to n do
    if (J[a]<>0) then
        J0 := [ op(J0), a ];
        J1 := [ op(J1), J[a] ];
    fi;
od;
redmx := submatrix( redmx, J0, J0 );
grob1 := submatrix( grob1, J0, 1..m );
n := nops( J0 );
J := J1;
for a to n do
    x := _leadingterm( J[a] )[1];
    o := J[a]/x;
    for t to n do
        redmx[a,t] := redmx[a,t]/x;
    od;
```

```
        if (a=1) then
            J := [ o, op(2..n,J) ];
        elif (a=n) then
            J := [ op(1..n-1,J), o ];
        else
            J := [ op(1..a-1,J), o, op(a+1..n,J) ];
        fi;
    od;
    coeff_mx := multiply( redmx, grob1 );
    [ "isreduced", J, coeff_mx ];
end;

#######################################################################
#
#     HERE IS THE CODE FOR red_gb.
#
#######################################################################

red_gb := proc(minrec)
    local  x, t, i, j, k, l, m, n, a, o, Da, Db, r, J, J0, J1,
           index1, index;
    global morder;
    if not is_min_gbasis(minrec) then
      ERROR(`input not a minimal Grobner basis`);
    fi;
    J := minrec;
    n := nops(J);
    i:=1;
    if n<>1 then
        while (i <= n) do
                if (i=1) then
                    Da := [ ];
                    Db := J[2..n];
                 elif (i=n) then
                    Da := J[1..(n-1)];
                    Db := [ ];
                 else
                    Da := J[1..(i-1)];
                    Db := J[(i+1)..n];
                 fi;
            r := normalf( J[i], [ op(Da), op(Db) ], morder );
            J := [ op(Da), r, op(Db) ];
            i := i + 1;
        od;
    fi;
    J0 := [ ];
    J1 := [ ];
    for a to n do
```

```
        if (J[a]<>0) then
            J0 := [ op(J0), a ];
            J1 := [ op(J1), J[a] ];
        fi;
    od;
    n := nops( J0 );
    J := J1;
    for a to n do
        x := _leadingterm( J[a] )[1];
        o := J[a]/x;
        if (a=1) then
            J := [ o, op(2..n,J) ];
        elif (a=n) then
            J := [ op(1..n-1,J), o ];
        else
            J := [ op(1..a-1,J), o, op(a+1..n,J) ];
        fi;
    od;
    J;
end;

######################################################################
#
#     HERE IS THE CODE FOR ring.
#
######################################################################

ring := proc(torder,varlist)
     local badform,i;
     global morder;
     badform := false;
     if is(torder, list) then
        if is(torder[1], list) then
           for i from 1 to (nops(torder)-1) do
              badform := is(nops(torder[i]) <> nops(torder[i+1]));
              if badform then break fi;
              od;
           badform := badform or is(nops(torder) <> nops(varlist));
           if badform or is(nops(torder) <> nops(torder[1])) then
              ERROR(`Not a valid matrix`);
           else
              morder := termorder(poly_algebra(op(varlist)),'matrix'
                        (torder, varlist));
           fi;
        elif is(nops(torder) <> 2) or is(torder[1] >= torder[2]) or
             is(torder [1] <= 0) or is(torder[2]<>nops(varlist)) then
           ERROR(`Not a valid elimination form`);
        else
```

```
         morder := termorder(poly_algebra(op(varlist)),'matrix'
                (elimination(torder[1], torder[2]), varlist));
       fi;
    elif is(torder = grlex) then
       morder := termorder(poly_algebra(op(varlist)),'matrix'
                (grlex(nops(varlist)), varlist));
    elif is(torder = grevlex) then
       morder := termorder(poly_algebra(op(varlist)), tdeg(op(varlist)));
    elif is(torder = lex) then
       morder := termorder(poly_algebra(op(varlist)), plex(op(varlist)));
    elif (torder = current) then
    else
       ERROR(`First argument is not a valid monomial order`);
    fi;
    morder;
end;

######################################################################
#
#     HERE IS THE CODE FOR grlex.  THIS IS USED BY ring TO GENERATE
#     A MATRIX WHICH GIVES THE GRLEX ORDER.
#
######################################################################

grlex := proc(n)
   local u,i,j;
   if n=1 then
      [[1]];
   else
      u := [0 $ i=1..j-1,1,0 $ i=j+1..n];
      [[1 $ i=1..n],u $ j=1..n-1]
   fi;
end;

######################################################################
#
#     HERE IS THE CODE FOR elimination.  THIS IS USED BY ring TO
#     GENERATE A MATRIX WHICH GIVES THE ELIMINATION ORDER.
#
######################################################################

elimination := proc(k,n)
   local u,i,j;
   if k >= n or k = 0 then
     ERROR(`Not a valid elimination form.`);
   elif n=1 then
     [[1]];
   else
```

```
      u := [0 $ i=1..n-j-1, -1, 0 $i=n-j+1..n];
      [[1 $ i=1..k, 0 $i=k+1..n],[0 $ i=1..k, 1 $i=k+1..n],u $ j=0..n-k-2,
       u $ j =n-k..n-2]
    fi;
end;
```

{VERSION 3 0 "SUN SPARC SOLARIS" "3.0" }
{USTYLETAB {CSTYLE "Maple Input" -1 0 "Courier" 0 1 255 0 0 1 0 1 0 0
1 0 0 0 0 }{CSTYLE "2D Math" -1 2 "Times" 0 1 0 0 0 0 0 0 2 0 0 0 0 0
0 }{CSTYLE "2D Comment" 2 18 "" 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 }
{CSTYLE "2D Output" 2 20 "" 0 1 0 0 255 1 0 0 0 0 0 0 0 0 0 }{CSTYLE "
" -1 256 "" 1 18 0 0 0 0 0 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 257 "" 1 24
0 0 0 0 0 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 258 "" 0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 }{CSTYLE "" -1 259 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE "
" -1 260 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 261 "" 0 1 0
0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 262 "" 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 }{CSTYLE "" -1 263 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE ""
-1 264 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE "" -1 265 "" 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE "" -1 266 "" 0 1 0 0 0 0 0 1 0 0 0 0 0
0 0 }{CSTYLE "" -1 267 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE ""
-1 268 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE "" -1 269 "" 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE "" -1 270 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 271 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 272 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 273 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 274 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 275 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 276 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 277 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 278 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 279 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 280 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 281 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 282 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 283 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 284 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 285 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 286 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 287 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 288 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 289 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 290 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 291 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 292 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 293 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 294 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 295 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 296 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 297 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 298 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 299 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 300 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 301 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 302 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 303 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE ""
-1 304 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 305 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 306 "" 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 }{CSTYLE "" -1 307 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{PSTYLE "Norm
al" -1 0 1 {CSTYLE "" -1 -1 "" 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 }0 0 0
-1 -1 -1 0 0 0 0 0 0 -1 0 }{PSTYLE "Text Output" -1 2 1 {CSTYLE "" -1
-1 "Courier" 1 10 0 0 255 1 0 0 0 0 0 1 3 0 3 }1 0 0 -1 -1 -1 0 0 0 0

0 0 -1 0 }{PSTYLE "Heading 1" 0 3 1 {CSTYLE "" -1 -1 "" 1 18 0 0 0 0
0 1 0 0 0 0 0 0 0 }1 0 0 0 8 4 0 0 0 0 0 0 -1 0 }{PSTYLE "Heading 2"
3 4 1 {CSTYLE "" -1 -1 "" 1 14 0 0 0 0 0 0 0 0 0 0 0 0 }0 0 0 -1 8
2 0 0 0 0 0 0 -1 0 }{PSTYLE "Warning" 2 7 1 {CSTYLE "" -1 -1 "" 0 1 0
0 255 1 0 0 0 0 0 0 1 0 0 }0 0 0 -1 -1 -1 0 0 0 0 0 0 -1 0 }{PSTYLE "M
aple Output" 0 11 1 {CSTYLE "" -1 -1 "" 0 1 0 0 0 0 0 0 0 0 0 0 0 0
}3 3 0 -1 -1 -1 0 0 0 0 0 0 -1 0 }{PSTYLE "" 11 12 1 {CSTYLE "" -1 -1
"" 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 }1 0 0 -1 -1 -1 0 0 0 0 0 0 -1 0 }}
{SECT 0 {EXCHG {PARA 0 "" 0 "" {TEXT -1 0 "" }{TEXT 256 0 "" }{TEXT
257 29 "Tutorial for the gbr5 package" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 32 "By William Gryc, Amherst College" }}{PARA 0 "" 0 ""
{TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 843 "   Well, you finally di
d it. After hours of deliberation and procrastination, you have final
ly downloaded and saved the gbr5 files on your computer, just like you
r professor asked you to. Now you are expecting a long-winded and com
plex tutorial. Well, you aren't getting it here! This tutorial is de
signed to be clear and quick, so you can start working as soon as poss
ible. If you want longer explanations, try looking at the help access
ed by opening the worksheet \"gbr5hlp.mws\" which, hopefully, you also
 downloaded. You'll be sure to find whatever you want to know there. \+
 Also, please be advised that some of the commands in the package are \+
slow compared to regular Maple commands, but they should be adequate i
n computing the simpler examples in \"Ideals, Varieties, and Algorithm
s.\" But, now in the spirit of being quick, onto...." }}}{SECT 0
{PARA 3 "" 0 "" {TEXT -1 26 "The Basics: Loading `gbr5`" }}{PARA 0 ""
0 "" {TEXT -1 344 "Even though you managed to get this tutorial to run
ning, that's not all there is to it. There's still the issue of loadi
ng the package. This is not too terribly difficult, however. Be sure
 you've also downloaded \"gbr5.mpl\", \"gbr5hlp.mws\". Then, assuming
 that you started Maple from the directory containing these files, the
n all you type is" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{EXCHG {PARA 0 ">
" 0 "" {MPLTEXT 1 0 17 "read(`gbr5.mpl`):" }}{PARA 7 "" 1 "" {TEXT
-1 32 "Warning, new definition for norm" }}{PARA 7 "" 1 "" {TEXT -1
33 "Warning, new definition for trace" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 247 "Please note that to use the interactive tutorial, you ju
st have to press Enter (or Return, depending on your keyboard) on each
 Maple input line. After hitting Enter on the previous Maple input li
ne, the package is loaded and we are ready to work." }}}{EXCHG {PARA
0 "" 0 "" {TEXT -1 124 "Also note that reading in \"gbr5.mpl\" generat
es two warning messages concerning norm and trace. These can be safel
y ignored." }}}}{SECT 0 {PARA 3 "" 0 "" {TEXT -1 27 "ring() and Relate
d Commands" }}{PARA 0 "" 0 "" {TEXT -1 567 "The most basic command in \+
the package is ring(). This command sets the ring and the monomial or
der (see Chapter 2, Section 2 of the text) which most other commands i
n the will use. In fact, if you try to use any commands without first
 setting ring(), you'll get a nasty error message saying that you must
 set ring() first. The arguments for ring() are a monomial order and \+
a variable list. The valid monomial orders are lex, grlex, grevlex (s

ee Chapter 2 Section 2), [k,n] (the kth elimination order on n variabl
es; see Exercise 12d in Chapter 2 Section 2), and [" }{TEXT 268 2 "v1
" }{TEXT -1 5 ",...," }{TEXT 269 2 "vn" }{TEXT -1 168 "]. The last or
der is a matrix order and is not discussed in the text explicitly. Sa
y you were working in the ring k[x1,...,xn]. Then a matrix order woul
d consist of " }{TEXT 260 1 "n" }{TEXT -1 31 " linearly independent ve
ctors \{" }{TEXT 258 2 "v1" }{TEXT -1 5 ",...," }{TEXT 259 2 "vn" }
{TEXT -1 17 "\} each of length " }{TEXT 261 2 "n," }{TEXT -1 13 " such
 that x^" }{XPPEDIT 18 0 "alpha;" "6#%&alphaG" }{TEXT -1 5 " > x^" }
{XPPEDIT 18 0 "beta;" "6#%%betaG" }{TEXT -1 5 " iff " }{TEXT 262 2 "v1
" }{TEXT -1 1 "*" }{XPPEDIT 18 0 "alpha;" "6#%&alphaG" }{TEXT -1 3 " >
 " }{TEXT 263 3 "v1*" }{XPPEDIT 18 0 "beta;" "6#%%betaG" }{TEXT -1 4 "
 or " }{TEXT 264 2 "v1" }{TEXT -1 1 "*" }{XPPEDIT 18 0 "alpha;" "6#%&a
lphaG" }{TEXT -1 3 " = " }{TEXT 265 3 "v1*" }{XPPEDIT 18 0 "beta;" "6#
%%betaG" }{TEXT -1 5 " and " }{TEXT 266 2 "v2" }{TEXT -1 1 "*" }
{XPPEDIT 18 0 "alpha;" "6#%&alphaG" }{TEXT -1 3 " > " }{TEXT 267 2 "v2
" }{TEXT -1 1 "*" }{XPPEDIT 18 0 "beta;" "6#%%betaG" }{TEXT -1 351 " ,
 etc. All monomials orders the package deals with can be written as m
atrix orders. As you will see in a moment, this is how the package se
ts monomial orders (except for lex and grevlex, which use the built in
 Maple orders plex and tdeg, respectively). So, say you wanted to set
 the ring k[x,y,z] and use the monomial order grevlex. So, you type \+
" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 23 "ring(grevlex, [x,y,z]);
" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "
" 0 "" {TEXT -1 119 "Then you change your mind and decide on a matrix \+
order with the simplest linearly independant vectors you can think of.
" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 41 "ring([[1,0,0],[0,1,0],[
0,0,1]], [x,y,z]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }
}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 183 "Notice that grevlex was not cap
italized. This is important, as lex, grlex, and grevlex must be enter
ed exactly as shown here, in all lower case letters, for ring() to wor
k correctly." }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT
-1 531 "The ring() commands creates term orders in two ways. For lex \+
and grevlex, it uses the terms orders plex and tdeg from the Groebner \+
package. However, for grlex and elimination orders, ring() uses matri
ces created by the grlex() and elimination() commands. These are the \+
\"related commands\" that the section title refers to. The lone argum
ent for grlex() is the integer that is the number of variables in the \+
ring. So, if we were working in k[x1,x2,x3,x4] and wanted the matrix \+
that yields grevlex order over this ring, we'd type" }}}{EXCHG {PARA
0 "> " 0 "" {MPLTEXT 1 0 9 "grlex(4);" }}{PARA 11 "" 1 "" {XPPMATH 20
"6#7&7&\"\"\"F%F%F%7&F%\"\"!F'F'7&F'F%F'F'7&F'F'F%F'" }}}{EXCHG {PARA
0 "" 0 "" {TEXT -1 206 "Notice that grlex(), as well as elimination(),
 does not depend on the current ring and monomial order set by ring().
   elimination() works differently; it needs one extra argument. The \+
form is elimination(" }{TEXT 270 3 "k,n" }{TEXT -1 9 "), where " }
{TEXT 271 1 "n" }{TEXT -1 44 " is the number of variables in the ring \+
and " }{TEXT 272 1 "k" }{TEXT -1 175 " is the number of variables to e

liminate. So, to get the matrix that gives a monomial order that elim
inates the first three variables of a ring with five variables, you en
ter" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 17 "elimination(3,5);" }
}{PARA 11 "" 1 "" {XPPMATH 20 "6#7'7'\"\"\"F%F%\"\"!F&7'F&F&F&F%F%7'F&
F&F&F&!\"\"7'F&F&F)F&F&7'F&F)F&F&F&" }}}{EXCHG {PARA 0 "" 0 "" {TEXT
-1 25 "And you have your matrix." }}}}{SECT 0 {PARA 3 "" 0 "" {TEXT
-1 22 "The Division Algorithm" }}{PARA 0 "" 0 "" {TEXT -1 482 "One of \+
the major steps in computing Groebner bases is computing remainders vi
a the Division algorithm. The command div_alg() implements this algor
ithm. The two arguments of div_alg are a polynomial and a set of poly
nomials. div_alg returns a list with two elements: the first element
 is the remainder, and the second is a list of quotients, ordered in c
orrespondence with the ordering of the given set of polynomials. Let'
s use Problem 1a in Chapter 2 Section 3 for an example." }}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "ring(grlex, [x,y]);" }}{PARA 11 ""
1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 57 "div_alg(x^7*y^2 + x^3*y^2 - y + 1, [x*y^2 - x, x - y^
3]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7$,**$)%\"xG\"\"(\"\"\"\"\"\"*
$)F\"\"$F)F*%\"yG!\"\"F*F*7$,&*$)F\"\"F)F**$)F\"\"#F)F*\"\"!" }}}
{EXCHG {PARA 0 "" 0 "" {TEXT -1 70 "Notice div_alg() divides with resp
ect to the term order set by ring()." }}}}{SECT 0 {PARA 3 "" 0 ""
{TEXT -1 41 "Computing Groebner Bases without Matrices" }}{PARA 0 ""
0 "" {TEXT -1 465 "This package provides four different ways to comput
e Groebner bases. The first, slowbasis_gb implements a naive version \+
of Buchberger's algorithm (see Chapter 2 Section 2). The parameter of
 this command is a list of polynomials [f1,...,fs] slowbasis_gb then \+
returns a list of polynomials that form a Groebner basis for <f1,...,f
s> with respect to the termorder set by ring. For an example, we'll u
se Exercise 2b from Chapter 2 Section 7. First we set the ring." }}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 17 "ring(lex, [x,y]);" }}{PARA
11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 32 "Then we find the Groebner basis." }}}{EXCHG {PARA 0 "> "
0 "" {MPLTEXT 1 0 49 "slowbasis_gb([x^2 + y, x^4 + 2*x^2*y + y^2 + 3])
;" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%0Current~basis:~G7$,&*$)%\"xG\"
\"#\"\"\"\"\"\"%\"yGF,,**$)F)\"\"%F+F,*&F(F+F-F,F**$)F-F*F+F,\"\"$F,"
}}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G7#!\"$" }}{PARA 11 "" 1 "
" {XPPMATH 20 "6#%3Local~divisions:~1G" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#(%(Added:~G7\"" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Local~divisi
ons:~2G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%=Total~divisions~performed
:~3G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7%,&*$)%\"xG\"\"#\"\"\"\"\"%
\"yGF*,**$)F\"\"%F)F**&F&F)F+F*F(*$)F+F(F)F*\"\"$F*!\"$" }}}{EXCHG
{PARA 0 "" 0 "" {TEXT -1 165 "Notice that the steps of calculation wer
e also printed out, as well as how many divisions were performed. To \+
prevent this, just type \"nosteps\" as a second argument." }}}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 58 "slowbasis_gb([x^2 + y, x^4 + 2*x^2*
y + y^2 + 3], nosteps);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7%,&*$)%\"x
G\"\"#\"\"\"\"\"%\"yGF*,**$)F\"\"%F)F**&F&F)F+F*F(*$)F+F(F)F*\"\"$F
*!\"$" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 67 "Sometimes slowbasis_gb g

ets out of hand. Consider this example:. " }}}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 23 "ring(grevlex, [x,y,z]);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1
0 50 "basisprime := [x^3 - z^2, y^3 + z, x^2*y + x*y^2];" }}{PARA 11 "
" 1 "" {XPPMATH 20 "6#>%+basisprimeG7%,&*$)%\"xG\"\"$\"\"\"\"\"\"*$)%
\"zG\"\"#F+!\"\",&*$)%\"yGF*F+F,F/F,,&*&)F)F0F+F5F,F,*&F)F,)F5F0F+F,"
}}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 25 "slowbasis_gb(basisprime);
" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%0Current~basis:~G7%,&*$)%\"xG\"
\"$\"\"\"\"\"\"*$)%\"zG\"\"#F+!\"\",&*$)%\"yGF*F+F,F/F,,&*&)F)F0F+F5F,
F,*&F)F,)F5F0F+F," }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G7$,&*&
%\"yG\"\"\")%\"zG\"\"#\"\"\"!\"\"*&%\"xGF)F+F)F.,&*&)F0F,F-F+F-F)*(F(F
-F0F-F+F-F)" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Local~divisions:~3G"
}}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G7&,&*$)%\"zG\"\"$\"\"\"!
\"\"*(F)\"\"\"%\"xGF.)%\"yG\"\"#F+F.F&,&F'F.F-F,F&" }}{PARA 11 "" 1 "
" {XPPMATH 20 "6#%3Local~divisions:~7G" }}{PARA 12 "" 1 "" {XPPMATH
20 "6#(%(Added:~G7.,&*&%\"xG\"\"\")%\"zG\"\"$\"\"\"F)*&F(F-)F+\"\"#F-!
\"\"F&,&F'F1F.F)F&,&*$)F+\"\"%F-F1*$F*F-F)F3,&F4F)F7F1F3F&F&F2F&" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#%4Local~divisions:~26G" }}{PARA 11 ""
1 "" {XPPMATH 20 "6#(%(Added:~G7\"" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#
%5Local~divisions:~174G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%?Total~div
isions~performed:~210G" }}{PARA 12 "" 1 "" {XPPMATH 20 "6#77,&*$)%\"xG
\"\"$\"\"\"\"\"\"*$)%\"zG\"\"#F)!\"\",&*$)%\"yGF(F)F*F-F*,&*&)F'F.F)F3
F*F**&F'F*)F3F.F)F*,&*&F3F)F,F)F/*&F'F)F-F*F/,&*&F6F)F-F)F**(F3F)F'F)F
-F)F*,&*$)F-F(F)F/*(F-F)F'F)F8F)F*F?,&F@F*FBF/F?,&*&F'F)FAF)F**&F'F)F,
F)F/FD,&FEF/FFF*FD,&*$)F-\"\"%F)F/F@F*FH,&FIF*F@F/FHFDFDFGFD" }}}
{EXCHG {PARA 0 "" 0 "" {TEXT -1 338 "The returned basis is quite unnec
essarily repetitive. The reason for this repetition is that the algor
ithm is using G' as opposed to G is division (see Buchberger's algorit
hm in Chapter 2 Section 7). A second command, altbasis_gb uses G inst
ead. Notice that the repetion is now gone, and the number of division
s decreases dramatically." }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0
24 "altbasis_gb(basisprime);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%0Cur
rent~basis:~G7%,&*$)%\"xG\"\"$\"\"\"\"\"\"*$)%\"zG\"\"#F+!\"\",&*$)%\"
yGF*F+F,F/F,,&*&)F)F0F+F5F,F,*&F)F,)F5F0F+F," }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#(%(Added:~G7$,&*&%\"yG\"\"\")%\"zG\"\"#\"\"\"!\"\"*&%\"
xGF)F+F)F.,&*&)F0F,F-F+F-F)*(F(F-F0F-F+F-F)" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%3Local~divisions:~3G" }}{PARA 11 "" 1 "" {XPPMATH 20 "
6#(%(Added:~G7#,&*$)%\"zG\"\"$\"\"\"!\"\"*(F)\"\"\"%\"xGF.)%\"yG\"\"#F
+F." }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Local~divisions:~7G" }}{PARA
11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G7$,&*&%\"xG\"\"\")%\"zG\"\"$\"\"
\"F)*&F(F-)F+\"\"#F-!\"\",&*$)F+\"\"%F-F1*$F*F-F)" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%3Local~divisions:~5G" }}{PARA 11 "" 1 "" {XPPMATH 20 "
6#(%(Added:~G7\"" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%4Local~divisions:
~13G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%>Total~divisions~performed:~2
8G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7*,&*$)%\"xG\"\"$\"\"\"\"\"\"*$)
%\"zG\"\"#F)!\"\",&*$)%\"yGF(F)F*F-F*,&*&)F'F.F)F3F*F**&F'F*)F3F.F)F*,
&*&F3F)F,F)F/*&F'F)F-F*F/,&*&F6F)F-F)F**(F3F)F'F)F-F)F*,&*$)F-F(F)F/*(
F-F)F'F)F8F)F*,&*&F'F)FAF)F**&F'F)F,F)F/,&*$)F-\"\"%F)F/F@F*" }}}

{EXCHG {PARA 0 "" 0 "" {TEXT -1 279 "While altbasis_gb() can dramatica
lly improve performance, we can still do better. The quickbasis_gb() \+
command implements the improvements to Buchberger's algorithm as detai
led in Chaptert 2 Section 9. Let's use two examples to show how quick
basis_gb can outperform altbasis_gb:" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 21 "basis1 := [x^2, y^4];" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#>%'basis1G7$*$)%\"xG\"\"#\"\"\"*$)%\"yG\"\"%F*" }}}{EXCHG {PARA
0 "> " 0 "" {MPLTEXT 1 0 20 "altbasis_gb(basis1);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#(%0Current~basis:~G7$*$)%\"xG\"\"#\"\"\"*$)%\"yG\"\"%F*
" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G7\"" }}{PARA 11 "" 1 "
" {XPPMATH 20 "6#%3Local~divisions:~1G" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#%=Total~divisions~performed:~1G" }}{PARA 11 "" 1 "" {XPPMATH 20
"6#7$*$)%\"xG\"\"#\"\"\"*$)%\"yG\"\"%F(" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 22 "quickbasis_gb(basis1);" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#(%0Current~Basis:~G7$*$)%\"xG\"\"#\"\"\"*$)%\"yG\"\"%F*" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#%=Total~divisions~performed:~0G" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#7$*$)%\"xG\"\"#\"\"\"*$)%\"yG\"\"%F("
}}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 283 "Clearly, basis1 is already a G
roebner basis by definition, and yet altbasis_gb() still performs a di
vision. quickbasis_gb() cuts out this division by noticing that x^2 a
nd y^4 are relatively prime, and thus no divisons need to be performed
 (see Propostion 4 of Chapter 2 Section 9)." }}}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 32 "basis2 := [x^2, x*y^3, x^2*y^3];" }}{PARA 11 "" 1
"" {XPPMATH 20 "6#>%'basis2G7%*$)%\"xG\"\"#\"\"\"*&F(\"\"\")%\"yG\"\"$
F**&F'F*F-F*" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 20 "altbasis_gb
(basis2);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%0Current~basis:~G7%*$)%
\"xG\"\"#\"\"\"*&F(\"\"\")%\"yG\"\"$F**&F'F*F-F*" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#(%(Added:~G7\"" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Loc
al~divisions:~3G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%=Total~divisions~
performed:~3G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7%*$)%\"xG\"\"#\"\"\"
&F&\"\"\")%\"yG\"\"$F(&F%F(F+F(" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 22 "quickbasis_gb(basis2);" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#(%0Current~Basis:~G7%*$)%\"xG\"\"#\"\"\"*&F(\"\"\")%\"yG\"\"$F**
&F'F*F-F*" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%=Total~divisions~perform
ed:~2G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7%*$)%\"xG\"\"#\"\"\"*&F&\"
\"\")%\"yG\"\"$F(*&F%F(F+F(" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 319 "A
gain, quickbasis_gb() is able to cut out a division, even though none \+
of the leading terms of basis2 are relatively prime. However, notice \+
that the third leading term, x^2y^3, divides the LCM of x^2 and xy^3 (
in fact, x^2y^3 is the LCM of x^2 and xy^3). By Propostion 10 of Chap
ter 2 Section 9, if the remainders of " }{TEXT 273 1 "S" }{TEXT -1 16
"(x^2, xy^3) and " }{TEXT 274 1 "S" }{TEXT -1 52 "(x^2, x^2y^3) are bo
th calculated, the remainder of " }{TEXT 275 1 "S" }{TEXT -1 100 "(xy^
3, x^2y^3) need not be calculated. Thus, quickbasis_gb is implementin
g this second improvement." }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0
"" 0 "" {TEXT -1 198 "It should be noted, however, that Maple's built-
in gbasis() command is faster than quickbasis_gb(). However, quickbas
is_gb() should be fast enough for most of the examples and problems in

 the text." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 38 "Minimizing and Red
ucing Groebner bases" }}{PARA 0 "" 0 "" {TEXT -1 160 "For added conven
ience, there are also commands to minimize and reduce your computed Gr
oebner bases. For an example, let's use Problem 9 of Chapter 2 Sectio
n 7." }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 21 "ring(lex, [x,y,z,w])
;" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0
"> " 0 "" {MPLTEXT 1 0 61 "basis := [3*x - 6*y - 2*z, 2*x - 4*y + 4*w,
 x - 2*y - z - w];" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%&basisG7%,(%\"
xG\"\"$%\"yG!\"'%\"zG!\"#,(F'\"\"#F)!\"%%\"wG\"\"%,*F'\"\"\"F)F,F+!\"
\"F0F4" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 44 "basisprime := qui
ckbasis_gb(basis, nosteps);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%+basi
sprimeG7&,(%\"xG\"\"$%\"yG!\"'%\"zG!\"#,(F'\"\"#F)!\"%%\"wG\"\"%,*F'\"
\"\"F)F,F+!\"\"F0F4,&F+F/F0!#7" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1
208 "To get a minimal Groebner basis from this, we can use the min_gb(
) command. min_gb() takes a list of polynomials that form a Groebner \+
basis under ring() as its lone argument. So, to make basisprime minim
al," }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 32 "basisprime :=min_gb(
basisprime);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%+basisprimeG7$,*%\"x
G\"\"\"%\"yG!\"#%\"zG!\"\"%\"wGF,,&F+!\"%F-!#7" }}}{EXCHG {PARA 0 ""
0 "" {TEXT -1 336 "Getting a reduced Groebner basis from this new basi
sprime, we use the red_gb command. red_gb() takes a list of polynomia
ls that for a MINIMAL Groebner basis under ring() as its argument. If
 the basis isn't a minimal Groebner basis, red_gb() will protest and w
ill refuse to do its job. So, to make basisprime a reduced Groebner b
asis," }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "red_gb(basisprime)
;" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7$,(%\"xG\"\"\"%\"yG!\"#%\"wG\"\"
#,&%\"zGF&F)\"\"$" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 34 "And the Groe
bner basis is reduced." }}}}}{SECT 0 {PARA 3 "" 0 "" {TEXT -1 38 "Comp
uting Groebner Bases with Matrices" }}{PARA 0 "" 0 "" {TEXT -1 60 "Con
sider the following type of problem: You have an ideal <" }{TEXT 284
9 "f1,...,fs" }{TEXT -1 40 "> and compute a Groebner basis for it, \{
" }{TEXT 285 9 "g1,...,gt" }{TEXT -1 12 "\}. Since <" }{TEXT 286 9 "
f1,...,fs" }{TEXT -1 5 "> = <" }{TEXT 287 9 "g1,...,gt" }{TEXT -1 12 "
>, for each " }{TEXT 288 2 "fi" }{TEXT -1 15 ", there exists " }{TEXT
289 9 "a1,...,at" }{TEXT -1 11 " such that " }{TEXT 290 2 "fi" }{TEXT
-1 3 " = " }{TEXT 291 4 "a1g1" }{TEXT -1 3 " + " }{TEXT 292 4 "a2g2" }
{TEXT -1 9 " + ... + " }{TEXT 293 4 "atgt" }{TEXT -1 65 ", and vice ve
rsa. For the first problem, it is easy to find the " }{TEXT 294 9 "a1
,...,at" }{TEXT -1 8 " since \{" }{TEXT 296 8 "g1,..,gt" }{TEXT -1 55
"\} is a Groebner basis the division algorithm gives the " }{TEXT 295
2 "ai" }{TEXT -1 301 ". There is a function in the package, quot_mx()
, that does exactly this. quot_mx() takes two arguments. The first i
s a list of polynomials that generate an ideal, the second is a Groebn
er basis under the termorder set in ring() that generates the same ide
al. The output is a matrix Q where Q is a " }{TEXT 303 1 "s" }{TEXT
-1 3 " x " }{TEXT 304 1 "t" }{TEXT -1 12 " matrix and " }}{PARA 0 ""
0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 297 2 "f1"
}{TEXT -1 15 "]   [" }{TEXT 300 2 "g1" }{TEXT -1 1 "]" }}

```
{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 298 2 "f2" }{TEXT -1 15 "]      \+
      [" }{TEXT 301 2 "g2" }{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT
-1 20 "[...]    = Q * [...]" }}{PARA 0 "" 0 "" {TEXT -1 24 "[...]     \+
        [...]" }}{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 299 2 "fs" }
{TEXT -1 16 "]             [" }{TEXT 302 2 "gt" }{TEXT -1 1 "]" }}
{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 31 "Consider \+
the following example:" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 23 "ri
ng(grevlex, [x,y,z]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG
" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 52 "gbase := quickbasis_gb(
[x - z^4, y - z^5], nosteps);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%&gb
aseG7(,&%\"xG\"\"\"*$)%\"zG\"\"%\"\"\"!\"\",&%\"yGF(*$)F+\"\"&F-F.,&*&
F+F(F'F(F.F0F(,&*$)F'\"\"#F-F.*&)F+\"\"$F-F0F(F(,&*&)F+F9F-)F0F9F-F(*$
)F'F<F-F.,&*&)F0F<F-F+F-F(*$)F'F,F-F." }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 40 "Q := quot_mx([x - z^4, y - z^5], gbase);" }}{PARA 11
"" 1 "" {XPPMATH 20 "6#>%\"QG-%'matrixG6#7$7(\"\"\"\"\"!F+F+F+F+7(%\"z
GF+F*F+F+F+" }}}{PARA 0 "" 0 "" {TEXT -1 133 "Notice that Q is not uni
que, as the most obvious choice for Q is not the matrix given.  Let's \+
verify that Q has the desired property." }}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 26 "base := matrix(6,1,gbase);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#>%%baseG-%'matrixG6#7(7#,&%\"xG\"\"\"*$)%\"zG\"\"%\"\"
\"!\"\"7#,&%\"yGF,*$)F/\"\"&F1F27#,&*&F/F,F+F,F2F5F,7#,&*$)F+\"\"#F1F2
*&)F/\"\"$F1F5F,F,7#,&*&)F/F@F1)F5F@F1F,*$)F+FCF1F27#,&*&)F5FCF1F/F1F,
*$)F+F0F1F2" }}}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 27 "simplify(multiply(Q,base));" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#-%'matrixG6#7$7#,&%\"xG\"\"\"*$)%\"zG\"\"%\"\"\"!\"\"7#
,&%\"yGF**$)F-\"\"&F/F0" }}}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "
" 0 "" {TEXT -1 43 "The opposite question of how to represent \{" }
{TEXT 305 9 "g1,...,gt" }{TEXT -1 6 "\} in <" }{TEXT 306 9 "f1,...,fs
" }{TEXT -1 133 "> is trickier, since the division algorithm can't do \+
the job in this case.  However, you can compute the matrix while compu
ting the \{" }{TEXT 307 8 "g1,..,gt" }{TEXT -1 321 "\}. Computing Groe
bner bases with matrices isn't a whole lot different than computing th
em without matrices (that is, from the user's standpoint).  You still \+
use the ring() command to set the term-ordering.  However, you just us
e one command, mxgb(), to compute a reduced Groebner basis and its mat
rix M.  The matrix M an " }{TEXT 277 1 "t" }{TEXT -1 3 " x " }{TEXT
276 1 "s" }{TEXT -1 17 " matix such that:" }}{PARA 0 "" 0 "" {TEXT -1
0 "" }}{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 278 2 "g1" }{TEXT -1 15 "
]            [" }{TEXT 281 2 "f1" }{TEXT -1 1 "]" }}{PARA 0 "" 0 ""
{TEXT -1 1 "[" }{TEXT 279 2 "g2" }{TEXT -1 15 "]            [" }
{TEXT 282 2 "f2" }{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT -1 20 "[...] \+
  = M * [...]" }}{PARA 0 "" 0 "" {TEXT -1 24 "[...]             [...
]" }}{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 280 2 "gt" }{TEXT -1 16 "] \+
            [" }{TEXT 283 2 "fs" }{TEXT -1 1 "]" }}{PARA 0 "" 0 ""
{TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 278 "mxgb() takes a list of p
olynomials as its argument.  If you do not desire to see the basis and
 matrix at each \"step\" (at the unminimized step, and the unreduced s
teps), a second argument of \"nosteps\" should be added.  Let's use Pr
```

oblem 2c of Chapter 2 Section 7 as an example. " }}{EXCHG {PARA 0 "> \+
" 0 "" {MPLTEXT 1 0 25 "ring(grevlex, [x, y, z]);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1
0 35 "mxbase := mxgb([x - z^4, y - z^5]);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6$%4Unminimized~basis:~G7(,&%\"xG\"\"\"*$)%\"zG\"\"%\"\"
\"!\"\",&%\"yGF'*$)F*\"\"&F,F-,&*&F*F'F&F'F-F/F',&*$)F&\"\"#F,F-*&)F*
\"\"$F,F/F'F',&*&)F*F8F,)F/F8F,F'*$)F&F;F,F-,&*&)F/F;F,F*F,F'*$)F&F+F,
F-" }}{PARA 11 "" 1 "" {XPPMATH 20 "6$%5Unminimized~matrix:~G-%'matrix
G6#7(7$\"\"\"\"\"!7$F*F)7$,$%\"zG!\"\"F)7$,&%\"xGF/*$)F.\"\"%\"\"\"F/*
$)F.\"\"$F67$,(*&F8F6%\"yGF)F/*$)F2\"\"#F6F/*&F4F6F2F)F/,&*&)F.F@F6F=F
6F)*&F2F6F8F6F)7$,**&FDF6)F=F@F6F/*(F2F6F8F6F=F6F/*$)F2F9F6F/*&F4F6F?F
6F/,(*&FIF6F.F)F)*(F2F6FDF6F=F6F)*&F?F6F8F6F)" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6$%2Minimized~basis:~G7',&%\"xG\"\"\"*$)%\"zG\"\"%\"\"\"!
\"\",&*&F*F'F&F'F-%\"yGF',&*$)F&\"\"#F,F-*&)F*\"\"$F,F0F'F',&*&)F*F4F,
)F0F4F,F'*$)F&F7F,F-,&*&)F0F7F,F*F,F'*$)F&F+F,F-" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6$%3Minimized~matrix:~G-%'matrixG6#7'7$\"\"\"\"\"!7$,$%\"
zG!\"\"F)7$,&%\"xGF.*$)F-\"\"%\"\"\"F.*$)F-\"\"$F57$,(*&F7F5%\"yGF)F.*
$)F1\"\"#F5F.*&F3F5F1F)F.,&*&)F-F?F5F<F5F)*&F1F5F7F5F)7$,**&FCF5)F<F?F
5F.*(F1F5F7F5F<F5F.*$)F1F8F5F.*&F3F5F>F5F.,(*&FHF5F-F)F)*(F1F5FCF5F<F5
F)*&F>F5F7F5F)" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%'mxbaseG7$7',&%\"x
G!\"\"*$)%\"zG\"\"%\"\"\"\"\",&*&F,F/F(F/F/%\"yGF),&*$)F(\"\"#F.F)*&
)F,\"\"$F.F2F/F/,&*&)F,F6F.)F2F6F.F/*$)F(F9F.F),&*&)F2F9F.F,F.F)*$)F(F
-F.F/-%'matrixG6#7'7$F)\"\"!7$F,F)7$,&F(F)F*F)*$F8F.7$,(F7F)F4F)*&F+F.
F(F.F),&*&F<F.F2F.F/*&F(F.F8F.F/7$,*F;F/*(F(F.F8F.F2F.F/F>F/*&F+F.F5F.
F/,(*&F=F.F,F.F)*(F(F.F<F.F2F.F)*&F5F.F8F.F)" }}}{EXCHG {PARA 0 "" 0 "
" {TEXT -1 46 "Let's see if this matrix is what we say it is." }}}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 37 "base := matrix(2,1,[x-z^4, y
 - z^5]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%%baseG-%'matrixG6#7$7#,
&%\"xG\"\"\"*$)%\"zG\"\"%\"\"\"!\"\"7#,&%\"yGF,*$)F/\"\"&F1F2" }}}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 36 "simplify(multiply(mxbase[2],
 base));" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#-%'matrixG6#7'7#,&%\"xG!\"
\"*$)%\"zG\"\"%\"\"\"\"\"\"7#,&*&F-F0F)F0F0%\"yGF*7#,&*$)F)\"\"#F/F**&
)F-\"\"$F/F4F0F07#,&*&)F-F9F/)F4F9F/F0*$)F)F<F/F*7#,&*&)F4F<F/F-F/F**$
)F)F.F/F0" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 0 "" }}}{EXCHG
{PARA 0 "" 0 "" {TEXT -1 112 "Since this matrix gives the element of t
he Groebner basis, the matrix given by mxgb() is what we claimed it wa
s." }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 349 "As
 you may have noticed, there are quickbasis_mxgb, min_mxgb, and red_mx
gb commands which have counterparts that do not have the \"mx\" prefix
. All these commands are combined for the mxgb command, and are not m
eant for users. This is not to say that you cannot use them, but thes
e commands aren't terribly user friendly and we discourage their use.
" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 143 "That
's it for this tutorial. Remember, more information on these commands
 can be found below in the reference guide. Thank you and good luck!
" }}}}{SECT 0 {PARA 3 "" 0 "" {TEXT -1 16 "Acknowledgements" }}{PARA
0 "" 0 "" {TEXT -1 117 "Will Gryc and David Cox would like to thank th
e Charleton Trust for supporting Will's work on this Maple worksheet. \+

" }}}}{MARK "0" 0 }{VIEWOPTS 1 1 0 3 2 1804 }

```
{VERSION 3 0 "SUN SPARC SOLARIS" "3.0" }
{USTYLETAB {CSTYLE "Maple Input" -1 0 "Courier" 0 1 255 0 0 1 0 1 0 0
1 0 0 0 0 }{CSTYLE "2D Math" -1 2 "Times" 0 1 0 0 0 0 0 0 2 0 0 0 0 0
0 }{CSTYLE "2D Output" 2 20 "" 0 1 0 0 255 1 0 0 0 0 0 0 0 0 0 }
{CSTYLE "" -1 256 "" 1 24 0 0 0 0 0 0 0 0 0 0 0 0 0 }{CSTYLE "" -1
257 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 258 "" 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 259 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0
0 }{CSTYLE "" -1 260 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1
261 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 }{CSTYLE "" -1 262 "" 0 1 0 0 0
0 0 1 0 0 0 0 0 0 0 }{CSTYLE "" -1 263 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 }{CSTYLE "" -1 264 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1
265 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 266 "" 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 267 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0
0 }{CSTYLE "" -1 268 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1
269 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 270 "" 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 271 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0
0 }{CSTYLE "" -1 272 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1
273 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 274 "" 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 }{CSTYLE "" -1 275 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0
0 }{PSTYLE "Normal" -1 0 1 {CSTYLE "" -1 -1 "" 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 }0 0 0 -1 -1 -1 0 0 0 0 0 0 -1 0 }{PSTYLE "Heading 1" 0 3 1
{CSTYLE "" -1 -1 "" 1 18 0 0 0 0 0 1 0 0 0 0 0 0 0 }1 0 0 0 8 4 0 0 0
0 0 0 -1 0 }{PSTYLE "Heading 2" 3 4 1 {CSTYLE "" -1 -1 "" 1 14 0 0 0
0 0 0 0 0 0 0 0 0 0 }0 0 0 -1 8 2 0 0 0 0 0 0 -1 0 }{PSTYLE "Maple Out
put" 0 11 1 {CSTYLE "" -1 -1 "" 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 }3 3 0
-1 -1 -1 0 0 0 0 0 0 -1 0 }{PSTYLE "" 11 12 1 {CSTYLE "" -1 -1 "" 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 }1 0 0 -1 -1 -1 0 0 0 0 0 0 -1 0 }}
{SECT 0 {EXCHG {PARA 0 "" 0 "" {TEXT -1 0 "" }{TEXT 256 41 "Reference \+
worksheet for the gbr5 package " }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1
404 "The gbr5 package provides commands to compute Grobner bases which
 also can show the steps involved in computing them.  The major comman
ds in this package are listed below in order of need to know (i.e., th
e most basic command is first, followed by the next most basic command
, etc).  Maximize the command you would like to read about.  (To maxim
ize a command, click on the plus sign next to the command.)" }}}{SECT
1 {PARA 3 "" 0 "" {TEXT -1 37 "General Information about the Package"
}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}{PARA 0 "
" 0 "" {TEXT -1 16 "<function>(args)" }}}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 38 "The functions in \+
the gbr5 package are:" }}{PARA 0 "" 0 "" {TEXT -1 65 " ring          l
ex\011\011      grlex\011\011       grevlex       elimination" }}
{PARA 0 "" 0 "" {TEXT -1 110 " slowbasis_gb\011        altbasis_gb\011
           quickbasis_gb\011\n\011div_alg\011\011               quot_m
x\011\011            mxgb" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "
" 0 "" {TEXT -1 45 "This package uses the global variable morder." }}
{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 144 "Before a
ny of slowbasis_gb, altbasis_gb, quickbasis_gb, mxgb, quot_mx, or  div
```

_alg can be used, ring must be performed (see ring() for details)." }}
}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 6 "ring()" }}{SECT 0 {PARA 4 "" 0 "
" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 69 "ring() sets the \+
termorder and variables for the package to work under" }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}{PARA 0 "" 0 ""
{TEXT -1 21 "ring (torder,varlist)" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 68 "torder = the monomia
l order. Valid values are lex, grlex, grevlex, " }}{PARA 0 "" 0 ""
{TEXT -1 1 "[" }{TEXT 257 1 "k" }{TEXT -1 1 "," }{TEXT 258 1 "n" }
{TEXT -1 51 "] (the elimination order that eliminates the first " }
{TEXT 259 1 "k" }{TEXT -1 4 " of " }{TEXT 260 1 "n" }{TEXT -1 18 " var
iables), and [" }{TEXT 261 2 "v1" }{TEXT -1 4 ",..," }{TEXT 262 2 "vn
" }{TEXT -1 25 "] (a matrix order, where " }{TEXT 263 2 "vi" }{TEXT
-1 10 " is a 1 x " }{TEXT 264 1 "n" }{TEXT -1 13 " row vector)." }}
{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 121 "varlist \+
= a list of the variables of the ring. Note that if an elimination or
der or matrix order is used, there must be " }{TEXT 265 1 "n" }{TEXT
-1 22 " variables in varlist." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "
Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 84 "ring(torder, varlist) returns
 the term_order with respect to the torder and varlist." }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 21 "ring(grlex, [x,y,z]);" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#%+term_orderG" }}}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 7 "grlex()
" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 ""
{TEXT -1 57 "Creates a matrix whose row vectors produce a grlex order.
" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequence" }}{PARA 0
"" 0 "" {TEXT -1 8 "grlex(n)" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "
Parameters" }}{PARA 0 "" 0 "" {TEXT -1 32 " n = number of variables in
 ring" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 ""
0 "" {TEXT -1 6 "grlex(" }{TEXT 269 1 "n" }{TEXT -1 36 ") returns a li
st which represents a " }{TEXT 266 1 "n" }{TEXT -1 3 " x " }{TEXT 267
1 "n" }{TEXT -1 74 " matrix whose rows produce a matrix order which is
 equivalent to grlex on " }{TEXT 268 1 "n" }{TEXT -1 11 " variables."
}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> \+
" 0 "" {MPLTEXT 1 0 9 "grlex(6);" }}{PARA 12 "" 1 "" {XPPMATH 20 "6#7(
7(\"\"\"F%F%F%F%F%7(F%\"\"!F'F'F'F'7(F'F%F'F'F'F'7(F'F'F%F'F'F'7(F'F'F
'F%F'F'7(F'F'F'F'F%F'" }}}}{PARA 4 "" 0 "" {TEXT -1 0 "" }}}{SECT 1
{PARA 3 "" 0 "" {TEXT -1 13 "elimination()" }}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 64 "Creates a matrix w
hose row vectors produce an elimination order." }}}{SECT 0 {PARA 4 ""
0 "" {TEXT -1 16 "Calling Sequence" }}{PARA 0 "" 0 "" {TEXT -1 17 "eli
mination(k,n);" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}
{PARA 0 "" 0 "" {TEXT -1 40 "k = the number of variables to eliminate
" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 39 "n = t
he number of variables in the ring" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 12 "elimination(" }{TEXT
274 1 "k" }{TEXT -1 1 "," }{TEXT 270 1 "n" }{TEXT -1 36 ") returns a l
ist which represents a " }{TEXT 271 1 "n" }{TEXT -1 3 " x " }{TEXT

272 1 "n" }{TEXT -1 113 " matrix whose rows produce a matrix order whi
ch is equivalent to the elimination order that eliminates the first "
}{TEXT 273 1 "k" }{TEXT -1 4 " of " }{TEXT 275 1 "n" }{TEXT -1 11 " va
riables." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Example" }}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 17 "elimination(3,5);" }}{PARA 11 "" 1
"" {XPPMATH 20 "6#7'7\"\"\"F%F%\"\"!F&7'F&F&F&F%F%7'F&F&F&F&!\"\"7'F&
F&F)F&F&7'F&F)F&F&F&" }}}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 14 "slowba
sis_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 ""
0 "" {TEXT -1 58 "slowbasis_gb() finds a Groebner basis for the given \+
ideal." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}
{PARA 0 "" 0 "" {TEXT -1 26 "slowbasis_gb([f1,...,fs]);" }}{PARA 0 ""
0 "" {TEXT -1 35 "slowbasis_gb([f1,...,fs], nosteps];" }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1
79 "[f1,...,fs] = a list of polynomials in the ring defined by ring()
\n   " }}{PARA 0 "" 0 "" {TEXT -1 67 "nosteps = the string th
at indicates that no steps are to be printed" }}}{SECT 0 {PARA 4 "" 0
"" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 229 "slowbasis_gb(
[f1,...,fs]) returns a Groebner basis of <f1,...,fs> using a naive ver
sion of Buchberger's algorithm. This basis is generally neither minim
al nor reduced. Steps of constructing the Groebner basis are also pri
nted.\n" }}{PARA 0 "" 0 "" {TEXT -1 121 "slowbasis_gb([f1,...,fs], nos
teps) returns the same things, but steps of constructing the Groebner \+
basis are not printed." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Example
s" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "ring(grlex, [x,y]);" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "> "
0 "" {MPLTEXT 1 0 37 "slowbasis_gb([x^2*y - 1, x*y^2 - x]);" }}{PARA
11 "" 1 "" {XPPMATH 20 "6#(%0Current~basis:~G7$,&*&)%\"xG\"\"#\"\"\"%
\"yG\"\"\"F-!\"\"F-,&*&F)F-)F,F*F+F-F)F." }}{PARA 11 "" 1 "" {XPPMATH
20 "6#(%'Added~G7#,&%\"yG!\"\"*$)%\"xG\"\"#\"\"\"\"\"\"" }}{PARA 11 "
" 1 "" {XPPMATH 20 "6#%3Local~divisions:~1G" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#(%'Added~G7$,&*$)%\"yG\"\"#\"\"\"\"\"\"!\"\"F,,&*$)F)\"
\"$F+F,F)F-" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Local~divisions:~2G"
}}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%'Added~G7\"" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%3Local~divisions:~7G" }}{PARA 11 "" 1 "" {XPPMATH 20 "
6#%>Total~divisions~performed:~10G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#
7',&*&)%\"xG\"\"#\"\"\"%\"yG\"\"\"F+!\"\"F+,&*&F'F+)F*F(F)F+F'F,,&F*F,
$F&F)F+,&*$F/F)F+F,F+,&$)F*\"\"$F)F+F*F," }}}{EXCHG {PARA 0 "> " 0 "
" {MPLTEXT 1 0 46 "slowbasis_gb([x^2*y - 1, x*y^2 - x], nosteps);" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#7',&*&)%\"xG\"\"#\"\"\"%\"yG\"\"\"F+!
\"\"F+,&*&F'F+)F*F(F)F+F'F,,&F*F,*$F&F)F+,&*$F/F)F+F,F+,&*$)F*\"\"$F)F
+F*F," }}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}{PARA 0 ""
0 "" {TEXT -1 46 "ring(), altbasis_gb(), quickbasis_gb(), mxgb()" }}}}
{SECT 1 {PARA 3 "" 0 "" {TEXT -1 13 "altbasis_gb()" }}{SECT 0 {PARA 4
"" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 56 "altbasis_g
b() finds a Groebner basis for the given ideal" }}}{SECT 0 {PARA 4 ""
0 "" {TEXT -1 17 "Calling Sequences" }}{PARA 0 "" 0 "" {TEXT -1 25 "al
tbasis_gb([f1,...,fs]);" }}{PARA 0 "" 0 "" {TEXT -1 34 "altbasis_gb([f
1,...,fs], nosteps];" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Paramete

rs" }}{PARA 0 "" 0 "" {TEXT -1 79 "[f1,...,fs] = a list of polynomial
s in the ring defined by ring()\n   " }}{PARA 0 "" 0 ""
{TEXT -1 67 "nosteps = the string that indicates that no steps are to \+
be printed" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA
0 "" 0 "" {TEXT -1 267 "altbasis_gb([f1,...,fs]) returns a Groebner ba
sis of <f1,...,fs> using a slightly more insightful version of Buchber
ger's algorithm that slowbasis_gb(). This basis is generally neither \+
minimal nor reduced. Steps of constructing the Groebner basis are als
o printed.\n" }}{PARA 0 "" 0 "" {TEXT -1 120 "altbasis_gb([f1,...,fs],
 nosteps) returns the same things, but steps of constructing the Groeb
ner basis are not printed." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Exa
mples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "ring(grlex, [x,y]);
" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "
> " 0 "" {MPLTEXT 1 0 36 "altbasis_gb([x^2*y - 1, x*y^2 - x]);" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#(%0Current~basis:~G7$,&*&)%\"xG\"\"#\"
\"\"%\"yG\"\"\"F-!\"\"F-,&*&F)F-)F,F*F+F-F)F." }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#(%(Added:~G7#,&%\"yG!\"\"*$)%\"xG\"\"#\"\"\"\"\"\"" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Local~divisions:~1G" }}{PARA 11 ""
1 "" {XPPMATH 20 "6#(%(Added:~G7#,&*$)%\"yG\"\"#\"\"\"\"\"\"!\"\"F," }
}{PARA 11 "" 1 "" {XPPMATH 20 "6#%3Local~divisions:~2G" }}{PARA 11 ""
1 "" {XPPMATH 20 "6#(%(Added:~G7\"" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#
%3Local~divisions:~3G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%=Total~divis
ions~performed:~6G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7&,&*&)%\"xG\"\"
#\"\"\"%\"yG\"\"\"F+!\"\"F+,&*&F'F+)F*F(F)F+F'F,,&F*F,*$F&F)F+,&*$F/F)
F+F,F+" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 45 "altbasis_gb([x^2*
y - 1, x*y^2 - x], nosteps);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7&,&*&
)%\"xG\"\"#\"\"\"%\"yG\"\"\"F+!\"\"F+,&*&F'F+)F*F(F)F+F'F,,&F*F,*$F&F)
F+,&*$F/F)F+F,F+" }}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}
{PARA 0 "" 0 "" {TEXT -1 47 "ring(), slowbasis_gb(), quickbasis_gb(), \+
mxgb()" }}}{PARA 4 "" 0 "" {TEXT -1 0 "" }}}{SECT 1 {PARA 3 "" 0 ""
{TEXT -1 15 "quickbasis_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Pu
rpose" }}{PARA 0 "" 0 "" {TEXT -1 58 "quickbasis_gb() finds a Groebner
 basis for the given ideal" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Ca
lling Sequences" }}{PARA 0 "" 0 "" {TEXT -1 27 "quickbasis_gb([f1,...,
fs]);" }}{PARA 0 "" 0 "" {TEXT -1 36 "quickbasis_gb([f1,...,fs], noste
ps];" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "
" 0 "" {TEXT -1 79 "[f1,...,fs] = a list of polynomials in the ring d
efined by ring()\n   " }}{PARA 0 "" 0 "" {TEXT -1 67 "nosteps
 = the string that indicates that no steps are to be printed" }}}
{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT
-1 236 "quickbasis_gb([f1,...,fs]) returns a Groebner basis of <f1,...
,fs> using a streamlined version of Buchberger's algorithm. This basi
s is generally neither minimal nor reduced. Steps of constructing the
 Groebner basis are also printed.\n" }}{PARA 0 "" 0 "" {TEXT -1 122 "q
uickbasis_gb([f1,...,fs], nosteps) returns the same things, but steps \+
of constructing the Groebner basis are not printed." }}}{SECT 0 {PARA
4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1
0 19 "ring(grlex, [x,y]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_or

derG" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 38 "quickbasis_gb([x^2*
y - 1, x*y^2 - x]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%0Current~Basi
s:~G7$,&*&)%\"xG\"\"#\"\"\"%\"yG\"\"\"F-!\"\"F-,&*&F)F-)F,F*F+F-F)F."
}}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G,&%\"yG!\"\"*$)%\"xG\"\"#
\"\"\"\"\"\"" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%2Local~divisions:~G,
&%*localdivsG\"\"\"F'F'" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#(%(Added:~G
,&*$)%\"yG\"\"#\"\"\"\"\"\"!\"\"F+" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#
%3Local~divisions:~1G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%=Total~divis
ions~performed:~4G" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7&,&*&)%\"xG\"\"
#\"\"\"%\"yG\"\"\"F+!\"\"F+,&*&F'F+)F*F(F)F+F'F,,&F*F,*$F&F)F+,&*$F/F)
F+F,F+" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 47 "quickbasis_gb([x^
2*y - 1, x*y^2 - x], nosteps);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7&,&
&)%\"xG\"\"#\"\"\"%\"yG\"\"\"F+!\"\"F+,&&F'F+)F*F(F)F+F'F,,&F*F,*$F&
F)F+,&*$F/F)F+F,F+" }}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also"
}}{PARA 0 "" 0 "" {TEXT -1 46 "ring(), altbasis_gb(), quickbasis_gb(),
 mxgb()" }}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 8 "min_gb()" }}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 29 "To
 minimize a Groebner basis." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "C
alling Sequence" }}{PARA 0 "" 0 "" {TEXT -1 20 "min_gb([g1,...,gs]);"
}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 ""
{TEXT -1 97 "[g1,...,gs] = a list of polynomials that form a Groebner \+
basis under the term ordering of ring()." }}}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 111 "min_gb([g1,...,g
s]) returns a list of polynomials that form a minimal Groebner basis f
or the ideal <g1,...,gs>." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Exam
ples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 21 "ring(lex, [x,y,z,w])
;" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0
"> " 0 "" {MPLTEXT 1 0 82 "gb := quickbasis_gb([3*x - 6*y - 2*z, 2*x -
 4*y + 4*w, x - 2*y - z - w], nosteps);" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#>%#gbG7&,(%\"xG\"\"$%\"yG!\"'%\"zG!\"#,(F'\"\"#F)!\"%%\"wG\"\"%,
*F'\"\"\"F)F,F+!\"\"F0F4,&F+F/F0!#7" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 11 "min_gb(gb);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7$,*%
\"xG\"\"\"%\"yG!\"#%\"zG!\"\"%\"wGF*,&F)!\"%F+!#7" }}}}}{SECT 1 {PARA
3 "" 0 "" {TEXT -1 8 "red_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "
Purpose" }}{PARA 0 "" 0 "" {TEXT -1 27 "To reduce a Groebner basis." }
}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequence" }}{PARA 0 ""
0 "" {TEXT -1 20 "red_gb([g1,...,gs]);" }}}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 105 "[g1,...,gs] =
 a list of polynomials that form a minimal Groebner basis under the te
rm ordering of ring()." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsi
s" }}{PARA 0 "" 0 "" {TEXT -1 111 "red_gb([g1,...,gs]) returns a list \+
of polynomials that form a minimal Groebner basis for the ideal <g1,..
.,gs>." }}}{PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> \+
" 0 "" {MPLTEXT 1 0 21 "ring(lex, [x,y,z,w]);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1
0 90 "gb := min_gb(quickbasis_gb([3*x - 6*y - 2*z, 2*x - 4*y + 4*w, x \+
- 2*y - z - w], nosteps));" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#>%#gbG7$
,*%\"xG\"\"\"%\"yG!\"#%\"zG!\"\"%\"wGF,,&F+!\"%F-!#7" }}}{EXCHG {PARA

0 "> " 0 "" {MPLTEXT 1 0 11 "red_gb(gb);" }}{PARA 11 "" 1 "" {XPPMATH
20 "6#7$,(%\"xG\"\"\"%\"yG!\"#%\"wG\"\"#,&%\"zGF&F)\"\"$" }}}}{SECT 1
{PARA 3 "" 0 "" {TEXT -1 9 "div_alg()" }}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 72 "div_alg() performs
 the division algorithm for multivariable polynomials." }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequence" }}{PARA 0 "" 0 ""
{TEXT -1 24 "div_alg(f, [f1,...,fs]);" }}}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 32 "f = the polyno
mial to be divided" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 ""
{TEXT -1 43 "[f1,...,fs] = a list of polynomial divisors" }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 242 "
div_alg(f,[f1,...,fs]) returns a list. The list's first element is th
e remainder of the division with respect to the order and ring set by \+
ring(). The list's second element is the list of quotients, with resp
ect to the order of [f1,...,fs]." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1
8 "Examples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 28 "ring([[1,1],[
0,-1]], [x,y]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 55 "div_alg(5*x^2 - y, [x^2 + y,
 x^4 + 2*x^2*y + y^2 + 3]);" }}{PARA 0 "> " 0 "" {MPLTEXT 1 0 0 "" }}
{PARA 11 "" 1 "" {XPPMATH 20 "6#7$,$%\"yG!\"7$\"\"&\"\"!" }}}}{SECT
0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}{PARA 0 "" 0 "" {TEXT -1 6 "
ring()" }}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 9 "quot_mx()" }}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 47 "qu
ot_mx is a matrix of quotients (see Synopsis)" }}}{SECT 0 {PARA 4 ""
0 "" {TEXT -1 16 "Calling Sequence" }}{PARA 0 "" 0 "" {TEXT -1 34 "quo
t_mx([f1,...,fs], [g1,...,gt]);" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1
10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 79 "[f1,...,fs] = a list of
 polynomials, where each fi is in the ideal <g1,...,gt>\n" }}{PARA 0 "
" 0 "" {TEXT -1 120 "[g1,...,gt] = a list of polynomial that forms a G
roebner basis with respect to the monomial order and ring set by ring(
)" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 ""
{TEXT -1 161 "quot_mx([f1,...,fs], [g1,...,gt]) returns a matrix of qu
otients. In other words, we have [g1,...,gt]*Q^T = [f1,...,fs], where
 Q^T represents the transpose of Q." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 8 "Examples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 21 "ring(grevl
ex, [x,y]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#%+term_orderG" }}}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 53 "quot_mx([x^2*y - 1, x*y^2 - \+
x], [-y + x^2, y^2 - 1]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#-%'matrix
G6#7$7$%\"yG\"\"\"7$\"\"!%\"xG" }}}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1
6 "mxgb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "
" 0 "" {TEXT -1 85 "mxgb() computes a reduced Groebner basis and its c
orresponding transformation matrix." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 17 "Calling Sequences" }}{PARA 0 "" 0 "" {TEXT -1 19 "mxgb([f1,...,
fs]); " }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 26
"mxgb([f1,...,fs], nosteps]" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "P
arameters" }}{PARA 0 "" 0 "" {TEXT -1 67 "[f1,...,fs] = a list of pol
ynomials in the ring defined by ring()\n" }}{PARA 0 "" 0 "" {TEXT -1
67 "nosteps = the string that indicates that no steps are to be printe

d" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 ""
{TEXT -1 545 "mxgb([f1,...,fs]) returns a list. The first element, G=
[g1,...,gt], is a Groebner basis of <f1,...,fs>, using the streamlined
 version of Buchberger's algorithm of quickbasi_gb(), except that the \+
basis is reduced. The second element a list representing the coeffici
ent matrix, showing how each polynomial in the Groebner basis is repre
sented by the polynomials <f1,...,fs>. In other words, [f1,...,fs]*Q^
T=[g1,...,gt], where Q^T represents the transpose of Q. Steps of mini
mizing a reducing the Groebner basis and its matrix are also printed.
" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 78 "mxgb(
[f1,...,fs], nosteps) returns the same things, but steps are not print
ed." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA
0 "> " 0 "" {MPLTEXT 1 0 19 "ring([1,2], [x,y]);" }}{PARA 11 "" 1 ""
{XPPMATH 20 "6#%+term_orderG" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1
0 29 "mxgb([x^2*y - 1, x*y^2 - x]);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6
$%4Unminimized~basis:~G7&,&*&)%\"xG\"\"#\"\"\"%\"yG\"\"\"F,!\"\"F,,&*&
F(F,)F+F)F*F,F(F-,&F+F-*$F'F*F,,&*$F0F*F,F-F," }}{PARA 11 "" 1 ""
{XPPMATH 20 "6$%5Unminimized~matrix:~G-%'matrixG6#7&7$\"\"\"\"\"!7$F*F
)7$%\"yG,$%\"xG!\"\"7$,&F)F)*$)F-\"\"#\"\"\"F0*&F-F)F/F)" }}{PARA 11 "
" 1 "" {XPPMATH 20 "6%%2Minimized~basis:~G,&%\"yG!\"\"*$)%\"xG\"\"#\"
\"\"\"\"\",&*$)F%F*F+F,F&F," }}{PARA 11 "" 1 "" {XPPMATH 20 "6$%3Minim
ized~matrix:~G-%'matrixG6#7$7$%\"yG,$%\"xG!\"\"7$,&\"\"\"F/*$)F)\"\"#
\"\"\"F,*&F)F/F+F/" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7$7$,&%\"yG!\"\"
*$)%\"xG\"\"#\"\"\"\"\"\",&*$)F&F+F,F-F'F--%'matrixG6#7$7$F&,$F*F'7$,&
F-F-F/F'*&F&F-F*F-" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 38 "mxgb(
[x^2*y - 1, x*y^2 - x], nosteps);" }}{PARA 11 "" 1 "" {XPPMATH 20 "6#7
7,&%\"yG!\"\"*$)%\"xG\"\"#\"\"\"\"\"\",&*$)F&F+F,F-F'F--%'matrixG6#7
7F&,$F*F'7$,&F-F-F/F'*&F&F-F*F-" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 0 "" }}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}
{PARA 0 "" 0 "" {TEXT -1 48 "ring(), slowbasis_gb, altbasis_gb, quickb
asis_gb" }}}}}{MARK "10" 0 }{VIEWOPTS 1 1 0 1 1 1803 }

{VERSION 4 0 "IBM INTEL LINUX22" "4.0" }
{USTYLETAB {CSTYLE "Maple Input" -1 0 "Courier" 0 1 255 0 0 1 0 1 0 0
1 0 0 0 0 1 }{CSTYLE "2D Math" -1 2 "Times" 0 1 0 0 0 0 0 0 2 0 0 0 0
0 0 1 }{CSTYLE "2D Comment" 2 18 "" 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 256 "" 1 18 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
257 "" 1 24 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 258 "" 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 259 "" 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 260 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 261 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
262 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 263 "" 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 264 "" 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 265 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 266 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
267 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 268 "" 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 269 "" 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 270 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 271 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
272 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 273 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 274 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 275 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 276 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
277 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 278 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 279 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 280 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 281 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
282 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 283 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 284 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 285 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 286 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
287 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 288 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 289 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 290 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 291 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
292 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 293 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 294 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 295 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 296 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
297 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 298 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 299 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 300 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 301 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
302 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 303 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 304 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 305 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 306 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
307 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{PSTYLE "Normal" -1 0 1
{CSTYLE "" -1 -1 "" 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }0 0 0 -1 -1 -1 0

0 0 0 0 0 -1 0 }{PSTYLE "Heading 1" 0 3 1 {CSTYLE "" -1 -1 "" 1 18 0
0 0 0 0 1 0 0 0 0 0 0 0 1 }1 0 0 0 8 4 0 0 0 0 0 0 -1 0 }{PSTYLE "Head
ing 2" 3 4 1 {CSTYLE "" -1 -1 "" 1 14 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }0
0 0 -1 8 2 0 0 0 0 0 0 -1 0 }}
{SECT 0 {EXCHG {PARA 0 "" 0 "" {TEXT -1 0 "" }{TEXT 256 0 "" }{TEXT
257 29 "Tutorial for the gbr5 package" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 32 "By William Gryc, Amherst College" }}{PARA 0 "" 0 ""
{TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 843 "   Well, you finally di
d it. After hours of deliberation and procrastination, you have final
ly downloaded and saved the gbr5 files on your computer, just like you
r professor asked you to. Now you are expecting a long-winded and com
plex tutorial. Well, you aren't getting it here! This tutorial is de
signed to be clear and quick, so you can start working as soon as poss
ible. If you want longer explanations, try looking at the help access
ed by opening the worksheet \"gbr5hlp.mws\" which, hopefully, you also
 downloaded. You'll be sure to find whatever you want to know there. \+
 Also, please be advised that some of the commands in the package are \+
slow compared to regular Maple commands, but they should be adequate i
n computing the simpler examples in \"Ideals, Varieties, and Algorithm
s.\" But, now in the spirit of being quick, onto...." }}}{SECT 0
{PARA 3 "" 0 "" {TEXT -1 26 "The Basics: Loading `gbr5`" }}{PARA 0 ""
0 "" {TEXT -1 344 "Even though you managed to get this tutorial to run
ning, that's not all there is to it. There's still the issue of loadi
ng the package. This is not too terribly difficult, however. Be sure
 you've also downloaded \"gbr5.mpl\", \"gbr5hlp.mws\". Then, assuming
 that you started Maple from the directory containing these files, the
n all you type is" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{EXCHG {PARA 0 ">
" 0 "" {MPLTEXT 1 0 17 "read(`gbr5.mpl`):" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 247 "Please note that to use the interactive tutorial, you ju
st have to press Enter (or Return, depending on your keyboard) on each
 Maple input line. After hitting Enter on the previous Maple input li
ne, the package is loaded and we are ready to work." }}}{EXCHG {PARA
0 "" 0 "" {TEXT -1 124 "Also note that reading in \"gbr5.mpl\" generat
es two warning messages concerning norm and trace. These can be safel
y ignored." }}}}{SECT 0 {PARA 3 "" 0 "" {TEXT -1 27 "ring() and Relate
d Commands" }}{PARA 0 "" 0 "" {TEXT -1 567 "The most basic command in \+
the package is ring(). This command sets the ring and the monomial or
der (see Chapter 2, Section 2 of the text) which most other commands i
n the will use. In fact, if you try to use any commands without first
 setting ring(), you'll get a nasty error message saying that you must
 set ring() first. The arguments for ring() are a monomial order and \+
a variable list. The valid monomial orders are lex, grlex, grevlex (s
ee Chapter 2 Section 2), [k,n] (the kth elimination order on n variabl
es; see Exercise 12d in Chapter 2 Section 2), and [" }{TEXT 268 2 "v1
" }{TEXT -1 5 ",...," }{TEXT 269 2 "vn" }{TEXT -1 168 "]. The last or
der is a matrix order and is not discussed in the text explicitly. Sa
y you were working in the ring k[x1,...,xn]. Then a matrix order woul
d consist of " }{TEXT 260 1 "n" }{TEXT -1 31 " linearly independent ve

ctors \{" }{TEXT 258 2 "v1" }{TEXT -1 5 ",...," }{TEXT 259 2 "vn" }
{TEXT -1 17 "\} each of length " }{TEXT 261 2 "n," }{TEXT -1 13 " such
 that x^" }{XPPEDIT 18 0 "alpha;" "6#%&alphaG" }{TEXT -1 5 " > x^" }
{XPPEDIT 18 0 "beta;" "6#%%betaG" }{TEXT -1 5 " iff " }{TEXT 262 2 "v1
" }{TEXT -1 1 "*" }{XPPEDIT 18 0 "alpha;" "6#%&alphaG" }{TEXT -1 3 " >
 " }{TEXT 263 3 "v1*" }{XPPEDIT 18 0 "beta;" "6#%%betaG" }{TEXT -1 4 "
 or " }{TEXT 264 2 "v1" }{TEXT -1 1 "*" }{XPPEDIT 18 0 "alpha;" "6#%&a
lphaG" }{TEXT -1 3 " = " }{TEXT 265 3 "v1*" }{XPPEDIT 18 0 "beta;" "6#
%%betaG" }{TEXT -1 5 " and " }{TEXT 266 2 "v2" }{TEXT -1 1 "*" }
{XPPEDIT 18 0 "alpha;" "6#%&alphaG" }{TEXT -1 3 " > " }{TEXT 267 2 "v2
" }{TEXT -1 1 "*" }{XPPEDIT 18 0 "beta;" "6#%%betaG" }{TEXT -1 351 " ,
 etc. All monomials orders the package deals with can be written as m
atrix orders. As you will see in a moment, this is how the package se
ts monomial orders (except for lex and grevlex, which use the built in
 Maple orders plex and tdeg, respectively). So, say you wanted to set
 the ring k[x,y,z] and use the monomial order grevlex. So, you type \+
" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 23 "ring(grevlex, [x,y,z]);
" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 119 "Then you change your mind an
d decide on a matrix order with the simplest linearly independant vect
ors you can think of." }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 41 "ri
ng([[1,0,0],[0,1,0],[0,0,1]], [x,y,z]);" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 183 "Notice that grevlex was not capitalized. This is import
ant, as lex, grlex, and grevlex must be entered exactly as shown here,
 in all lower case letters, for ring() to work correctly." }}{PARA 0 "
" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 531 "The ring() comman
ds creates term orders in two ways. For lex and grevlex, it uses the \+
terms orders plex and tdeg from the Groebner package. However, for gr
lex and elimination orders, ring() uses matrices created by the grlex(
) and elimination() commands. These are the \"related commands\" that
 the section title refers to. The lone argument for grlex() is the in
teger that is the number of variables in the ring. So, if we were wor
king in k[x1,x2,x3,x4] and wanted the matrix that yields grevlex order
 over this ring, we'd type" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0
9 "grlex(4);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 206 "Notice that grle
x(), as well as elimination(), does not depend on the current ring and
 monomial order set by ring().   elimination() works differently; it n
eeds one extra argument. The form is elimination(" }{TEXT 270 3 "k,n
" }{TEXT -1 9 "), where " }{TEXT 271 1 "n" }{TEXT -1 44 " is the numbe
r of variables in the ring and " }{TEXT 272 1 "k" }{TEXT -1 175 " is t
he number of variables to eliminate. So, to get the matrix that gives
 a monomial order that eliminates the first three variables of a ring \+
with five variables, you enter" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT
1 0 17 "elimination(3,5);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 25 "And \+
you have your matrix." }}}}{SECT 0 {PARA 3 "" 0 "" {TEXT -1 22 "The Di
vision Algorithm" }}{PARA 0 "" 0 "" {TEXT -1 482 "One of the major ste
ps in computing Groebner bases is computing remainders via the Divisio
n algorithm. The command div_alg() implements this algorithm. The tw
o arguments of div_alg are a polynomial and a set of polynomials. div

_alg returns a list with two elements: the first element is the remai
nder, and the second is a list of quotients, ordered in correspondence
 with the ordering of the given set of polynomials. Let's use Problem
 1a in Chapter 2 Section 3 for an example." }}{EXCHG {PARA 0 "> " 0 "
" {MPLTEXT 1 0 19 "ring(grlex, [x,y]);" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 57 "div_alg(x^7*y^2 + x^3*y^2 - y + 1, [x*y^2 - x, x - y^
3]);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 70 "Notice div_alg() divides \+
with respect to the term order set by ring()." }}}}{SECT 0 {PARA 3 ""
0 "" {TEXT -1 41 "Computing Groebner Bases without Matrices" }}{PARA
0 "" 0 "" {TEXT -1 465 "This package provides four different ways to c
ompute Groebner bases. The first, slowbasis_gb implements a naive ver
sion of Buchberger's algorithm (see Chapter 2 Section 2). The paramet
er of this command is a list of polynomials [f1,...,fs] slowbasis_gb \+
then returns a list of polynomials that form a Groebner basis for <f1,
...,fs> with respect to the termorder set by ring. For an example, we
'll use Exercise 2b from Chapter 2 Section 7. First we set the ring.
" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 17 "ring(lex, [x,y]);" }}}
{EXCHG {PARA 0 "" 0 "" {TEXT -1 32 "Then we find the Groebner basis."
}}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 49 "slowbasis_gb([x^2 + y, x^
4 + 2*x^2*y + y^2 + 3]);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 165 "Noti
ce that the steps of calculation were also printed out, as well as how
 many divisions were performed. To prevent this, just type \"nosteps
\" as a second argument." }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 58
"slowbasis_gb([x^2 + y, x^4 + 2*x^2*y + y^2 + 3], nosteps);" }}}
{EXCHG {PARA 0 "" 0 "" {TEXT -1 67 "Sometimes slowbasis_gb gets out of
 hand. Consider this example:. " }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 23 "ring(grevlex, [x,y,z]);" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 50 "basisprime := [x^3 - z^2, y^3 + z, x^2*y + x*y^2];" }
}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 25 "slowbasis_gb(basisprime);
" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 338 "The returned basis is quite \+
unnecessarily repetitive. The reason for this repetition is that the \+
algorithm is using G' as opposed to G is division (see Buchberger's al
gorithm in Chapter 2 Section 7). A second command, altbasis_gb uses G
 instead. Notice that the repetion is now gone, and the number of div
isions decreases dramatically." }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT
1 0 24 "altbasis_gb(basisprime);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1
279 "While altbasis_gb() can dramatically improve performance, we can \+
still do better. The quickbasis_gb() command implements the improveme
nts to Buchberger's algorithm as detailed in Chaptert 2 Section 9. Le
t's use two examples to show how quickbasis_gb can outperform altbasis
_gb:" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 21 "basis1 := [x^2, y^4
];" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 20 "altbasis_gb(basis1);
" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 22 "quickbasis_gb(basis1);
" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 283 "Clearly, basis1 is already a
 Groebner basis by definition, and yet altbasis_gb() still performs a \+
division. quickbasis_gb() cuts out this division by noticing that x^2
 and y^4 are relatively prime, and thus no divisons need to be perform
ed (see Propostion 4 of Chapter 2 Section 9)." }}}{EXCHG {PARA 0 "> "

0 "" {MPLTEXT 1 0 32 "basis2 := [x^2, x*y^3, x^2*y^3];" }}}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 20 "altbasis_gb(basis2);" }}}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 22 "quickbasis_gb(basis2);" }}}{EXCHG
{PARA 0 "" 0 "" {TEXT -1 319 "Again, quickbasis_gb() is able to cut ou
t a division, even though none of the leading terms of basis2 are rela
tively prime. However, notice that the third leading term, x^2y^3, di
vides the LCM of x^2 and xy^3 (in fact, x^2y^3 is the LCM of x^2 and x
y^3). By Propostion 10 of Chapter 2 Section 9, if the remainders of \+
" }{TEXT 273 1 "S" }{TEXT -1 16 "(x^2, xy^3) and " }{TEXT 274 1 "S" }
{TEXT -1 52 "(x^2, x^2y^3) are both calculated, the remainder of " }
{TEXT 275 1 "S" }{TEXT -1 100 "(xy^3, x^2y^3) need not be calculated. \+
 Thus, quickbasis_gb is implementing this second improvement." }}
{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 198 "It shoul
d be noted, however, that Maple's built-in gbasis() command is faster \+
than quickbasis_gb(). However, quickbasis_gb() should be fast enough \+
for most of the examples and problems in the text." }}}{SECT 0 {PARA
4 "" 0 "" {TEXT -1 38 "Minimizing and Reducing Groebner bases" }}
{PARA 0 "" 0 "" {TEXT -1 160 "For added convenience, there are also co
mmands to minimize and reduce your computed Groebner bases. For an ex
ample, let's use Problem 9 of Chapter 2 Section 7." }}{EXCHG {PARA 0 "
> " 0 "" {MPLTEXT 1 0 21 "ring(lex, [x,y,z,w]);" }}}{EXCHG {PARA 0 "> \+
" 0 "" {MPLTEXT 1 0 61 "basis := [3*x - 6*y - 2*z, 2*x - 4*y + 4*w, x \+
- 2*y - z - w];" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 44 "basispri
me := quickbasis_gb(basis, nosteps);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT
-1 208 "To get a minimal Groebner basis from this, we can use the min_
gb() command. min_gb() takes a list of polynomials that form a Groebn
er basis under ring() as its lone argument. So, to make basisprime mi
nimal," }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 32 "basisprime :=min_
gb(basisprime);" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 336 "Getting a red
uced Groebner basis from this new basisprime, we use the red_gb comman
d. red_gb() takes a list of polynomials that for a MINIMAL Groebner b
asis under ring() as its argument. If the basis isn't a minimal Groeb
ner basis, red_gb() will protest and will refuse to do its job. So, t
o make basisprime a reduced Groebner basis," }}}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 19 "red_gb(basisprime);" }}}{EXCHG {PARA 0 "" 0 ""
{TEXT -1 34 "And the Groebner basis is reduced." }}}}}{SECT 0 {PARA 3
"" 0 "" {TEXT -1 38 "Computing Groebner Bases with Matrices" }}{PARA
0 "" 0 "" {TEXT -1 60 "Consider the following type of problem: You ha
ve an ideal <" }{TEXT 284 9 "f1,...,fs" }{TEXT -1 40 "> and compute a \+
Groebner basis for it, \{" }{TEXT 285 9 "g1,...,gt" }{TEXT -1 12 "\}. \+
 Since <" }{TEXT 286 9 "f1,...,fs" }{TEXT -1 5 "> = <" }{TEXT 287 9 "
g1,...,gt" }{TEXT -1 12 ">, for each " }{TEXT 288 2 "fi" }{TEXT -1 15
", there exists " }{TEXT 289 9 "a1,...,at" }{TEXT -1 11 " such that "
}{TEXT 290 2 "fi" }{TEXT -1 3 " = " }{TEXT 291 4 "a1g1" }{TEXT -1 3 " \+
+ " }{TEXT 292 4 "a2g2" }{TEXT -1 9 " + ... + " }{TEXT 293 4 "atgt" }
{TEXT -1 65 ", and vice versa. For the first problem, it is easy to f
ind the " }{TEXT 294 9 "a1,...,at" }{TEXT -1 8 " since \{" }{TEXT 296
8 "g1,..,gt" }{TEXT -1 55 "\} is a Groebner basis the division algorit

hm gives the " }{TEXT 295 2 "ai" }{TEXT -1 301 ". There is a function
 in the package, quot_mx(), that does exactly this. quot_mx() takes t
wo arguments. The first is a list of polynomials that generate an ide
al, the second is a Groebner basis under the termorder set in ring() t
hat generates the same ideal. The output is a matrix Q where Q is a \+
" }{TEXT 303 1 "s" }{TEXT -1 3 " x " }{TEXT 304 1 "t" }{TEXT -1 12 " m
atrix and " }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT
-1 1 "[" }{TEXT 297 2 "f1" }{TEXT -1 15 "]   [" }{TEXT 300
2 "g1" }{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 298 2 "
f2" }{TEXT -1 15 "]   [" }{TEXT 301 2 "g2" }{TEXT -1 1 "]" }
}{PARA 0 "" 0 "" {TEXT -1 20 "[...]   = Q * [...]" }}{PARA 0 "" 0 ""
{TEXT -1 24 "[...]   [...]" }}{PARA 0 "" 0 "" {TEXT -1 1 "[
" }{TEXT 299 2 "fs" }{TEXT -1 16 "]   [" }{TEXT 302 2 "gt"
}{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 ""
{TEXT -1 31 "Consider the following example:" }}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 23 "ring(grevlex, [x,y,z]);" }}}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 52 "gbase := quickbasis_gb([x - z^4, y - z^5], nosteps
);" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 40 "Q := quot_mx([x - z^4
, y - z^5], gbase);" }}}{PARA 0 "" 0 "" {TEXT -1 133 "Notice that Q is
 not unique, as the most obvious choice for Q is not the matrix given.
  Let's verify that Q has the desired property." }}{EXCHG {PARA 0 "> \+
" 0 "" {MPLTEXT 1 0 26 "base := matrix(6,1,gbase);" }}}{PARA 0 "" 0 "
" {TEXT -1 0 "" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 27 "simplify(m
ultiply(Q,base));" }}}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 ""
{TEXT -1 43 "The opposite question of how to represent \{" }{TEXT 305
9 "g1,...,gt" }{TEXT -1 6 "\} in <" }{TEXT 306 9 "f1,...,fs" }{TEXT
-1 133 "> is trickier, since the division algorithm can't do the job i
n this case. However, you can compute the matrix while computing the \+
\{" }{TEXT 307 8 "g1,..,gt" }{TEXT -1 321 "\}. Computing Groebner base
s with matrices isn't a whole lot different than computing them withou
t matrices (that is, from the user's standpoint). You still use the r
ing() command to set the term-ordering. However, you just use one com
mand, mxgb(), to compute a reduced Groebner basis and its matrix M. T
he matrix M an " }{TEXT 277 1 "t" }{TEXT -1 3 " x " }{TEXT 276 1 "s" }
{TEXT -1 17 " matix such that:" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}
{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 278 2 "g1" }{TEXT -1 15 "]   \+
      [" }{TEXT 281 2 "f1" }{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT
-1 1 "[" }{TEXT 279 2 "g2" }{TEXT -1 15 "]   [" }{TEXT 282
2 "f2" }{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT -1 20 "[...]   = M * [
...]" }}{PARA 0 "" 0 "" {TEXT -1 24 "[...]   [...]" }}
{PARA 0 "" 0 "" {TEXT -1 1 "[" }{TEXT 280 2 "gt" }{TEXT -1 16 "]   \+
       [" }{TEXT 283 2 "fs" }{TEXT -1 1 "]" }}{PARA 0 "" 0 "" {TEXT
-1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 278 "mxgb() takes a list of polynom
ials as its argument. If you do not desire to see the basis and matri
x at each \"step\" (at the unminimized step, and the unreduced steps),
 a second argument of \"nosteps\" should be added. Let's use Problem \+
2c of Chapter 2 Section 7 as an example. " }}{EXCHG {PARA 0 "> " 0 "
" {MPLTEXT 1 0 25 "ring(grevlex, [x, y, z]);" }}}{EXCHG {PARA 0 "> "

0 "" {MPLTEXT 1 0 35 "mxbase := mxgb([x - z^4, y - z^5]);" }}}{EXCHG
{PARA 0 "" 0 "" {TEXT -1 46 "Let's see if this matrix is what we say i
t is." }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 37 "base := matrix(2,1
,[x-z^4, y - z^5]);" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 36 "simp
lify(multiply(mxbase[2], base));" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 0 "" }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1 112 "Since this m
atrix gives the element of the Groebner basis, the matrix given by mxg
b() is what we claimed it was." }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}
{PARA 0 "" 0 "" {TEXT -1 349 "As you may have noticed, there are quick
basis_mxgb, min_mxgb, and red_mxgb commands which have counterparts th
at do not have the \"mx\" prefix. All these commands are combined for
 the mxgb command, and are not meant for users. This is not to say th
at you cannot use them, but these commands aren't terribly user friend
ly and we discourage their use." }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}
{PARA 0 "" 0 "" {TEXT -1 143 "That's it for this tutorial. Remember, \+
more information on these commands can be found below in the reference
 guide. Thank you and good luck!" }}}}{SECT 0 {PARA 3 "" 0 "" {TEXT
-1 16 "Acknowledgements" }}{PARA 0 "" 0 "" {TEXT -1 117 "Will Gryc and
 David Cox would like to thank the Charleton Trust for supporting Will
's work on this Maple worksheet. " }}}}{MARK "2 4 0 0" 0 }{VIEWOPTS
1 1 0 3 2 1804 1 1 1 1 }{PAGENUMBERS 0 1 2 33 1 1 }

{VERSION 4 0 "IBM INTEL LINUX22" "4.0" }
{USTYLETAB {CSTYLE "Maple Input" -1 0 "Courier" 0 1 255 0 0 1 0 1 0 0
1 0 0 0 0 1 }{CSTYLE "" -1 256 "" 1 24 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 257 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
258 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 259 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 260 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 261 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 262 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
263 "" 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 264 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 265 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 266 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 267 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
268 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 269 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 270 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{CSTYLE "" -1 271 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }
{CSTYLE "" -1 272 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1
273 "" 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 274 "" 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 1 }{CSTYLE "" -1 275 "" 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 1 }{PSTYLE "Normal" -1 0 1 {CSTYLE "" -1 -1 "" 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 }0 0 0 -1 -1 -1 0 0 0 0 0 0 -1 0 }{PSTYLE "Heading 1"
0 3 1 {CSTYLE "" -1 -1 "" 1 18 0 0 0 0 0 1 0 0 0 0 0 0 0 1 }1 0 0 0 8
4 0 0 0 0 0 0 -1 0 }{PSTYLE "Heading 2" 3 4 1 {CSTYLE "" -1 -1 "" 1
14 0 0 0 0 0 0 0 0 0 0 0 0 0 1 }0 0 0 -1 8 2 0 0 0 0 0 0 -1 0 }}
{SECT 0 {EXCHG {PARA 0 "" 0 "" {TEXT -1 0 "" }{TEXT 256 41 "Reference \+
worksheet for the gbr5 package " }}}{EXCHG {PARA 0 "" 0 "" {TEXT -1
404 "The gbr5 package provides commands to compute Grobner bases which
 also can show the steps involved in computing them. The major comman
ds in this package are listed below in order of need to know (i.e., th
e most basic command is first, followed by the next most basic command
, etc). Maximize the command you would like to read about. (To maxim
ize a command, click on the plus sign next to the command.)" }}}{SECT
1 {PARA 3 "" 0 "" {TEXT -1 37 "General Information about the Package"
}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}{PARA 0 "
" 0 "" {TEXT -1 16 "<function>(args)" }}}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 38 "The functions in \+
the gbr5 package are:" }}{PARA 0 "" 0 "" {TEXT -1 65 " ring   l
ex\011\011   grlex\011\011   grevlex   elimination" }}
{PARA 0 "" 0 "" {TEXT -1 110 " slowbasis_gb\011   altbasis_gb\011
          quickbasis_gb\011\n\011div_alg\011\011   quot_m
x\011\011   mxgb" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "
" 0 "" {TEXT -1 45 "This package uses the global variable morder." }}
{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 144 "Before a
ny of slowbasis_gb, altbasis_gb, quickbasis_gb, mxgb, quot_mx, or div
_alg can be used, ring must be performed (see ring() for details)." }}
}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 6 "ring()" }}{SECT 0 {PARA 4 "" 0 "
" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 69 "ring() sets the \+
termorder and variables for the package to work under" }}}{SECT 0

{PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}{PARA 0 "" 0 ""
{TEXT -1 21 "ring (torder,varlist)" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 68 "torder = the monomia
l order. Valid values are lex, grlex, grevlex, " }}{PARA 0 "" 0 ""
{TEXT -1 1 "[" }{TEXT 257 1 "k" }{TEXT -1 1 "," }{TEXT 258 1 "n" }
{TEXT -1 51 "] (the elimination order that eliminates the first " }
{TEXT 259 1 "k" }{TEXT -1 4 " of " }{TEXT 260 1 "n" }{TEXT -1 18 " var
iables), and [" }{TEXT 261 2 "v1" }{TEXT -1 4 ",..," }{TEXT 262 2 "vn
" }{TEXT -1 25 "] (a matrix order, where " }{TEXT 263 2 "vi" }{TEXT
-1 10 " is a 1 x " }{TEXT 264 1 "n" }{TEXT -1 13 " row vector)." }}
{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 121 "varlist \+
= a list of the variables of the ring. Note that if an elimination or
der or matrix order is used, there must be " }{TEXT 265 1 "n" }{TEXT
-1 22 " variables in varlist." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "
Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 84 "ring(torder, varlist) returns
 the term_order with respect to the torder and varlist." }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 21 "ring(grlex, [x,y,z]);" }}}}}{SECT 1 {PARA 3 "" 0 ""
{TEXT -1 7 "grlex()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}
{PARA 0 "" 0 "" {TEXT -1 57 "Creates a matrix whose row vectors produc
e a grlex order." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequ
ence" }}{PARA 0 "" 0 "" {TEXT -1 8 "grlex(n)" }}}{SECT 0 {PARA 4 "" 0
"" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 32 " n = number
 of variables in ring" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis
" }}{PARA 0 "" 0 "" {TEXT -1 6 "grlex(" }{TEXT 269 1 "n" }{TEXT -1 36
") returns a list which represents a " }{TEXT 266 1 "n" }{TEXT -1 3 " \+
x " }{TEXT 267 1 "n" }{TEXT -1 74 " matrix whose rows produce a matrix
 order which is equivalent to grlex on " }{TEXT 268 1 "n" }{TEXT -1
11 " variables." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 9 "grlex(6);" }}}}{PARA 4 "" 0 "
" {TEXT -1 0 "" }}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 13 "elimination()
" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 ""
{TEXT -1 64 "Creates a matrix whose row vectors produce an elimination
 order." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequence" }}
{PARA 0 "" 0 "" {TEXT -1 17 "elimination(k,n);" }}}{SECT 0 {PARA 4 ""
0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 40 "k = the n
umber of variables to eliminate" }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}
{PARA 0 "" 0 "" {TEXT -1 39 "n = the number of variables in the ring"
}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 ""
{TEXT -1 12 "elimination(" }{TEXT 274 1 "k" }{TEXT -1 1 "," }{TEXT
270 1 "n" }{TEXT -1 36 ") returns a list which represents a " }{TEXT
271 1 "n" }{TEXT -1 3 " x " }{TEXT 272 1 "n" }{TEXT -1 113 " matrix wh
ose rows produce a matrix order which is equivalent to the elimination
 order that eliminates the first " }{TEXT 273 1 "k" }{TEXT -1 4 " of \+
" }{TEXT 275 1 "n" }{TEXT -1 11 " variables." }}}{SECT 0 {PARA 4 "" 0
"" {TEXT -1 7 "Example" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 17 "el
imination(3,5);" }}}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 14 "slowbasis_g
b()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 ""

{TEXT -1 58 "slowbasis_gb() finds a Groebner basis for the given ideal
." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}{PARA
0 "" 0 "" {TEXT -1 26 "slowbasis_gb([f1,...,fs]);" }}{PARA 0 "" 0 ""
{TEXT -1 35 "slowbasis_gb([f1,...,fs], nosteps];" }}}{SECT 0 {PARA 4 "
" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 79 "[f1,...
,fs] = a list of polynomials in the ring defined by ring()\n   \+
   " }}{PARA 0 "" 0 "" {TEXT -1 67 "nosteps = the string that indicate
s that no steps are to be printed" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 229 "slowbasis_gb([f1,...,f
s]) returns a Groebner basis of <f1,...,fs> using a naive version of B
uchberger's algorithm. This basis is generally neither minimal nor re
duced. Steps of constructing the Groebner basis are also printed.\n"
}}{PARA 0 "" 0 "" {TEXT -1 121 "slowbasis_gb([f1,...,fs], nosteps) ret
urns the same things, but steps of constructing the Groebner basis are
 not printed." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "ring(grlex, [x,y]);" }}}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 37 "slowbasis_gb([x^2*y - 1, x*y
^2 - x]);" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 46 "slowbasis_gb([
x^2*y - 1, x*y^2 - x], nosteps);" }}}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 8 "See Also" }}{PARA 0 "" 0 "" {TEXT -1 46 "ring(), altbasis_gb(), \+
quickbasis_gb(), mxgb()" }}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 13 "altb
asis_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "
" 0 "" {TEXT -1 56 "altbasis_gb() finds a Groebner basis for the given
 ideal" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Calling Sequences" }}
{PARA 0 "" 0 "" {TEXT -1 25 "altbasis_gb([f1,...,fs]);" }}{PARA 0 ""
0 "" {TEXT -1 34 "altbasis_gb([f1,...,fs], nosteps];" }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1
79 "[f1,...,fs] = a list of polynomials in the ring defined by ring()
\n   " }}{PARA 0 "" 0 "" {TEXT -1 67 "nosteps = the string th
at indicates that no steps are to be printed" }}}{SECT 0 {PARA 4 "" 0
"" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 267 "altbasis_gb([
f1,...,fs]) returns a Groebner basis of <f1,...,fs> using a slightly m
ore insightful version of Buchberger's algorithm that slowbasis_gb(). \+
 This basis is generally neither minimal nor reduced. Steps of constr
ucting the Groebner basis are also printed.\n" }}{PARA 0 "" 0 ""
{TEXT -1 120 "altbasis_gb([f1,...,fs], nosteps) returns the same thing
s, but steps of constructing the Groebner basis are not printed." }}}
{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 19 "ring(grlex, [x,y]);" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 36 "altbasis_gb([x^2*y - 1, x*y^2 - x]);" }}}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 45 "altbasis_gb([x^2*y - 1, x*y^2 - x],
 nosteps);" }}}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}{PARA
0 "" 0 "" {TEXT -1 47 "ring(), slowbasis_gb(), quickbasis_gb(), mxgb()
" }}}{PARA 4 "" 0 "" {TEXT -1 0 "" }}}{SECT 1 {PARA 3 "" 0 "" {TEXT
-1 15 "quickbasis_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose
" }}{PARA 0 "" 0 "" {TEXT -1 58 "quickbasis_gb() finds a Groebner basi
s for the given ideal" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 17 "Calling
 Sequences" }}{PARA 0 "" 0 "" {TEXT -1 27 "quickbasis_gb([f1,...,fs]);

" }}{PARA 0 "" 0 "" {TEXT -1 36 "quickbasis_gb([f1,...,fs], nosteps];
" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "
" {TEXT -1 79 "[f1,...,fs] = a list of polynomials in the ring define
d by ring()\n   " }}{PARA 0 "" 0 "" {TEXT -1 67 "nosteps = th
e string that indicates that no steps are to be printed" }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT -1 236 "
quickbasis_gb([f1,...,fs]) returns a Groebner basis of <f1,...,fs> usi
ng a streamlined version of Buchberger's algorithm. This basis is gen
erally neither minimal nor reduced. Steps of constructing the Groebne
r basis are also printed.\n" }}{PARA 0 "" 0 "" {TEXT -1 122 "quickbasi
s_gb([f1,...,fs], nosteps) returns the same things, but steps of const
ructing the Groebner basis are not printed." }}}{SECT 0 {PARA 4 "" 0 "
" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "ri
ng(grlex, [x,y]);" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 38 "quickb
asis_gb([x^2*y - 1, x*y^2 - x]);" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 47 "quickbasis_gb([x^2*y - 1, x*y^2 - x], nosteps);" }}}}
{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}{PARA 0 "" 0 "" {TEXT
-1 46 "ring(), altbasis_gb(), quickbasis_gb(), mxgb()" }}}}{SECT 1
{PARA 3 "" 0 "" {TEXT -1 8 "min_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 29 "To minimize a Groebner b
asis." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequence" }}
{PARA 0 "" 0 "" {TEXT -1 20 "min_gb([g1,...,gs]);" }}}{SECT 0 {PARA 4
"" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 97 "[g1,..
.,gs] = a list of polynomials that form a Groebner basis under the ter
m ordering of ring()." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis
" }}{PARA 0 "" 0 "" {TEXT -1 111 "min_gb([g1,...,gs]) returns a list o
f polynomials that form a minimal Groebner basis for the ideal <g1,...
,gs>." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 21 "ring(lex, [x,y,z,w]);" }}}{EXCHG
{PARA 0 "> " 0 "" {MPLTEXT 1 0 82 "gb := quickbasis_gb([3*x - 6*y - 2*
z, 2*x - 4*y + 4*w, x - 2*y - z - w], nosteps);" }}}{EXCHG {PARA 0 "> \+
" 0 "" {MPLTEXT 1 0 11 "min_gb(gb);" }}}}}{SECT 1 {PARA 3 "" 0 ""
{TEXT -1 8 "red_gb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }
}{PARA 0 "" 0 "" {TEXT -1 27 "To reduce a Groebner basis." }}}{SECT 0
{PARA 4 "" 0 "" {TEXT -1 16 "Calling Sequence" }}{PARA 0 "" 0 ""
{TEXT -1 20 "red_gb([g1,...,gs]);" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 105 "[g1,...,gs] = a lis
t of polynomials that form a minimal Groebner basis under the term ord
ering of ring()." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}
{PARA 0 "" 0 "" {TEXT -1 111 "red_gb([g1,...,gs]) returns a list of po
lynomials that form a minimal Groebner basis for the ideal <g1,...,gs>
." }}}{PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0 "
" {MPLTEXT 1 0 21 "ring(lex, [x,y,z,w]);" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 90 "gb := min_gb(quickbasis_gb([3*x - 6*y - 2*z, 2*x - 4*
y + 4*w, x - 2*y - z - w], nosteps));" }}}{EXCHG {PARA 0 "> " 0 ""
{MPLTEXT 1 0 11 "red_gb(gb);" }}}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 9 "
div_alg()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "
" 0 "" {TEXT -1 72 "div_alg() performs the division algorithm for mult

ivariable polynomials." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 16 "Callin
g Sequence" }}{PARA 0 "" 0 "" {TEXT -1 24 "div_alg(f, [f1,...,fs]);" }
}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Parameters" }}{PARA 0 "" 0 ""
{TEXT -1 32 "f = the polynomial to be divided" }}{PARA 0 "" 0 ""
{TEXT -1 0 "" }}{PARA 0 "" 0 "" {TEXT -1 43 "[f1,...,fs] = a list of p
olynomial divisors" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }
}{PARA 0 "" 0 "" {TEXT -1 242 "div_alg(f,[f1,...,fs]) returns a list. \+
 The list's first element is the remainder of the division with respec
t to the order and ring set by ring(). The list's second element is t
he list of quotients, with respect to the order of [f1,...,fs]." }}}
{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples" }}{EXCHG {PARA 0 "> " 0
"" {MPLTEXT 1 0 28 "ring([[1,1],[0,-1]], [x,y]);" }}}{EXCHG {PARA 0 ">
 " 0 "" {MPLTEXT 1 0 55 "div_alg(5*x^2 - y, [x^2 + y, x^4 + 2*x^2*y + \+
y^2 + 3]);" }}{PARA 0 "> " 0 "" {MPLTEXT 1 0 0 "" }}}}{SECT 0 {PARA 4
"" 0 "" {TEXT -1 8 "See Also" }}{PARA 0 "" 0 "" {TEXT -1 6 "ring()" }}
}}{SECT 1 {PARA 3 "" 0 "" {TEXT -1 9 "quot_mx()" }}{SECT 0 {PARA 4 ""
0 "" {TEXT -1 7 "Purpose" }}{PARA 0 "" 0 "" {TEXT -1 47 "quot_mx is a \+
matrix of quotients (see Synopsis)" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT
-1 16 "Calling Sequence" }}{PARA 0 "" 0 "" {TEXT -1 34 "quot_mx([f1,..
.,fs], [g1,...,gt]);" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 10 "Paramete
rs" }}{PARA 0 "" 0 "" {TEXT -1 79 "[f1,...,fs] = a list of polynomials
, where each fi is in the ideal <g1,...,gt>\n" }}{PARA 0 "" 0 ""
{TEXT -1 120 "[g1,...,gt] = a list of polynomial that forms a Groebner
 basis with respect to the monomial order and ring set by ring()" }}}
{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}{PARA 0 "" 0 "" {TEXT
-1 161 "quot_mx([f1,...,fs], [g1,...,gt]) returns a matrix of quotient
s. In other words, we have [g1,...,gt]*Q^T = [f1,...,fs], where Q^T r
epresents the transpose of Q." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "
Examples" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 21 "ring(grevlex, [x
,y]);" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 53 "quot_mx([x^2*y - 1
, x*y^2 - x], [-y + x^2, y^2 - 1]);" }}}}}{SECT 1 {PARA 3 "" 0 ""
{TEXT -1 6 "mxgb()" }}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 7 "Purpose" }}
{PARA 0 "" 0 "" {TEXT -1 85 "mxgb() computes a reduced Groebner basis \+
and its corresponding transformation matrix." }}}{SECT 0 {PARA 4 "" 0
"" {TEXT -1 17 "Calling Sequences" }}{PARA 0 "" 0 "" {TEXT -1 19 "mxgb
([f1,...,fs]); " }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 ""
{TEXT -1 26 "mxgb([f1,...,fs], nosteps]" }}}{SECT 0 {PARA 4 "" 0 ""
{TEXT -1 10 "Parameters" }}{PARA 0 "" 0 "" {TEXT -1 67 "[f1,...,fs] = \+
 a list of polynomials in the ring defined by ring()\n" }}{PARA 0 ""
0 "" {TEXT -1 67 "nosteps = the string that indicates that no steps ar
e to be printed" }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Synopsis" }}
{PARA 0 "" 0 "" {TEXT -1 545 "mxgb([f1,...,fs]) returns a list. The f
irst element, G=[g1,...,gt], is a Groebner basis of <f1,...,fs>, using
 the streamlined version of Buchberger's algorithm of quickbasi_gb(), \+
except that the basis is reduced. The second element a list represent
ing the coefficient matrix, showing how each polynomial in the Groebne
r basis is represented by the polynomials <f1,...,fs>. In other words
, [f1,...,fs]*Q^T=[g1,...,gt], where Q^T represents the transpose of Q

. Steps of minimizing a reducing the Groebner basis and its matrix ar
e also printed." }}{PARA 0 "" 0 "" {TEXT -1 0 "" }}{PARA 0 "" 0 ""
{TEXT -1 78 "mxgb([f1,...,fs], nosteps) returns the same things, but s
teps are not printed." }}}{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "Examples
" }}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 19 "ring([1,2], [x,y]);" }}}
{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 29 "mxgb([x^2*y - 1, x*y^2 - x])
;" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 38 "mxgb([x^2*y - 1, x*y^2
 - x], nosteps);" }}}{EXCHG {PARA 0 "> " 0 "" {MPLTEXT 1 0 0 "" }}}}
{SECT 0 {PARA 4 "" 0 "" {TEXT -1 8 "See Also" }}{PARA 0 "" 0 "" {TEXT
-1 48 "ring(), slowbasis_gb, altbasis_gb, quickbasis_gb" }}}}}{MARK "1
3" 0 }{VIEWOPTS 1 1 0 1 1 1803 1 1 1 1 }{PAGENUMBERS 0 1 2 33 1 1 }

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 **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 **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 **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

MAT 401 - Seminar in Mathematics

August 30, 2001

- Save the file as **.emacs** in your home directory. Read the file to learn how to the use of emacs with Macaulay 2
- Maple worksheet for the class.
- Uncommented Macaulay2 test worksheet from last class

Homework problems from Cox, Little, O'Shea: 1, 8, 13, page 12 and 2, 6, 9

Sorin Popescu

2001-08-17

MAT 401 - Seminar in Mathematics

September 18, 2001

- Uncommented Macaulay2 worksheet worksheet for the class (computing GCDs of polynomials in one variable)
- **Homework problems** from Cox, Little, O'Shea: 1, 3 page 51 and 1 page 58

Sorin Popescu

2001-08-17

MAT 401 - Seminar in Mathematics

September 25, 2001

- Uncommented Macaulay2 worksheet worksheet for the class (term orders)
- **Homework problems** from Cox, Little, O'Shea: 2, 5, 7, 8 page 58

Sorin Popescu

2001-08-17

MAT 401 - Seminar in Mathematics

October 2, 2001

- Macaulay2 worksheet worksheet for the class (basic elimination theory)
- **Homework problems** from Cox, Little, O'Shea: 5, 8 page 85, 7 page 92

Sorin Popescu

2001-10-02

MAT 401 - Seminar in Mathematics

October 9, 2001

- Macaulay2 worksheet worksheet for the class (the resultant of two univariate generic polynomials shows up in the Lex GB of the ideal generated by them)
- Maple 6 worksheet (the resultant of two univariate generic polynomials of degree 3)
- **Homework problems** from Cox, Little, O'Shea: 2 page 92 (use M2 to compute the corresponding GBs), 10 page 92, 1, 3 page 98.

Sorin Popescu

2001-10-6

MAT 401 - Seminar in Mathematics

October 11, 2001

- Macaulay2 worksheet worksheet for the class (the enveloping curve for a family of circles)
- **Homework problems** from Cox, Little, O'Shea: 7, 8 page 98 (use M2 to compute the corresponding GBs), 2 and 3 page 118.

Sorin Popescu

2001-10-11

MAT 401 - Seminar in Mathematics

October 16, 2001

- Mini project: Macaulay2 worksheet by Dimitry Nisterenko (elementary method to check if a graph can be colored with three colors or not, from D. Bayer's thesis)

Sorin Popescu

2001-10-16

MAT 401 - Seminar in Mathematics

November 1, 2001

- Mini project: A first Macaulay2 worksheet for testing the Conti-Traveso algorithm

Sorin Popescu

2001-11-1