1 Overview

Org-mode and matlab-mode provide an efficient and effective system for creating scientific documents which contain MATLAB code and/or Simulink models along with the results of these. The results of running MATLAB code or simulating Simulink models is placed into the org-mode file by org-mode using org babel. Org babel is org-mode's ability to execute source code within org-mode files and optionally insert the results back in to the org-mode file. You define source code in code blocks, e.g.

```
#+begin_src LANGUAGE <OPTIONS>
  <CODE>
#+end_src
```

2 Example

With org-mode you can embed semantically colored code such as MATLAB within your document and semantically edit it using "Org -> Editing -> Edit Source Example" menu or `C-c '`. For example, here's a MATLAB enumeration class:

```
classdef WeekDays
   enumeration
       Monday, Tuesday, Wednesday, Thursday, Friday
   end
end
```

You can use org-mode babel to evaluate MATLAB code blocks. The evaluation is done by sending the MATLAB code to the `*MATLAB*` buffer created by `M-x matlab-shell`. To do the evaluation, the `*MATLAB*` must be waiting for input (showing the "»" prompt). If you type `C-c C-c` in the following code block, org-mode will evaluate the code in the `*MATLAB*` buffer and insert the value of `ans` just below the code block.

When the matlab code block header contains ":results verbatim", the value of the MATLAB `ans` variable is processed using `writematrix(ans, orgTmpFile, 'Delimiter', 'tab')` and then the contents of the *orgTmpFile* is inserted under the "#+RESULTS:"

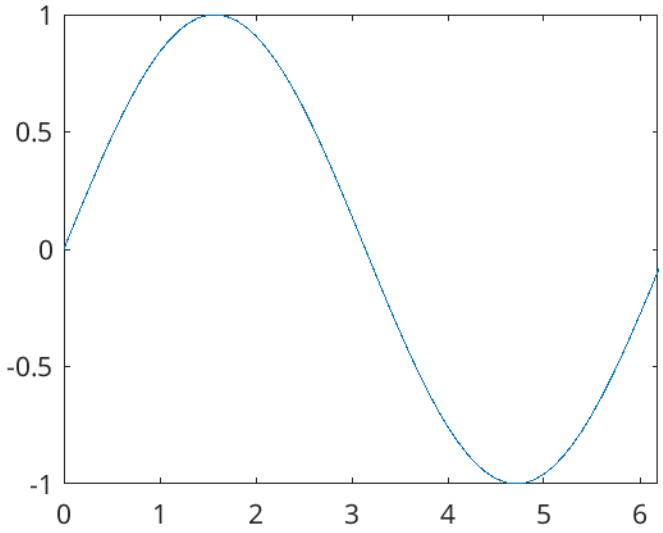
```
a = 2 + 3;
ans = magic(a);
```

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

You can also use org-mode babel evaluate MATLAB code blocks to plot and insert figures back in to this file as well as the published (exported) html, LaTeX, pdf, odx (word), etc. file. To do this we use a matlab code block with ":results file graphics" header argument that instructs org to run the MATLAB code in the `*MATLAB*` buffer created by `M-x matlab-shell`. which calls plot and the resulting figure is printed to the ':file NAME.png' header argument. The ':exports both' header argument says when exporting, keep the MATLAB code and also the figure when exporting.

For example,

```
t = [0 : 0.1 : 2*pi];
y = sin(t);
plot(t, y);
set(gcf, 'PaperUnits', 'inches', 'PaperPosition', [0 0 4 3]) % Set the size to 4" x 3"
```



You can also use LaTeX directly, for example:

$$y(t) = f_o(t, x_c, x_d, u, P) \quad \text{– outputs} \tag{1}$$
$$\dot{x_c}(t) = f_d(t, x_c, x_d, u, P) \quad \text{– derivatives} \tag{2}$$
$$x_d(t+h) = f_u(t, x_c, x_d, u, P) \quad \text{– update} \tag{3}$$

3 Setup and Export

1. Enable MATLAB code block export.

   To enable exporting of org containing matlab code blocks, you need to

   `M-x customize-variable RET org-babel-load-languages RET`

   and add matlab, then 'Save for future sessions' using the 'State' button.

   If matlab has not been added to org-babel-load-languages, when you try to evaluate a matlab code block, you will see

   `org-babel-execute-src-block: No org-babel-execute function for matlab!`

2. Use these files as a template for your org files.

   ```
   cd your-working-directory
   cp /path/to/Emacs-MATLAB-Mode/examples/matlab-and-org-mode.org your-file.org
   cp -r /path/to/Emacs-MATLAB-Mode/examples/css .      # If exporting to html
   ```

   Notice that within the *.org file there are several `#+<comments>`. These setup for LaTeX/PDF and HTML export.

3. Configure HTML export.

   You need the htmlize package (https://melpa.org/#/htmlize) to get coloring for HTML export. For HTML export we set the "#+html_head_extra" properties in our org file to configure CSS.

   HTML export uses

   - css/styles-from-org.css. This is generated by running

     `M-x org-html-htmlize-generate-css`

     and you'll want to update this for your version of Emacs.

- `css/styles.css`. This contains customizations which you can edit as desired.

4. Configure PDF export.

   To get colored, better looking PDF, use the minted package. This setup can go in your `~/.emacs`:

```
(defun setup-org-pdf ()
  "Customize org PDF generation for color and more."
  (if (not (boundp 'org-latex-src-block-backend))
      (message "Unable to configure org PDF export because it is too old.")
    (setq org-latex-src-block-backend 'minted
          org-latex-packages-alist '(("cache=false" "minted"))
          org-latex-minted-options '(("xleftmargin" "1em")
                                     ("breaklines" "true")
                                     ("fontsize" "\\small"))
          org-latex-image-default-width ""
          ;; Default value of org-latex-pdf-process does not include -shell-escape which is
          ↪  needed for minted
          ;; Also improve latex log file error messages by adding -file-line-error
          org-latex-pdf-process '("%latex -file-line-error -shell-escape -interaction
          ↪  nonstopmode -output-directory %o %f"
                                  "%latex -file-line-error -shell-escape -interaction
                                  ↪  nonstopmode -output-directory %o %f"
                                  "%latex -file-line-error -shell-escape -interaction
                                  ↪  nonstopmode -output-directory %o %f")
          ;; Keep *.log files to aid in debugging.
          org-latex-logfiles-extensions (remove "log" org-latex-logfiles-extensions))

    ;; Color the hyper links, see
    ;;
    ↪  https://tex.stackexchange.com/questions/823/remove-ugly-borders-around-clickable-cross-referenc
    (add-to-list 'org-latex-default-packages-alist

                 ↪  '("colorlinks=true,linkcolor={red!50!black},citecolor={blue!50!black},urlcolor={bl
                   "hyperref" nil))))

(eval-after-load "ox-latex"
  '(setup-org-pdf))
```

5. Export.

   After this setup, you can use the "Org -> Export/Publish" or `C-c C-e` to export to HTML, PDF, etc.