

Serious numbers

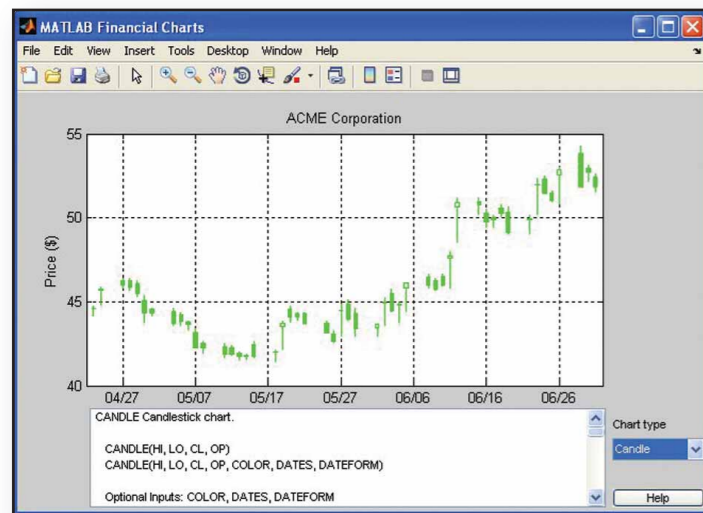


Figure 2

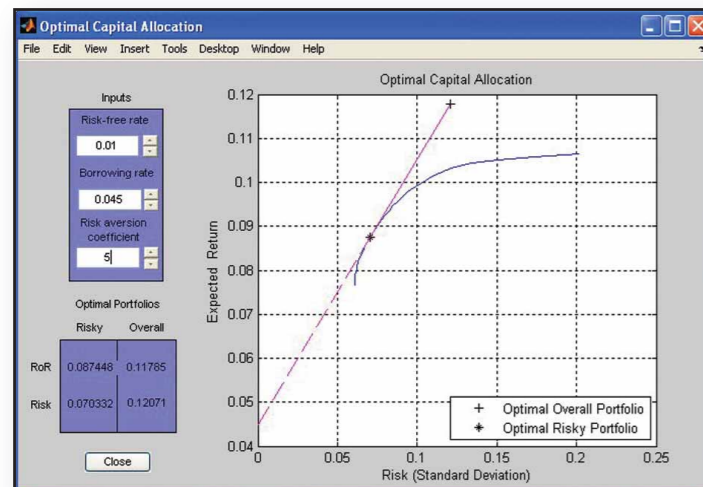


Figure 3

```

Editor - C:\Program Files\MATLAB\R2009a\toolbox\finance\findemos\capaldemo.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function capaldemo(command)
2 % CAPALDEMO Capital Allocation demo.
3 % This is a demo for the function PORTALLOC.M
4
5 % Author(s): M. Reyes-Katter, 03/05/98
6 % Copyright 1995-2005 The MathWorks, Inc.
7 % $Revision: 1.6.2.7 $ $Date: 2008/03/28 15:22:24 $
8
9 persistent hPRisk PRoR Pwts rb rf myHandles hOverall hRisky %#ok
10 persistent hcal1_minus hcal2_minus hcal1_plus hcal2_plus
11
12
13 if nargin < 1
14     command = 'start';
15 end
16
17
18 if strcmpi(command, 'start')
19
20     load capaldemo
21
22     % Create the user interface
23     hui = capalui;
24
25     % Collect all interesting handles in the UI's userdata
26     myHandles = setuihandles(hui);

```

Figure 4

command to (re)execute it immediately. It also allows you to invoke a couple of handy utilities from a right-click menu – the M-File builder and the Profiler. The M-File builder takes a command or series of commands and automatically builds them into a MATLAB script. So if you have been debugging a set of commands individually and wish to compile them into a single script it's just a case of Ctrl- or Shift-selecting the commands in the command line window history pane and clicking Create M-File. The M-File builder is also accessible via a check box in various other MATLAB windows, such as the Import Wizard (see 'Acquiring data' below).

The Profiler is used to improve the performance of your MATLAB code (M-Code) by timing the execution of its various elements, to assist in identifying infelicities such as unnecessary function calls. The Profiler works in conjunction with a utility called M-Lint, which in addition to highlighting outright errors will also identify code that is inefficient, such as unused input arguments or variables. If you are using parallel processing to speed things up, there is also a Parallel Profiler that shows how much time each processing session takes in evaluating each function and also how much time it spends communicating or waiting for communications with other sessions.

GUI

Though the command line sits at the core of MATLAB, the program also includes plenty of GUI driven elements, such as the financial charting and optimal capital allocation tools shown in Figures 2 and 3. Underlying all these GUIs is an M file; for example, a segment of the file for the optimal capital allocation tool is shown in Figure 4.

The opportunity here is that it is relatively trivial to build your own GUIs to front your own M-Files using the GUIDE tool shown in Figure 5a. As you add components to your GUI and save it, the associated M-File is automatically updated. For example, in Figure 5b the segment of code added in response to the 'TEST' command button in Figure 5a is highlighted.

Figure 6 shows a simple GUI built for the purposes of this review; it invokes several

At the coal face – MATLAB users' opinions



Miles Kumaresan,
Managing Director and
Head of Quantitative
Trading at TransMarket
Group Ltd

The group I head at TransMarket makes extensive use of MATLAB in designing and testing trading models. Personally, I've been using it for about eight years and found getting to grips with it straightforward.

You don't need huge programming expertise to become productive with MATLAB; if you are competent in VBA then that will take you a long way with MATLAB.

Though it isn't unique to MATLAB, its use of vector notation is particularly useful given our focus on time series data. You can apply a transformation to an entire time series in a single operation, without having to resort to loops. As a result, you can achieve a considerable amount in MATLAB with a single line of code.

Memory management is robust in that MATLAB can easily handle large data sets without instability. Given that we may often be working with tick data sets running into hundreds of thousands of rows and very large sparse matrices, that facility is obviously important.

However, while MATLAB's memory management is good, we find it slow when reading or writing data to disk. Data in plain text ASCII format that might only take five seconds to analyse once in memory can easily take twenty seconds to load. Java or C++ would read that data from disk in a fraction of the time.

Unless you are operating at the absolute bleeding edge of high frequency trading, MATLAB is fast enough for most purposes. We compile some of our MATLAB files as C++ DLLs and we find that we have managed to get the latency of calling those

libraries below one millisecond. While that would still be too slow for the very highest frequency trading, it is perfectly adequate for most trading activity.

David Knox, CEO,
I-TRADERS



We've been using MATLAB for about six months and so far we've been impressed. You can broadly split our usage into two areas, general R & D of trading strategies/indicators and real time publication of those to clients on the Web. The latter isn't fully live as yet, but we've been testing for a while and like the way MATLAB makes it easy to distribute data and models to those who don't actually have the application themselves. When compared with the alternative of testing something and then having to recode it from scratch in another language, MATLAB's ability to quickly and automatically compile M-Files into a distributable format is a real time saver.

While our main MATLAB user here had used it before joining us, other personnel have found it easy to pick up. As a result, we've been pleased with the productivity possible; for us it seems to strike exactly the right balance between usability and power.

The large number of MATLAB users worldwide is a major plus for us. We frequently find that the basic functions we need have already been written by somebody else and we can build on those as necessary when developing our proprietary algorithms. We've tried out a few of the open source toolboxes that are MATLAB-compatible and found they work very well, but without all the support/documentation you get with the MATLAB toolboxes it does admittedly take a bit longer to get up to speed.

Serious numbers

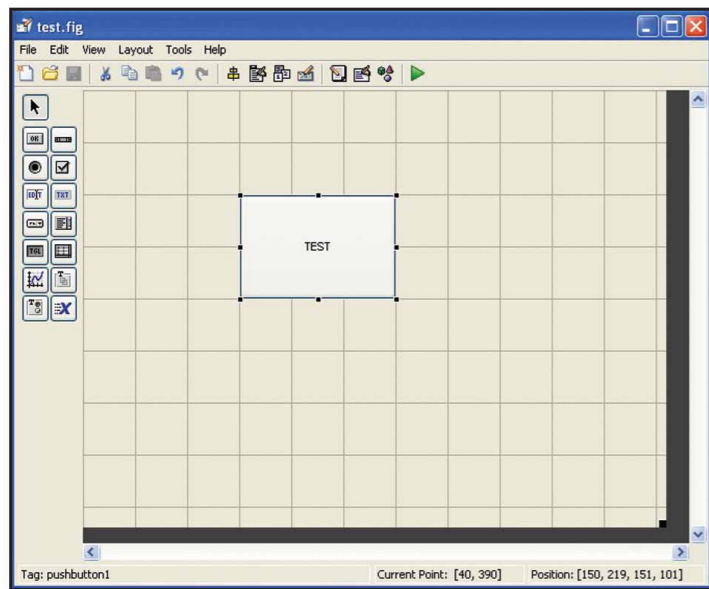


Figure 5a

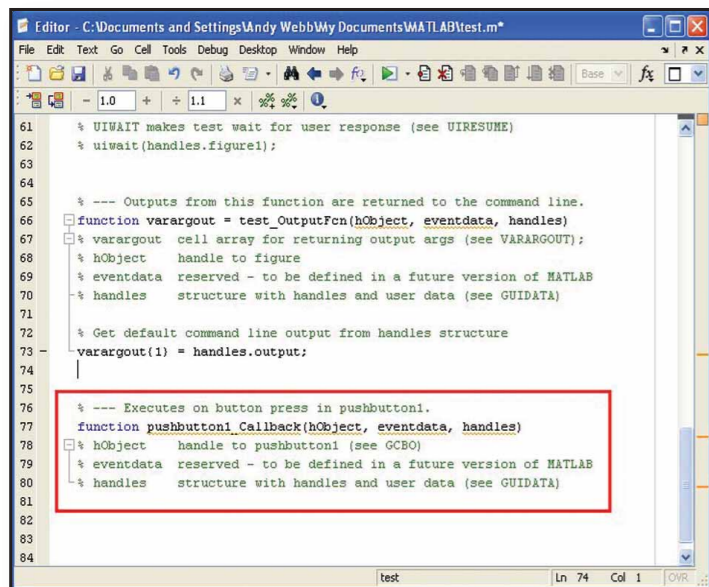


Figure 5b

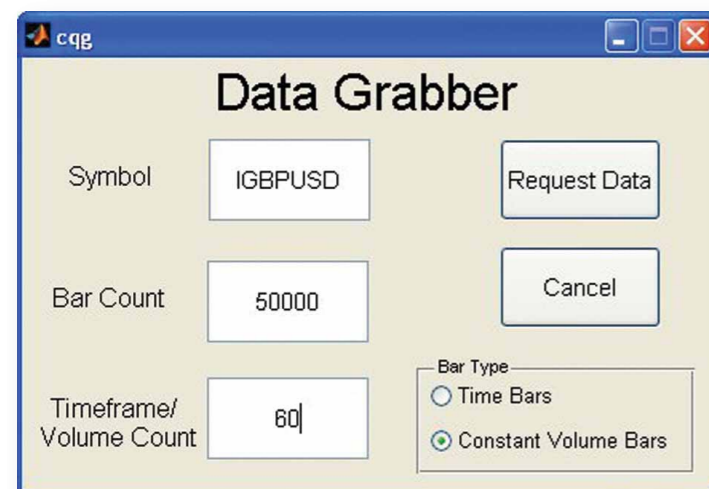


Figure 6

functions that in turn initiate a series of calls to CQG's API to retrieve either time or constant volume bars for a security. Though not shown in the GUI, it automatically saves the data retrieved to a text file and names it based on a combination of the current date/time and data entered into the GUI. Alternatively it would be fairly simple to expand the GUI to give the user a choice of outputs, such as assigning the data to a variable or writing it to a larger database.

Acquiring data

One of most fundamental tasks when building, testing and deploying financial models is obviously data acquisition. The MATLAB Datafeed Toolbox provides support for a number of popular data vendors, including Bloomberg, Reuters MDS and Thomson Datastream. Other data vendors and execution platforms have taken various alternative approaches to connecting to MATLAB. Those such as RTS Realtime Systems and 4th Story are both members of MATLAB's third party vendor program (see sidebar 'MATLAB Third-Party Products & Services').

CQG has written and made available a selection of M-Files that interface directly with the CQG API and allow both historical and real time data to be retrieved into MATLAB. In addition, a further set of M-Files make it possible for trading models running in MATLAB to route orders back out through the CQG API to various partner brokers/clearers.

If you want to import your own static data in a variety of common formats, MATLAB makes it pretty straightforward. Figure 7a shows the first step in its Import Wizard when importing a file of data in Excel format (note the 'Generate M-Code' check box at the bottom). As default, the wizard creates variables for the data on the basis of a preview (highlighted by the blue rectangle), which detected the column headers and numeric data separately. The disadvantage with this is that you end up with the data in a rather amorphous mass, which doesn't exactly help with manipulation. Figure 7b shows the solution, which is to choose the other radio button. This creates separate vectors (more on MATLAB's handling of

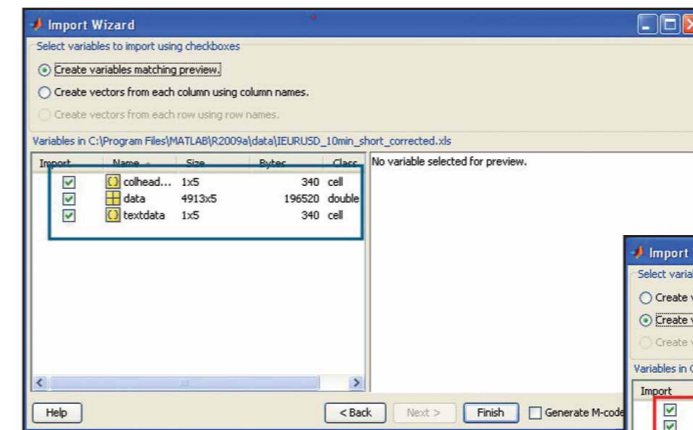


Figure 7a

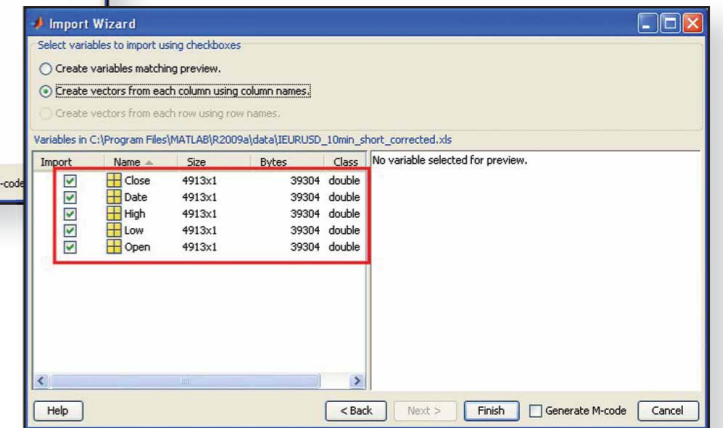


Figure 7b

vectors and matrices follows below) for each column and names the assigned variable after the column heading; as a result each individual time series is readily accessible.

Another advantage of separate vectors is that the min/max values for each series are visible in the variables window. Figure 8 shows an abortive attempt at a candlestick chart, with the reason for this error

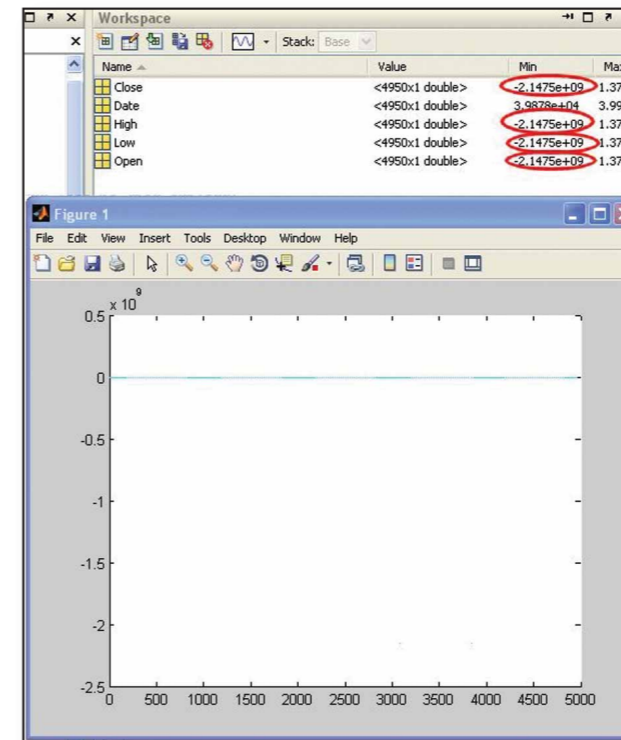


Figure 8

highlighted in red above; namely, that the data is corrupt and contains large negative numbers.

Incidentally, while MATLAB's Import Wizard is very effective, there are a few things to be aware of.

Those accustomed to using conventional date formats in historic data files will need to remember to convert them to serial dates first or the Import Wizard will not recognise them as a separate vector. I also found that when importing data from Excel files MATLAB spawned multiple

Excel processes in the background (see the segment from the Windows Task Manager in Figure 9). The snag was that these processes did not terminate when

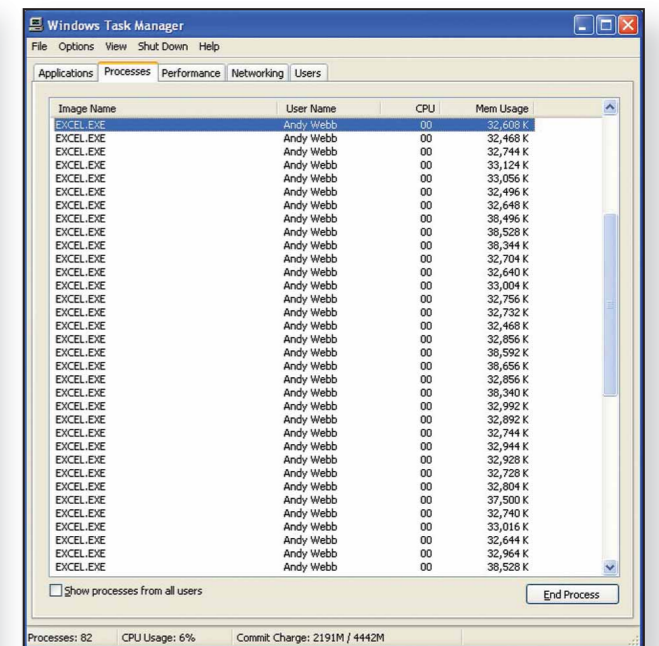


Figure 9

the wizard completed, so when importing a succession of Excel files I began encountering out of memory errors and discovered I had some thirty Excel processes running that had locked up nearly two gigs of memory.

An alternative way of importing data from Excel files is to use the xlsread function at the MATLAB command line, which worked fine. ▶

Serious numbers

MATLAB Third-Party Products & Services

MATLAB's Third-Party Products & Services program includes a number of specialist providers. Here, two such providers talk about their reasons for coupling their products with MATLAB



**John Melonakos, CEO,
AccelerEyes**

Our product, AccelerEyes Jacket, solves a key problem my three co-founders and I faced as PhD candidates processing large datasets using MATLAB. During our graduate studies, we found the limitations of conventional CPUs were resulting in slow calculation times for our

algorithms, sometimes taking days to run. To resolve that problem for our own use, the four of us started to collaborate on a method for running MATLAB code on a graphics processing unit (GPU). This collaboration then evolved into a commercial product called Jacket, a full production version of which was launched in January of this year.

We chose NVIDIA's Tesla GPU and CUDA environment as our target platform for Jacket as it was clearly the most functional and robust option. The GPU power computing market is still relatively young and the few competing options were clearly too immature or too low level.

We've had tremendous support from both The MathWorks and NVIDIA in developing Jacket, which has appreciably reduced our time to market. Our objective has been to screen MATLAB users from the GPU environment so that running Jacket is transparent for them and they can just focus on their core research. All they need to use are four MATLAB GPU functions that allow them to transfer MATLAB calculations to and from an NVIDIA GPU to benefit from its parallel processing power and thereby considerably reduce their overall processing times.

Obviously there are a lot of MATLAB users in finance and many of them have a need for parallel processing. As a result we've seen significant interest from the finance sector in GPU computing. In addition to the performance increase, many of them are also attracted by some of the other associated benefits - such as lower power and rack space requirements than conventional CPU computing.

**Anthony Tassone, VP
Algorithmic Trading
Solutions, RTS Realtime
Systems**

Our decision to partner with MATLAB was driven by the direction in which our clients were (and still are) going. An increasing number of them have been looking to separate out their analytics from their execution. This is primarily in the interests of efficiency; a trading engine has a mixture of tasks to perform, some of which are extremely time sensitive and some less so. For example, you don't typically need to know the gamma of an option a hundred times a second. As a result, you don't need or want to put the calculation for that gamma into your execution algorithm, as it will unnecessarily slow the overall trading process up and impact performance.

Therefore we have focused on establishing a bidirectional link between RTS Tango and MATLAB that facilitates the optimal exchange of data and instructions. For instance, MATLAB and Tango servers can sit alongside each other in a data centre and the Tango execution server can grab the calculated values it needs from the MATLAB server only when it needs them.

Most things you can do in the Tango client you can do in the MATLAB client, including coding trading strategies, so you have the flexibility to always execute the right tasks/code in the most appropriate place. In addition, it is possible to control Tango from MATLAB, so trading strategies can be turned on/off and new strategies compiled on the fly. This extends beyond individual markets, so users can also build risk management applications that control multiple Tango instances across a range of different trading venues.

The overall objective is that traders and quants can focus on developing ideas and models in MATLAB and then be able to leverage them on Tango. In an ideal world they will need to spend as little time as possible thinking about efficient execution (which they should be able to take for granted) and instead concentrate on adding value through exploring new trading ideas.



This only kicked off one background Excel process, but again this process did not terminate once the import completed and had to be killed manually.

Neither of these problems is a MATLAB bug; I discovered that it is in fact caused by certain other applications that may be running on the same machine (Google Desktop Search is a common culprit). When MATLAB spawns the first Excel process, these other applications immediately attach themselves to it and won't release it. As long as one is aware that this can happen, it's not a show stopper because the superfluous processes can be terminated manually. However, given the ubiquity of data imports from Excel files, it is something worthwhile remembering.

Manipulating data

Something long appreciated by MATLAB users across all industry segments has been its dexterity when manipulating vectors and matrices. Its ability to conduct complex transformations on even large data sets means that exploring the most intricate and large scale inter-market relationships is possible.

Even though MATLAB programs are interpreted not compiled, its facility with vectors is fundamental to its performance. MATLAB vectors and matrices can be manipulated most effectively when they are stored in columns in contiguous blocks of RAM. As a result any code you write that takes advantage of this behaviour will be more efficient - in many cases, by a large margin. This is particularly important for those migrating from other programs that are not tuned for vector/matrix performance and who might therefore be naturally inclined to write code that uses loops.

Loops will of course work in MATLAB, but should be avoided if a "vectorised" way of achieving the same result is available. For example, it is possible to loop through a dataset to perform an operation on the data points individually and create a new vector variable to hold the transformed data. However, at each iteration MATLAB will be adding individual elements to a vector that it will be increasing in size on the fly. This incurs an extra overhead in the form of unnecessary memory allocation calls, but in addition the individual transformed data points will not be stored contiguously in memory. As a result, even though the end result is a vector, any subsequent operations performed on it will be inefficient, as they will not be reading the data from a contiguous memory block. The alternative (and far more efficient) way to achieve the same result is wherever possible to size the output vector variable before writing to it.

MATLAB has many specific functions intended to make the use of looping redundant and thereby increase code efficiency. A classic example is re-dimensioning matrices. Assume you have a 10,000 row by 100 column matrix and you wish (for whatever reason) to re-dimension it to 1000 by 1000. You can write a loop that will (slowly) accomplish this, or you can use the purpose built MATLAB reshape function, which will do it nearly instantaneously. As your data sets increase in size, the benefits of thinking and programming in MATLAB vector terms obviously increase.

On that note, one of the most important factors that determine the maximum data set size that MATLAB can handle is the process limit (the maximum virtual memory a process or application can address). This is a function of the operating system and varies considerably. A standard 32-bit XP or Vista set up running 32-bit MATLAB is limited to 2GB. At the other end of the spectrum, 64-bit MATLAB running on a 64-bit OS will stretch to 8TB - which will hopefully cover most eventualities.

Crunching data

As mentioned earlier, a large selection of toolboxes and add-ons are available for MATLAB. There simply isn't space to deal with them all here, so only elements of those toolboxes likely to be of use to someone migrating to MATLAB from a more traditional analytics platform are covered. Therefore if you're burning to know Automated Trader's take on the Bioinformatics Toolbox, our apologies for any disappointment.

For those migrating from charting-based development platforms, the Financial Toolbox offers some familiar territory. The Financial Time Series GUI provides simple charting functionality and can also apply a variety of analytics (including common technical analysis studies) to displayed data. There are also various utilities for handling and converting dates, which work well. Useful - from bitter experience, one always seems to spend too much time manipulating between date formats and splitting out intraday time stamps; anything that eases this tedious drudgery is welcome.

The toolbox has some handy utilities to deal with the time consuming side of data housekeeping. Among many others, these include time frame rescaling, interpolation techniques for missing data values, differencing and data subset extraction (handy for in/out of sample testing). The descriptive statistics provided cover all the usual items, such as min, max, ►

Serious numbers

mean, covariance, standard deviation, correlation etc. Where appropriate, the calculations will automatically ignore non-numbers (NaNs).

The investment performance metrics in the Financial Toolbox include standard items like Sharpe ratio and risk adjusted return. However, a quick search of MATLAB Central reveals downloadable code for a variety of other metrics, including Sortino and Omega ratios. The Econometrics Toolbox offers more than sixty functions ranging from GARCH to multiple time series modelling to price/return utilities. While some of the functionality may be of more interest to economists than those building trading models, those looking to test for stationarity of pairs and synthetic pairs have their bases covered by a selection of Augmented Dickey-Fuller and Phillips-Perron unit root tests.

The Statistics Toolbox packs in several hundred functions ranging across probability distributions, regression analysis and hypothesis tests. As with the Econometrics Toolbox, an appreciable proportion of these will probably remain unused by many traders, but others such as analysis of distribution will be useful when evaluating model performance stats.

Several other toolboxes will be of relevance for those looking to extend their trading techniques. For example, the Optimization Toolbox has obvious application when testing model parameters and the Wavelet Toolbox for building filter applications for time series.

Get the trades out

One gripe we have heard from readers already using MATLAB is that while it delivers well as a development and testing environment, it doesn't make the final connection as a trade execution tool. While that may be strictly true, it is a little like complaining that your Ferrari is no good at hauling concrete blocks; it has no pretensions to do this and was never intended to do so.

In practice this perceived gap is already being filled by a growing number of providers. Ernie Chan's article over the page illustrates one MATLAB user's approach to this. At the same time, MATLAB's third party provider program (see sidebar 'MATLAB Third-Party Products & Services') has encouraged trading platforms such as RTS Realtime Systems to closely integrate their products. As mentioned earlier, CQG has

also made code libraries available that allow users to quickly establish data and trade execution connections between MATLAB and CQG. This trend obviously benefits end users and is only likely to accelerate as other data and trade execution providers wake up to the potential of connecting their applications to MATLAB.

Where these connecting vendors offer charting and trading platforms of the type described in the 'Ease versus power' section above, there is potentially an additional productivity benefit for those looking to use MATLAB. For traders intending to explore trading strategies with a more quantitative bias, a well-integrated combination of traditional chart based products and MATLAB is potentially ideal in terms of shorter time to productivity. Existing proprietary study or function outputs can be exported straight into MATLAB, rather than the studies or functions themselves having to be recoded for MATLAB at the outset. In the long term, recoding may be the more efficient route, but if sufficient interoperability is available, it isn't immediately mandatory.

In addition, the integrated charting interface of traditional software is typically good at allowing users to 'eyeball debug' logic errors. (For example, a limit order profit target inadvertently set twenty big figures away from the intended level.) Again, this is something that is perfectly reproducible in programs such as MATLAB, but if interoperability means that it doesn't have to be immediately recoded from scratch then there is both a comfort and speed benefit.

End result

From a 'How quickly can I do anything useful?' perspective, MATLAB looks good. While one could spend a lifetime exploring its every last esoteric nook and cranny, it is easy to get the basics up and running. Apart from the excellent documentation, webinars, seminars and sample code already mentioned, its basic

	USD	GBP	EUR
MATLAB	1950.00	1400.00	1950.00
Financial Toolbox	1500.00	1100.00	1500.00
Statistics Toolbox	1000.00	750.00	1000.00
Datafeed Toolbox	1000.00	750.00	1000.00
Econometrics Toolbox	1000.00	750.00	1000.00
Wavelets Toolbox	1000.00	750.00	1000.00
Optimization Toolbox	1000.00	750.00	1000.00


Figure 10: MATLAB pricing (as at April 27th 2009)

method of operation is simple to grasp. Anyone who has experience with VBA or the macro languages used by many charting-based trading platforms should start being productive within a week. Those without such experience shouldn't take much longer. Even if you can't find the answer to a specific question in the help files, the size of the user community makes it reasonably likely that someone has already written the trading-related function you need (or one you can adapt) and published it on MATLAB Central or any one of the numerous other MATLAB-focused sites on the Web.

MATLAB pricing is not what you might call trivial, but given the level of functionality available it certainly isn't unreasonable either (see Figure 10). The prices in Figure 10 are for perpetual licences, but if you want support and upgrades after one year you have to subscribe to The MathWorks Software Maintenance Service. This isn't exorbitant; for example, a one year subscription for MATLAB (without any toolboxes) is GBP252 and most toolbox annual subscriptions are GB69.

So is MATLAB good value? From the perspective and intended demographic of this review, I think the answer has to be a resounding yes. OK, the price of the core MATLAB application isn't exactly chickenfeed, but you do get a lot of functionality. The same applies to the toolboxes, but the slight snag from the perspective of someone expanding their trading methods and using MATLAB for the first time is deciding which toolboxes are likely to be relevant at the outset.

One obvious solution is to take a one month free trial. Another is to try the free toolbox from www.spatial-econometrics.com, which also contains equivalent functions to those in the MATLAB Statistics and Finance Toolboxes. As it is written in M-Code, it is MATLAB-compatible - though the help/support functionality is understandably smaller scale.

Whichever route you take, the fact remains that MATLAB has the potential to support just about anything you could conceivably wish to achieve in terms of building and testing (and through third party vendors, also deploying) trading models. We like it... 



Chan's encounter

We've met Ernest P Chan before, back in the Q2 2008 issue, when he wrote about combining machine learning and regime switching to achieve profitability. Here, he tells us the story of an encounter with MATLAB that has transformed his order execution. Do you know how to use MATLAB as an automated execution system?

Many traders are familiar with MATLAB as a powerful software platform for backtesting trading strategies. This is especially true for those who would like to analyze and trade a large number (maybe thousands) of stocks simultaneously. MATLAB is a language that is built around matrix manipulation and processing, so many calculations involving multiple stocks are just as easy as calculations involving a single stock.

In my book *Quantitative Trading* (Wiley 2008), I have described a number of examples of how backtesting is usually done in MATLAB. However, it was also true that MATLAB suffered from a major deficiency relative to more familiar trading platforms such as TradeStation - after a strategy has been backtested, it wasn't easy 