

MINIMAT TUTORIAL

Joel Robbin

June 92

The purpose of this tutorial is to give you a feeling for how MINIMAT interacts with the user and what it can do. The tutorial is not intended to be a definitive account¹ of anything: MINIMAT's commands are explained more carefully in the sequel. To use this tutorial simply type the indicated commands at MINIMAT and compare the output with the text.

That portion of the text in this chapter which is actually produced by the computer is set in typewriter type

Like this.

All text typed into the computer by the user appears in typewriter type on a line beginning with the MINIMAT's prompt as in

```
#> A = [1 2; 3 4]
```

All other text in typewriter type represents output produced by MINIMAT as in

```
A = [ 1.000    2.000
      3.000    4.000 ]
```

1 Before You Begin

Before you use MINIMAT you should make a backup copy using the `copy` command from the Operating System. If you are completely new to computers you will have to learn a few things first, like (1) how to turn the computer on and off; (2) how to boot the operating system; (3) how to change the default disk/directory; (4) how to execute a program from the operating system; (5) how to copy files from one disk to another. These details will be peculiar to the particular computer system you are using and will not be explained here.

¹You can find a complete account of MINIMAT in the *MINIMAT Reference Manual*, but most users will find that information in this text together with the online information accessible through MINIMAT's *help facility* (explained below) is more than adequate for their needs.

The easiest way to use MINIMAT is to put all the files on the distribution diskette in one directory (if you have a hard disk) or diskette (if you do not) and to make that disk/directory the default. We shall assume you have done this.

2 Starting Up

We begin the session by typing `minimat` (followed by **[Enter]**) at the DOS prompt; this causes DOS to start the MINIMAT program. First MINIMAT displays a sign-on message. Pressing any key will cause MINIMAT to continue; MINIMAT will continue in any case after three seconds.

MINIMAT finds the file `mmlogin.m`² on the default directory and executes³ the commands in that file.

At the `#>` prompt, the following may be entered:

```
help                (to view the file minimat.hlp)
type filename.ext   (to view the file filename.ext)
edit filename.ext    (to edit the file filename.ext)
mmdemo              (a demonstration)
```

Press a key:

We press a key and MINIMAT continues by clearing the screen and printing the prompt.

3 The Prompt

When MINIMAT is waiting for input it issues a prompt:

```
#>
```

If at some point in the session MINIMAT has not issued the prompt when you think it should, it may be that you have not completed typing the last line (perhaps there is an open parenthesis). In this case (if you do not know

²If MINIMAT does not find this file it will simply print the prompt.

³If you press a key while MINIMAT is executing a command, MINIMAT asks you if you want to interrupt the calculation. If you respond affirmatively, then MINIMAT prints the prompt.

how to complete the statement correctly) you may press **[Esc]** (the escape key) to recover the prompt. If you press **[Enter]** before you have typed a complete statement, the message line at the top of the screen shows what symbol is needed to complete that statement.

4 Sample Session

We evaluate some simple MINIMAT expressions and assign them to variables. Note how the variable **ans** contains the result of the last expression evaluated which was not explicitly assigned to some variable.

```
#> 3*4-1

ans = 11.000

#> t=rand(1,1), x=cos(t), y=sin(t), x^2+y^2

t   = 0.557
x   = 0.849
y   = 0.528
ans = 1.000
```

Note that MINIMAT's notations are similar to those used in mathematics and most programming languages like BASIC. It uses the asterisk ***** to denote multiplication (if we write AB when we mean $A * B$, then MINIMAT will think of AB as a symbol by itself having nothing to do with either A or B). Note also that several statements may appear the same line if they are separated by commas.

At this point we ask for help.

```
#> help
```

```
*****      MINIMAT HELP FILE.      *****
```

```
Here are some MINIMAT statements:
```

```
a=[1 2 3; 4 5 6; 7 8 9]
b=ref(a)
x=[1 -2 1] '
a*x,    b*x
[m n] = size(a)
b=a'+a
c=a*b/2
c=a+2*c
```

```
*****
```

```
* Press the PgDn key to see more.      *
```

```
* Type Alt-x to return to MINIMAT      *
```

```
*****
```

(We show only one screenful of the `help` facility here.) We type [**Alt-X**] to return to MINIMAT.

At this point we use MINIMAT's convenient notations to perform Gaussian elimination. First we enter a simple matrix:

```
#> A = [ 1 2 3; 4 5 6; 7 8 9 ]
```

```
A = [ 1.000  2.000  3.000
      4.000  5.000  6.000
      7.000  8.000  9.000 ]
```

Next we subtract an appropriate multiple of the first row from the second to obtain a zero in the (2,1)-position:

```
#> A(2,:) = A(2, :)-A(2,1)*A(1, :)
```

```
A = [ 1.000  2.000  3.000
      0.000 -3.000 -6.000
      7.000  8.000  9.000 ]
```

We do the same for the third row to produce a zero in the (3,1)-position:

```
#> A(3,:) = A(3:)-A(3,1)*A(1,:)
```

```
A = [ 1.000  2.000  3.000
      0.000 -3.000 -6.000
      0.000 -6.000 -12.000 ]
```

We scale the second row to produce a one in the (2,2)-position:

```
#> A(2,:) = A(2:)/A(2,2)
```

```
A = [ 1.000  2.000  3.000
      0.000  1.000  2.000
      0.000 -6.000 -12.000 ]
```

We subtract a multiple of the second row from the third to produce a zero in the (3,2)-position.

```
#> A(3,:) = A(3:)-A(3,2)*A(2,:)
```

```
A = [ 1.000  2.000  3.000
      0.000  1.000  2.000
      0.000  0.000  0.000 ]
```

Finally we produce a zero in the (1,2)-position.

```
#> A(1,:) = A(1:)-A(1,2)*A(2,:)
```

```
A = [ 1.000  0.000 -1.000
      0.000  1.000  2.000
      0.000  0.000  0.000 ]
```

We use MINIMAT's `what` command to find out which `.M` MINIMAT can find.

```
#> what
```

```
Directory of C:\MM\*.M
```

```
  mmdemo.m      mmlogin.m      intinv.m      randsutr.m
```

rref.m	solve.m	rrefmul.m	randsoln.m
signum.m	next.m	adj.m	compan.m
sourfram.m	bed.m	rbasis.m	nbasis.m

We decide to find out what the .M file `rrefmul.m` does.

```
#> type rrefm

function [R M] = rrefmul(A)
%
%   R = reduced row echelon form of A
%   M = invertible matrix which transforms A to R:
%
%               M*A = R
%
[m,n] = size(A);
A = [A eye(m)];
A = rref(A);
R = A(:,1:n);
M = A(:,n+1:n+m);
```

The `type` command listed the file `rrefmul.m` on the screen. The comments at the beginning of the listing indicate that the function `rrefmul` accepts a matrix `A` as input and produces two matrices `R` and `M` as output: `R` is the reduced row echelon form of `A` and `M` is the matrix which pre-multiplies `A` to put `A` into reduced row echelon form.

We try it out. We press **[Alt-X]** to exit the `type` facility (just as we exited the `help` facility). Then we enter the same matrix we entered before.

```
#> A=[1 2 3; 4 5 6; 7 8 9]

A = [ 1.000    2.000    3.000
      4.000    5.000    6.000
      7.000    8.000    9.000 ]
```

Next we invoke the .M function using the form suggested by the heading (the first line of the file, following the keyword `function`):

```
#> [R M]=rrefmul(A)

R = [ 1.000   0.000  -1.000
      0.000   1.000   2.000
      0.000   0.000   0.000 ]

M = [ 0.000  -2.667   1.667
      0.000   2.333  -1.333
      1.000  -2.000   1.000 ]
```

We used the same letters – R , M , A – in calling `rrefmul` that are used in the text of the file `rrefmul.m`, but this is not necessary. Notice that the value for R is the same as the value for A we obtained before. We check the claim made in the listing of `rrefmul.m` above:

```
#> M*A

ans =[ 1.000   0.000  -1.000
       0.000   1.000   2.000
      -0.000   0.000   0.000 ]
```

It works! ($M \cdot A = R$.)

At this point we could terminate MINIMAT and return to DOS by typing

```
#> quit
```

Don't do this yet, however. The next few sections show how to use various aspects of MINIMAT's user interface, and you might as well try them out now.

5 Function Keys and Menus

If you press the function key **[F1]** at the MINIMAT prompt, you will bring up the **Help menu**. If you press the function key **[F10]** you will get the **Main menu**. (This is indicated on the menu line at the bottom of the screen, so you don't have to remember which key to press.) You select menu items by either using the up/down arrow keys to highlight the item you want

Command	Key Stroke	Command	Key Stroke
Page up	[PgUp]	Page down	[PgDn]
Cursor left	[←]	Cursor right	[→]
Cursor up	[↑]	Cursor down	[↓]
Start of line	[Home]	End of line	[End]
Top of file	[Ctrl-Home]	End of file	[Ctrl-End]
New line	[Enter]	To end & new line	[Shift-Enter]
Scroll up	[Ctrl-PgUp]	Scroll down	[Ctrl-PgDn]
Scroll right	[Ctrl-←]	Scroll left	[Ctrl-→]
Delete under	[Del]	Delete before	[Backspace]
Delete line	[Ctrl-Y]	Cancel statement	[Esc]
Help menu	[F1]	Local help	[F2]
Recall	[F3]	Main menu	[F10]
(The local help key [F2] usually displays this table)			

MINIMAT's Function Keys

and then selecting that item by pressing the **[Enter]** (Return) key or else by typing the letter enclosed in the brackets.

The Menu facility works by passing an appropriate text line to MINIMAT's command processor (this line will then appear on the screen) so you don't need to use the Menu system if you can remember the commands and like to type.

6 Snow and Color

If MINIMAT creates “snow” on your monitor when it writes to the screen, execute the **snow** command from the **change defaults** option on the main menu ([F10]). This will eliminate the snow but slow down the screen response. If you are using a monochrome monitor and the screen is hard to read, try changing colors using the **colors** command from the defaults menu. In either case MINIMAT will print the command line which causes the change. You can put a copy of this line(s) in your `mmlogin.m` file. If you cannot see the menus or the menu bar on your monitor type

```
#> colors bland
```

to change the colors to black and white. (After this command, the menu bar and menus should be visible on *any* monitor.)

7 Transcript

MINIMAT remembers everything it prints to the screen. In effect, it keeps a **transcript** of what has transpired. You can recall what has scrolled off the screen by using the cursor keys on the numeric keypad. Using the cursor keys by themselves will move the cursor around the screen. Pressing the control key [Ctrl] and a cursor key simultaneously will scroll the screen or move to the beginning or end of your transcript. The **saveas** command described below can be used to write the transcript to a file on the disk for subsequent viewing, editing, or printing.⁴

⁴If you know that you will want to save the transcript, it is best to use the **diary** command (described below) rather than the **saveas** command. The **diary** command saves the output as it is being produced so that it will not be lost in the event something goes wrong with the computer.

WARNING

If MINIMAT runs out of memory, it may be forced to discard some of the transcript. It does this from the front so that the output created first is discarded first.

If you type input when the cursor is before the prompt, the cursor will immediately move to the prompt. This makes it impossible for you to alter the transcript from MINIMAT⁵ except by adding new text at the end.

8 Recall

If you press the **recall key** [**F3**] at the prompt, the last statement you entered will be copied to the prompt where you can edit it and re-execute it if you like. Use the [**End**] key to move the cursor to the end of the line or the [**Ctrl-End**] key to move it to the end of the text (these will be the same if the cursor is on the last line of the text) before pressing [**Enter**].

If the cursor is before the active prompt, the recall key [**F3**] behaves differently depending on what kind of text the cursor is at. The message line at the top of the screen indicates what the recall key will do at the current cursor position. Try it out with the following:

1. Enter a statement like `A=rsnd(3,4)` with a typing error in it. Press [**F3**] and correct it to `A=rand(3,4)`,
2. Move the cursor to the line containing the error message produced by your erroneous statement and press [**F3**].
3. Move the cursor to a line containing an earlier statement you typed and press [**F3**].
4. Move the cursor to a matrix which MINIMAT has printed and press [**F3**].

⁵Of course, you can always edit any diary file which MINIMAT may have produced.

In each case observe how the message line at the top of the screen indicates what will happen when you press [F3].

9 Diary

MINIMAT provides the capability of keeping a record of all or part a session on the disk. If you type

```
#> diary a:\homework.1
```

MINIMAT will create a file called `homework.1` in the root directory on drive `a:` (this is the floppy disk drive) and echo all subsequent screen output to this file. You can substitute any filename you like for `a:\homework.1`. If the drive specification `a:` or the path specification (this is `\` in the example) is absent then the default is assumed so that the command

```
#> diary homework.1
```

will create the file `homework.1` in the default directory on the default disk (which is probably `c:` if your computer has a hard drive.) Finally if the file specification is missing altogether, as in

```
#> diary
```

MINIMAT will create a file named `diary.` in the default directory of the default disk. If a file matching the file specification already exists, MINIMAT will ask you whether you want to Overwrite the existing file, Append to it, or Cancel the `diary` command altogether.⁶

To stop echoing to the diary file you simply type

```
#> diary off
```

at the prompt. This closes the diary file. The diary file will also be automatically closed when you quit MINIMAT.

⁶WARNING: Other versions of Matlab may not do this.

10 SaveAs

If you want to save the entire transcript the command

```
#> saveas myfile
```

will save it in a disk file **myfile**. This is useful if you forgot to open a diary file. If appropriate, MINIMAT gives you the Overwrite, Append, Cancel option as for the diary file.

The **saveas** command operates independently of the **diary** command. Thus it is possible to create two (or more) copies of the transcript. The difference between the two commands is that the **diary** command saves the transcript as it is produced, while the **saveas** command saves it all at once. Remember that if you have produced a very long transcript, MINIMAT may have forgotten the earliest portion of it.

11 Viewing the Diary

If you want to look at the diary file from MINIMAT you can *type* it like any other text file. For example, the command

```
#> type a:\homework.1
```

will display that file on the screen. You can even type a diary file that is currently being created.

If you want to study your diary file away from the computer you will have to print it. If you have a printer, you can print the diary file using a suitable print program like the DOS **copy** program. This can be invoked from MINIMAT using a **shell escape**. It works as follows. You enter a command line to MINIMAT exactly as you would enter it in DOS, but you precede it with an exclamation point. DOS will process the command and return control to MINIMAT. For example, to send the file **a:\homework.1** to the printer you type⁷

```
#> !copy a:\homework.1 prn
```

⁷If nothing happens make sure the printer is turned on and is connected properly.

WARNING

Never execute a “Terminate but Stay Resident” program (that is, one which continues to work after other programs have been invoked) as a shell escape. The DOS PRINT program is such a program (since after it starts printing but before it is finished you can invoke other programs). The MINIMAT shell escape `!print filename` may cause the computer to freeze. This caveat applies to any program which invokes other programs, not just MINIMAT.

12 Comments

MINIMAT will ignore any line which begins with a percent sign `%`. This allows programmers to insert comments in `.M` files, but can also be used to insert text into diary files. For example, if you enter

```
#> % This is
#> % a comment.
```

MINIMAT will not respond but will copy the lines into the diary file if one is open. (If the percent sign `%` were not present, MINIMAT would print an error message.)

13 Homework

The diary file and `%` comments provide a handy way for keeping a record of your homework assignments. You can get a blank diskette⁸ (a double density 360K diskette ought to be more than adequate for the entire semester) and

⁸You will have to *format* the diskette before using it. This is done by executing a DOS (Disk Operating System) program called `FORMAT`.

store all your homework assignments on it, say in files named `homework.1`, `homework.2`, ... You can insert % comments in the diary file to keep a record of what problem you are solving and to answer any questions which may be part of the problem. As an example, suppose you have been assigned the following:

Exercise .1 *Type the following command into MINIMAT:*

```
#> a=4, a1=sqrt(a*a)
#> b=-3, b1=sqrt(b*b)
```

(`sqrt(c)` is MINIMAT's name for $\sqrt{c}$.) Why does $a = a_1$ but $b \neq b_1$?

Let's solve this simple problem to illustrate the proper format.

1. Open the diary file (MINIMAT responds):

```
#> diary a:\hw.0

* Diary file: A:\hw.0
* Opened at Date/Time: 88-07-09 14:19:40
* (Type DIARY OFF to close the diary.)
```

2. Identify yourself and the problem:

```
#> % Sally Student. Homework #1 due September 8.
#> % Exercise A
```

3. Do the computation:

```
#> a=4, a1=sqrt(a*a)

a  =  4.000
a1 =  4.000

#> b=-3, b1=sqrt(b*b)

b  = -3.000
b1 =  3.000
```

4. Answer the question:

```
#> sqrt(9)

ans = 3.000

#> % sqrt is the square root function.
#> % it computes the positive square root.
```

14 Editing and Shell Escape

MINIMAT has a built-in editor which you can use for writing `.M` functions (see page ??) or editing the transcript. You can invoke it from the main menu (**[F10]**) or by typing the `edit` command as in

```
#> edit myfunc.m
```

(Here `myfunc.m` can be replaced by any other legal filename.) If the file `myfunc.m` exists it is loaded for editing; otherwise, a new empty file by that name is created.

MINIMAT's built-in editor is rather primitive (though it should be adequate for the simple tasks required in this book). If you have another editor which you prefer you can invoke it from MINIMAT with the **shell escape** as in

```
#> !qedit myfunc.m
```

Indeed, any command line preceded by a `!` will be passed to DOS and executed as if you had typed it at the DOS prompt. This is faster than exiting MINIMAT and does not discard any variables you have defined. On returning from a shell escape (or MINIMAT's `edit` command) MINIMAT will check the time stamps on any `.M` functions in its memory. If it finds a non-match, it will clear all the `.M` functions from its memory so that subsequent calls invoke the newer versions.