MATLAB Tips and Tricks
You Need to Know

Yair Altman
UndocumentedMatlab.com
altman@gmail.com
Key takeaways

• Tips and tricks to improve productivity with MATLAB
  o Usability and interactivity
  o Visualizing large data sets
  o Run-time performance
  o MATLAB Central

• Some tips/tricks are easy to apply and have great effect

• MATLAB can be used to create professional-quality user-facing applications
New App Designer in MATLAB

• GUIDE replacement in native MATLAB
• Web-based figures
• Great-looking widgets/dials/knobs
• Integrated layout designer, code editor
• Enables using CSS for professional-looking UI
Old non-web GUI

• Programmed using GUIDE or straight m-code
• Based on Java Swing UI components
• Built-in support for HTML formatting (no Java knowledge/programming required!)
• Enables using Java for polished professional UI
• Might become unsupported in a future release
fontStr = @(font) ['<html><font face="" font "">' font];
htmlStr = cellfun(fontStr, listfonts, 'uniform',false);
uicontrol('style','popupmenu', 'string',htmlStr, 'pos',[20,350,60,20]);
HTML formatting of menu labels

```matlab
set(hMenuItem, 'Label',... ...
  '<html>2: C:\My Documents\doc.txt<br>'...
  '<font size="-1" face="Courier New" color="red">&nbsp;&nbsp; '...
  'Date: 15-Jun-2011 13:23:45<br>&nbsp;&nbsp; Size: 123 KB']);
```

```matlab
txt1 = '<html><b><u><i>Save</i></u>';
txt2 = '<font color="red"><sup>this file';
txt3 = '<br>this file as...';
set(findall(hFig,'tag','figMenuFileSave'),   'Label',[txt1,txt2]);
set(findall(hFig,'tag','figMenuFileSaveAs'), 'Label',[txt1,txt3]);
```
Visualizing complex data in a GUI

• General concepts
  o Focus user attention (summary, anomalies)
  o Hide less-important data by default
  o Combination of tables (quantitative) & charts (qualitative)
  o Enable data drill-down (e.g., heat-map)

• Enable easy sorting, filtering, searching
  o Pre-sort displayed data

• Summarize data in separate tabs, sub-tables

• Use **GUI Layout Toolbox (GUILT)**: large tabs, flex-panels
  o Excellent free toolbox on the MATLAB File Exchange
Real-world example

![Pairformax - pairs trading, max performance - pairformax.com](Image)

**Results for BERY / PKG (1 Hour)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Sector</th>
<th>Bar size</th>
<th>APR</th>
<th>% Sharpe</th>
<th>Max DD</th>
<th>% Max DD</th>
<th>durring</th>
<th>trades</th>
<th>% win</th>
<th>win</th>
<th>Avg loss</th>
<th>Avg hold</th>
<th>Profit</th>
<th>Profit/Trade</th>
<th>OK?</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERY</td>
<td>Consumer</td>
<td>1 Hour</td>
<td>55.87%</td>
<td>-0.04%</td>
<td>-6.66%</td>
<td>37.0 days</td>
<td>28</td>
<td>61%</td>
<td>2.76%</td>
<td>-1.60%</td>
<td>38</td>
<td>$5,856</td>
<td>$209</td>
<td></td>
<td></td>
<td>Maybe test strategy</td>
</tr>
<tr>
<td>PKG</td>
<td>Packaging &amp; Containers</td>
<td>9 months</td>
<td>55.87%</td>
<td>-0.04%</td>
<td>-6.66%</td>
<td>37.0 days</td>
<td>28</td>
<td>61%</td>
<td>2.76%</td>
<td>-1.60%</td>
<td>38</td>
<td>$5,856</td>
<td>$209</td>
<td></td>
<td></td>
<td>Maybe test strategy</td>
</tr>
<tr>
<td>PKG</td>
<td>Packaging Corporation of America (Consumer Goods - Packaging &amp; Containers)</td>
<td>9 months</td>
<td>55.87%</td>
<td>-0.04%</td>
<td>-6.66%</td>
<td>37.0 days</td>
<td>28</td>
<td>61%</td>
<td>2.76%</td>
<td>-1.60%</td>
<td>38</td>
<td>$5,856</td>
<td>$209</td>
<td></td>
<td></td>
<td>Maybe test strategy</td>
</tr>
</tbody>
</table>

**BERY / PKG (1 Hour) backtesting results (test strategy)**

<table>
<thead>
<tr>
<th>Entry date</th>
<th>Exit date</th>
<th>Duration</th>
<th>Side</th>
<th>Z in</th>
<th>Z out</th>
<th>Spread in</th>
<th>Spread out</th>
<th>BERY in</th>
<th>PKG in</th>
<th>PKG β in</th>
<th>BERY out</th>
<th>PKG out</th>
<th>PKG β out</th>
<th>Profit</th>
<th>Profit/Trade</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/22/2015 14:00</td>
<td>10/26/2015 11:00</td>
<td>3.9 days</td>
<td>Long</td>
<td>-1.15</td>
<td>-1.35</td>
<td>-1.74</td>
<td>-2.04</td>
<td>33.40</td>
<td>64.70</td>
<td>35.14</td>
<td>33.76</td>
<td>66.87</td>
<td>35.79</td>
<td>-230.20</td>
<td></td>
</tr>
<tr>
<td>10/27/2015 15:00</td>
<td>10/29/2015 14:00</td>
<td>2.0 days</td>
<td>Long</td>
<td>-2.06</td>
<td>-1.68</td>
<td>-3.13</td>
<td>-3.92</td>
<td>32.72</td>
<td>67.01</td>
<td>35.84</td>
<td>33.18</td>
<td>68.13</td>
<td>36.17</td>
<td>-26.12</td>
<td></td>
</tr>
<tr>
<td>11/03/2015 12:00</td>
<td>11/10/2015 14:00</td>
<td>5.1 days</td>
<td>Long</td>
<td>-2.43</td>
<td>-1.64</td>
<td>-3.57</td>
<td>-2.48</td>
<td>32.60</td>
<td>68.47</td>
<td>35.27</td>
<td>32.98</td>
<td>65.76</td>
<td>35.46</td>
<td>-512.31</td>
<td></td>
</tr>
</tbody>
</table>

© Yair Altman
UndocumentedMatlab.com
Run-time performance

1. MATLAB’s built-in Profiler tool
   - Very useful, but largely under-appreciated

2. General programming tips
   - Caching
   - Loop-invariant hoisting (LIH)

3. Parallelization
   - Minimize broadcast data
   - Vectorization vs. CPU/GPU parallelization

4. Graphics
**Built-in Profiler tool in MATLAB**

![Profiler window](image)

**Profile Summary**

Generated on 10-Jul-2015 00:38:26 using performance time.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Calls</th>
<th>Total Time</th>
<th>Self Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ode23</code></td>
<td>1</td>
<td>0.577 s</td>
<td>0.191 s</td>
</tr>
<tr>
<td><code>funfun/private/odearguments</code></td>
<td>1</td>
<td>0.378 s</td>
<td>0.103 s</td>
</tr>
<tr>
<td><code>lotka</code></td>
<td>34</td>
<td>0.262 s</td>
<td>0.262 s</td>
</tr>
<tr>
<td><code>odeget</code></td>
<td>11</td>
<td>0.013 s</td>
<td>0.011 s</td>
</tr>
<tr>
<td><code>funfun/private/odefinalize</code></td>
<td>1</td>
<td>0.004 s</td>
<td>0.004 s</td>
</tr>
<tr>
<td><code>funfun/private/odemass</code></td>
<td>1</td>
<td>0.003 s</td>
<td>0.003 s</td>
</tr>
<tr>
<td><code>odeget/getknownfield</code></td>
<td>11</td>
<td>0.003 s</td>
<td>0.003 s</td>
</tr>
<tr>
<td><code>funfun/private/odeevents</code></td>
<td>1</td>
<td>0.001 s</td>
<td>0.001 s</td>
</tr>
</tbody>
</table>

**Total Time Plot**

(dark band = self time)

**Self time** is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.
% faster variant of datestr, for integer date values since 1/1/2000
function dateStrs = datestr2(dateVals, varargin)
    persistent dateStrsCache
    persistent dateValsCache
    if isempty(dateStrsCache)
        origin = datenum('1-Jan-2000');
        dateValsCache = origin:(now+100);
        dateStrsCache = datestr(dateValsCache, varargin{:});
    end
    [tf,loc] = ismember(dateVals, dateValsCache);
    if all(tf)
        dateStrs = dateStrsCache(loc,:);
    else
        dateStrs = datestr(dateVals, varargin{:});
    end
end  % datestr2
Caching example – *datestr*

% Prepare a 1000-vector of dates, starting 3 years ago until today
>> dateVals = fix(now)+(-1000:0);

% Run the standard datestr function → 50mS
>> tic; s1=datestr(dateVals); toc
⇒ Elapsed time is 0.049089 seconds.

>> tic; s1=datestr(dateVals); toc
⇒ Elapsed time is 0.048086 seconds.

% Now run datestr2 function → 0.3 mS (x150 faster)
>> tic; s2=datestr2(dateVals); toc
⇒ Elapsed time is 0.222031 seconds. % cache preparation: 222 mS

>> tic; s2=datestr2(dateVals); toc
⇒ Elapsed time is 0.000313 seconds. % subsequent calls: 0.3 mS
Loop-invariant hoisting (LIH)

for iter = 1 : 1000
    newData = constantFunction();
    result(iter) = max(max(newData)) + rand(1);
end

newData = constantFunction();
maxNewData = max(max(newData));
for iter = 1 : 1000
    result(iter) = maxNewData + rand(1);
end

result = maxNewData + rand(1,1000); %vectorized
Parallel processing

• Use built-in implicit parallelization (vectorization)
• Use explicit parallelization with the Parallel and Distributed Computing toolboxes
• Control # workers: `parpool(feature('numcores'))`
• Minimize amount of broadcast data
• Have realistic speedup expectations (*Amdahl’s law*)
Graphics performance

• Install latest graphics driver on your computer
• Reduce the number of graphic objects (plot markers, non-visible lines/patches)
• Multi-segment lines using NaN
• Update existing graphic objects, don’t re-plot
• Minimize `drawnow` calls, use 'limitrate'
Real-world example
Multi-segment lines using NaN

% Original (slow) code: 33 secs
line([lons1'; lons2'], [lats1'; lats2']);

% Faster code: limit the display to the axes limits
hAxes = handle(gca); %trick: make hAxes.XLim work on old MATLABs
lonLimits = hAxes.XLim;
latLimits = hAxes.YLim;
isOk = (within(lons1,lonLimits) | within(lons2,lonLimits)) & ... 
     (within(lats1,latLimits) | within(lats2,latLimits));
line([lons1(isOk)', lons2(isOk)'], [lats1(isOk)', lats2(isOk)']);

% Multi-segment line using NaNs (fastest): 0.6 secs
lons = [lons1'; lons2'; nan(1,numel(lons2))];
lats = [lats1'; lats2'; nan(1,numel(lats2))];
line(lons(:), lats(:));
data = [0];
while someCondition()
    newValue = getNewValue();
    data = [data, newValue];
    plot(hAxes, data);
    drawnow
end

% Faster, less flicker: update existing graphic object
hLine = plot(hAx, 0,0);  % initial line
while someCondition()
    hLine.XData(end+1) = hLine.XData(end) + 1;
    hLine.YData(end+1) = getNewValue();
    % alternatively: set(hLine, 'XData',xVals, 'YData',yVals)
    drawnow limitrate
end
MATLAB Central

- **http://mathworks.com/matlabcentral**
- Huge repository of community content
  - Blogs
  - File Exchange
  - Usage examples
  - Forum (“Answers”, newsgroup)
  - Coding challenges (“Cody”)
- Use the ratings for high-quality content
Conclusions

• MATLAB is **not** inherently ‘slow’

• MATLAB GUI is **not** inherently ‘simplistic’

• MATLAB can be used to create professional-quality user-facing applications, while enjoying MATLAB’s benefits (RAD, functionality, reliability)

• As with everything in life, there are tradeoffs:
  o For better performance, invest extra code development time
  o For better appearance, invest extra design time
  o For better usability, use Java controls