Using the MIDAS Matlab Toolbox via GNU Octave

Allin Cottrell*

August 30, 2016

1 Introduction

Eric Ghysels' MIDAS Matlab Toolbox is the benchmark implementation of MIDAS (Mixed Data Sampling) methods in econometrics, written by the economist who pioneered these methods.[1] Octave is a free, open-source clone of Matlab, albeit—not surprisingly—not quite 100% compatible with the latter. It's often the case that for complex programs, some tweaks are required to get Matlab code to run on Octave, and the MIDAS Toolbox is a case in point. This note explains how to get the MIDAS Toolbox running on Octave. However, parts of the following may be of interest to Matlab users too, since some of the tweaks described below are also required to run Ghysels' code on Matlab R2016a for Linux.[2]

I begin by stating what's needed on the Octave side, then describe my modifications to the MIDAS Toolbox files. An annotated listing of these modifications is given in section 4. Section 5 provides instructions for installing the modified version of the Toolbox, and section 6 goes into some technicalities, for anyone who's interested.

2 On the Octave side

First, you will need the `io`, `optim` and `statistics` packages installed; and `optim` requires `struct`. If you don't already have these, it's simply a matter of executing the following commands at the Octave prompt (when online, of course):[3]

```
pkg install -forge io
pkg install -forge struct
pkg install -forge optim
pkg install -forge statistics
```

Second, the MIDAS code calls the Matlab function `addParameter`, which is not supported in Octave (or not in Octave version $\leq$ 4.0.3 anyway). You can fix this by downloading a patched version of the file `inputParser.m`. This can be found at

*Professor of Economics, Wake Forest University, Winston-Salem, North Carolina. Email: `cottrell@wfu.edu`.

[1] See Eric Ghysels' homepage at `http://www.unc.edu/~eghysels/` for links.

[2] I'm not in a position to check whether this also applies to current Matlab on platforms other than Linux.

[3] The following example assumes that you are managing your Octave installation yourself. If you are running a Linux distribution that offers such management (e.g. a Debian-based one), you may prefer to let it handle the package installation; consult the documentation for your distribution.

http://savannah.gnu.org/bugs/download.php?file_id=34282

and you can find some discussion of the point at

http://savannah.gnu.org/bugs/?45367

You should put this file in the directory from which you wish to call the MIDAS code. You can expect a run-time warning from Octave:

```
function ./inputParser.m shadows a core library function
```

but the substitution seems to be harmless.

3 On the MIDAS Toolbox side

There are four sorts of files in the Toolbox: "driver" files that exercise the Toolbox (`app*.m`); data files that are read by the drivers (in Microsoft Excel formats); top-level function files that are called directly by the drivers; and function files in the `private` subdirectory that provide underlying functionality.

Here's an overview of what has to be done to get the Toolbox working on Octave:

1. The driver files need to be modified in respect of their use of the `xlsread` function, which does not work in the same way on Octave. In addition, the data used in the DCC and GARCH examples must be prepared in a different format for use with Octave. Both of these points also apply to current Matlab for Linux.

2. There is a minor knock-on effect from revised handling of dates in the drivers: this affects the top-level function file `MIDAS_ADL.m`.

3. The Toolbox code calls some optimization functions (e.g. `fminunc`) with extra parameters appended to the call. This approach doesn't work on Octave, and has to be replaced by use of function handles.[4] This applies to the private files `bnls_adl_new.m`, `bnlsNN_adl_new.m` and `enls1_adl_new.m`. The revisions to these files are Matlab-compatible.

4. The GARCH-related code calls the Matlab functions `optimoptions` and `fmincon` which are not available in Octave. I have substituted calls to `optimset` and `sqp` respectively, but the substitutions are conditional on execution under Octave and should not affect behavior under Matlab.

Besides the above-mentioned changes, which are required to get things working on Octave, I made one further, cosmetic change to `MIDAS_ADL.m`: I added some alternative code—triggered only when running on Octave—to print regression output. The original code uses the `disp` function, applied to a two dimensional cell-array, to display regression results, and that works nicely under Matlab. On Octave, however, this use of `disp` produces a somewhat raw dump of the array in a single column, which is not so easy to read.

[4] See `https://lists.gnu.org/archive/html/help-octave/2009-08/msg00124.html` for discussion of this point.

4 Modified and added files

My modifications to the original MIDAS Toolbox files are available in the form of a "patch" file (see the following section for details). The patch makes several small changes as described in Table 1.

File	*Revisions*
`appADLMIDAS1.m`	data reading
`appADLMIDAS2.m`	data reading
`appADLMIDAS3.m`	data reading
`appADLMIDAS4.m`	data reading
`appDCCMIDAS1.m`	data reading, data files
`appGARCHMIDAS1.m`	data reading, data files
`MIDAS_ADL.m`	date handling
`MixFreqData.m`	date handling
`GarchMidas.m`	options handling, optimizer
`private/bnls_adl_new.m`	parameter passing
`private/bnlsNN_adl_new.m`	parameter passing
`private/enls1_adl_new.m`	parameter passing

Table 1: Files modified for use with Octave

In addition, the MIDAS for octave package provides the plain text data files listed in Table 2. They were prepared by opening the appropriate Excel files in Gnumeric, exporting the required data as text, then ensuring that missing values are represented as `NaN`.

File	*Source file*	*Range*
`DEXJPUS.txt`	`DEXJPUS.xls`	`B14:B9249`
`DGS10.txt`	`DGS10.xls`	`B16:B9251`
`INDPRO.txt`	`INDPRO.xls`	`B42:B576`
`NASDAQCOMbig.txt`	`NASDAQCOM.xls`	`B22:B11669`
`NASDAQCOM.txt`	`NASDAQCOM.xls`	`B22:B9257`
`NASdates.txt`	`NASDAQCOM.xls`	`A22:A11669`

Table 2: Added data files

5 Installation

Preliminary note: to complete the installation of what I'll call the "MIDAS octave" package you will need the patch utility. This is part of the standard kit on unix-type systems. A version for MS Windows can be found at `http://gnuwin32.sourceforge.net/packages/patch.htm`.

If you do not already have the MIDAS Matlab Toolbox installed, the first step is to install it. As mentioned above, you can find a link at `http://www.unc.edu/~eghysels/`. You will have to

create a MathWorks login if you don't already have one, then you can download `MIDASv2.0.zip`. Unzip this file in a suitable location.

Assuming you have `MIDASv2.0` and `patch` installed you should then

1. Download the MIDAS octave archive from
   `http://users.wfu.edu/cottrell/midas/MIDASv2.0_octave.zip`
2. unzip this archive in your `MIDASv2.0` directory; and
3. apply the patch `MIDASv2.0_octave.patch`.

If you're working in a terminal window on Linux, suitable shell commands to accomplish these tasks would look like the following:

```
cd ~/econometrics/MIDASv2.0 # adapt for your setup
wget http://users.wfu.edu/cottrell/midas/MIDASv2.0_octave.zip
unzip MIDASv2.0_octave.zip
patch -b -p0 < MIDASv2.0_octave.patch
```

Some notes on the above:

- The archive `MIDASv2.0_octave.zip` contains `license.txt` (Eric Ghysels' license file for the MIDAS software), the `patch` file which is used to modify the files noted in Table 1, and the plain text data files listed in Table 2.
- The unzipping will not of itself overwite any existing Toolbox files, but the `patch` command will. You may wish to make a full backup of the original MIDAS installation first.
- In the `patch` command, the `-b` flag tells the program to make back-ups (with suffix `.orig`) of the files it modifies; and `-p0` says to keep paths just as specified in the input file.

To verify that things have gone OK you might try creating a text file named (say) `octcheck.m` with the following content

```
pkg load io
pkg load optim
pkg load statistics
appADLMIDAS1
```

then doing

```
octave octcheck.m
```

6 Technical notes

For the sake of completeness and extensibility, this section explains a few technical issues relating to my revision of some of the MIDAS Toolbox files.

6.1 Reading new-style Excel data (xlsx)

The driver files pertaining to ADL modeling, `appADLMIDAS*.m`, all read from the new-style Excel file `mydata.xlsx`, each sheet of which which contains dates in the first column and numerical data in the second. Here's an example of such a read:

```
[DataY,DataYdate] = xlsread('mydata.xlsx','sheet1');
```

The first point here is trivial: both Octave and Matlab for Linux demand that the case of sheet names in the `xlsx` file is respected, so `sheet1` above (and later `sheet2`) must be replaced by `Sheet1` and `Sheet2`, respectively.

However, there's a more substantive issue. In the call above, the idea is that `DataY` gets the actual numerical data while `DataYdate` gets the dates in text form. That does not happen in Octave or Matlab for Linux: rather, `DataY` gets both columns as numeric values (so a $T \times 2$ matrix, with Microsoft-style date serial numbers in the first column) and `DataYdate` gets nothing, becoming an empty array. It's therefore necessary to make a change such as the following:

Original Toolbox code:

```
[DataY,DataYdate] = xlsread('mydata.xlsx','sheet1');
DataYdate = DataYdate(2:end,1);
```

Revised code:

```
% the start of Microsoft's clock
dateplus = datenum('1899-12-30', 'yyyy-mm-dd');
Y = xlsread('mydata.xlsx', 'Sheet1');
DataY = Y(:,2);
DataYdate = Y(:,1) + dateplus;
```

While the original `DataYdate` held dates as strings, the revised version holds them as Matlab serial numbers; since this variable is passed as an argument to `MIDAS_ADL`, a small accommodating change is required in `MIDAS_ADL.m`.

6.2 Reading old-style Excel data (xls)

The DCC and GARCH driver files read from a number of `xls` files. In this case the issue is that they read a specific range of cells, as in

```
y1 = xlsread('NASDAQCOM.xls','B22:B9257');
```

This is not supported by Octave (which anyway seems to have trouble with `xls` files in general) nor by Matlab on Linux.[5] While it is not a very elegant solution, I resorted to saving just the required portions of the `xls` data in question as plain text files, so that the above call becomes

```
y1 = load('NASDAQCOM.txt');
```

That is the explanation for the several added `txt` files listed in section 4.

6.3 Conditionality of code

I stated above that some of my changes to the MIDAS Toolbox files would be triggered only when running on Octave. The precise mechanism for this is conditioning on the existence of

[5] The message from Matlab is "`Range cannot be used in 'basic' mode. The entire sheet will be loaded.`"

the variable `octave_config_info`; my understanding is that this variable will always exist when Octave is running but never[6] when Matlab is running. So the trope in question is as follows:

```
if exist('octave_config_info')
    octave-specific code
else
    matlab-specific code
end
```

6.4 A funny thing

One change remains unexplained so far: a one-liner in `MixFreqData.m`. I don't understand why it's needed for Octave (it's not needed for Matlab on Linux), and I suspect it may betoken a subtle bug in the replacement `inputParser.m` (see section 2). (If anyone can enlighten me I'd be grateful.) Anyway, here's the setup. The variables `EstStart` and `EstEnd` are defined in string form in the driver files, to set the estimation period. For example we have in `appADLMIDAS1.m`:

```
EstStart = '1985-01-01';
EstEnd = '2009-01-01';
```

These variables are passed as "varargs" arguments to `MIDAS_ADL`, using the `addParameter` mechanism, from where they're passed on to `MixFreqData`, where we find the lines

```
estStart = datenum(estStart);
estEnd = datenum(estEnd);
```

to obtain numeric dates from the original string representations.[7] On Octave, the first of these statements works OK but the second fails! But going via a "date vector" works:

```
estEnd = datenum(datevec(estEnd));
```

Since this doesn't cause any trouble for Matlab, I haven't bothered to wrap it in Octave-only conditionality (section 6.3).

[6] OK, with very low probability: a Matlab user *might* happen to define a variable of this name.

[7] There are no typos in the lines of code above: the capitalization of the variable names is changed by the parameter specification of `MixFreqData`.