# MATLAB and LaTeX integration

Dr. James William Andrews

January 11, 2024

## 1 Why MATLAB? Why LaTeX?

MATLAB and LaTeX integration can be beneficial when working on projects to help keep the results in the report or documentation up to date. This can prevent the MATLAB code base and the written results from going out of sync. Furthermore, it can ensure correct results from earlier versions making it into the final draft. Consistency and reproducibility is also increased.

## 2 Including MATLAB code in LaTeX

A straightforward integration method is to include your MATLAB code in LaTeX so readers can read the MATLAB code with the MATLAB editor's style. There are many ways to achieve this, the simplest is to use the package matlab-prettifier. To include this package use the following in your LaTeX preamble,

```
\usepackage{matlab-prettifier}
```

you can now implement a lstlisting environment in LaTeX by implementing,

```
\begin{lstlisting}[<options>]
MATLAB code...
\end{lstlisting}
```

There are three style options that can be used,

1. Matlab-editor
2. Matlab-bw
3. Matlab-Pyglike

Before moving onto examples you might want to directly include the m-file without reproducing the code, this may be achieved with,

```
\lstinputlisting[style=Matlab-editor]{file-name.m}
```

where file-name is replaced with the path and file name of the m-file of interest.

### 2.1 Example 1 Option: Matlab - editor

The LaTeX required to achieve this is given by,

```
\documentclass[a4paper]{article}
\usepackage{matlab-prettifier}
\begin{document}

\begin{lstlisting}[style=Matlab-editor]
% Test m-file
for i=1:10
    fprintf('Print %d\n', i)
end
\end{lstlisting}
\end{document}
```

An example of a MATLAB code displayed in LaTeX

```
% Test m-file
for i=1:10
    fprintf('Print %d\n', i)
end
```

## 2.2 Example 2 Option: Matlab - bw

The LaTeX required to achieve this is given by,

```
\documentclass[a4paper]{article}
\usepackage{matlab-prettifier}
\begin{document}

\begin{lstlisting}[style=Matlab-bw]
% Test m-file
for i=1:10
    fprintf('Print %d\n', i)
end
\end{lstlisting}
\end{document}
```

An example of a MATLAB code displayed in LaTeX

```
% Test m-file
for i=1:10
    fprintf('Print %d\n', i)
end
```

## 2.3 Example 3 Option: Matlab - Pyglike

The LaTeX required to achieve this is given by,

```
\documentclass[a4paper]{article}
\usepackage{matlab-prettifier}
\begin{document}

\begin{lstlisting}[style=Matlab-bw]
% Test m-file
for i=1:10
    fprintf('Print %d\n', i)
end
\end{lstlisting}
\end{document}
```

An example of a MATLAB code displayed in LaTeX

```
% Test m-file
for i=1:10
    fprintf('Print %d\n', i)
end
```

## 2.4 Example 4 From a file

The LaTeX required to achieve this is given by,

```
\documentclass[a4paper]{article}
\usepackage{matlab-prettifier}
\begin{document}

\lstinputlisting[style=Matlab-editor]{test.m}

\end{document}
```

The MATLAB m-file is given in the same directory as the latex file that is being compiled and is named test.m the example of the LaTeX display is:

```
% Test m - file
for i =1:10
fprintf ( ' Print % d \ n ' , i )
end
```

# 3 Compiling LaTeX in MATLAB

A neater solution is to use MATLAB to generate the relevant LaTeX files and compile with pdflatex (or equivalent) to obtain a more sustainable approach. This approach has the advantage that output files can be constructed and then the LaTeX file updated every time you run your code.

## 3.1 pdflatex

To run pdflatex you will need to have downloaded and installed either MiKTeX or TeX Live. The distribution download pages for MiKTeX and TeX Live.

## 3.2 Creating LaTeX files from m-files

It is possible to generate a LaTeX file directly from MATLAB using the publish function. In general this is achieved from the following MATLAB m-file,

```
% Example MATLAB script for generating LaTeX documentation
    using publish

% Set the path to the directory containing your MATLAB m-file
    of interest
mFileName = 'mfile.m';
mFile = strcat('/path/to/your/',mFileName);

% Set the output directory for the LaTeX documentation
outputDirectory = '/path/to/output/directory';

% Specify the output LaTeX file name (without extension)
outputFileName = fullfile(outputDirectory, strrep(mFileName, '.
    m', ''))

% Use the publish function to generate LaTeX documentation
publish(mFile, 'format', 'latex', 'outputDir', outputDirectory)
    ;
```

### 3.2.1 Example 5

The code in the previous section requires you to input the relevant paths here is an example of specified paths on my computer.

```
% Example MATLAB script for generating LaTeX documentation
    using publish

% Set the path to the directory containing your MATLAB m-file
    of interest
mFileName = 'Example_script.m';

mFile = strcat('C:\Users\James\OneDrive\Desktop\MATLAB_Example\
    ',mFileName);

% Set the output directory for the LaTeX documentation
outputDirectory = '/path/to/output/directory';
```

```
outputDirectory = 'C:\Users\James\OneDrive\Desktop\
    MATLAB_Example\doc\'

% Specify the output LaTeX file name (without extension)
outputFileName = fullfile(outputDirectory, strrep(mFileName, '.
    m', ''))

% Use the publish function to generate LaTeX documentation
publish(mFile, 'format', 'latex', 'outputDir', outputDirectory)
    ;
```

this code produced the following LaTeX file,

```
% example_script.m
% This script demonstrates the use of add_numbers and multiply_numbers functions.

% Example usage of add_numbers
a = 5;
b = 3;
sum_result = add_numbers(a, b);
disp(['Sum: ' num2str(sum_result)]);

% Example usage of multiply_numbers
c = 4;
d = 2;
product_result = multiply_numbers(c, d);
disp(['Product: ' num2str(product_result)]);
       \color{lightgray} Sum: 8
Product: 8
```

Note you will need to modify the paths for your machine to achieve the same result.

## 3.3 Running LaTeX from MATLAB

Now we will focus on compiling LaTeX from MATLAB so you can create an entire workflow for updating documents without leaving MATLAB. Provided pdflatex is available on your machine you can call your system commands from within MATLAB using the function system.

### 3.3.1 Example 6: Compile LaTeX

In windows this can be achieved by (note it is assumed that example 5 has be run immediately prior to this code.

```
% Name of the output LaTeX file
latexFileName = fullfile(outputDirectory, strcat(mFileName((1:
    end-1)),'tex'))
latexCompileCommand = ['pdflatex -output-directory='
    outputDirectory ' ' latexFileName];

% Execute the LaTeX compile often it is best to run this twice
    or % 3 times
[status, result] = system(latexCompileCommand);
```

Now the pdf-file can be obtained from the previously published tex-file.

# 4 Output files

Often you will want to include generated text, tables and figures into your LaTeX documents

## 4.1 Text files

It is possible to use the following commands to include information within a text file or another tex-file. Note another tex-file should not contain the preamble and only contain the relevant latex

information. Overall, we can use input for inserting small files or when a page break is not required, while the include command is more suitable for large files.

```
\input{file-name}
```

and

```
\include{file-name}
```

You will need to provide the path to the file in place of file-name.

### 4.1.1 Example 7: Text

Include the following text file,

```
Text and tex files can be included. This is an example.
```

Using the input command the following is obtained:
Text and tex files can be included. This is an example.

## 4.2 Tables

Assuming you have a csv file (which you can create using fprintf commands in MATLAB, see Mathematical Workshop Spring and Numerical Methods and Programming for details or csvwrite). The following packages are required:

1. csvsimple - used to handle csv-files
2. booktabs - adds support for better table formatting

Furthermore, csvreader[] defines the parameters for reading the csv file. The remaining elements are standard table and tabular environment options. You can customise this code according to your specific needs.

### 4.2.1 Example 8: Tables

Tables can be created from csv files. The csv file contains the following,

```
Name,Age,Occupation
John,30,Engineer
Alice,25,Doctor
Bob,35,Teacher
```

and the following latex will create the table

```
\begin{table}
  \centering
  \caption{Example Table}
  \label{tab:example}
  \csvreader[
    head to column names,
    tabular=|l|c|r|,
    table head=\hline \textbf{Name} & \textbf{Age} & \textbf{Occupation} \\ \hline,
    late after line=\\ \hline,
    table foot=\hline,
  ]{csv.csv}{}
  {\Name & \Age & \Occupation}
\end{table}
```

This produces Table 1.

Table 1: Example Table

| Name | Age | Occupation |
|---|---|---|
| John | 30 | Engineer |
| Alice | 25 | Doctor |
| Bob | 35 | Teacher |

## 4.3 Figures

Generating figures in MATLAB has been discussed in both Mathematics Workshop Spring and Numerical Methods and Programming. This enables you to output your figures in different formats here is a good list of preferences for the output file:

1. eps - great format but can be difficult to send to other non-mathematics users
2. png - lossless format and strongly supported by Windows operating systems
3. jpg - only use if this is the format you have as the compression is lossy.

png is often a great compromise as the standard output from MATLAB figures. Since the saveas command has been discussed in year 1 and year 2 MATLAB modules we will jump to the example

### 4.3.1 Example 9: Figures

The matlab m-file,

```
% Your MATLAB script code to create a figure
x = linspace(0, 2*pi, 100);
y = sin(x);
figure;
plot(x, y);
xlabel('x-axis')
ylabel('y-axis')

% Save the figure
saveas(gcf, 'sine_wave_plot.png');  % You can specify the file format (e.g., 'sine_wave_plot.p
```

The output of this m-file is the png file, MATLAB uses the file ending to decide on the output format. The image can be seen in LaTeX by using the following latex commands,

```
\begin{figure}
    \centering
    \includegraphics{sine_wave_plot.png}
    \caption{Caption}
    \label{fig:MATLAB_example}
\end{figure}
```

The figure is then given in Figure 1

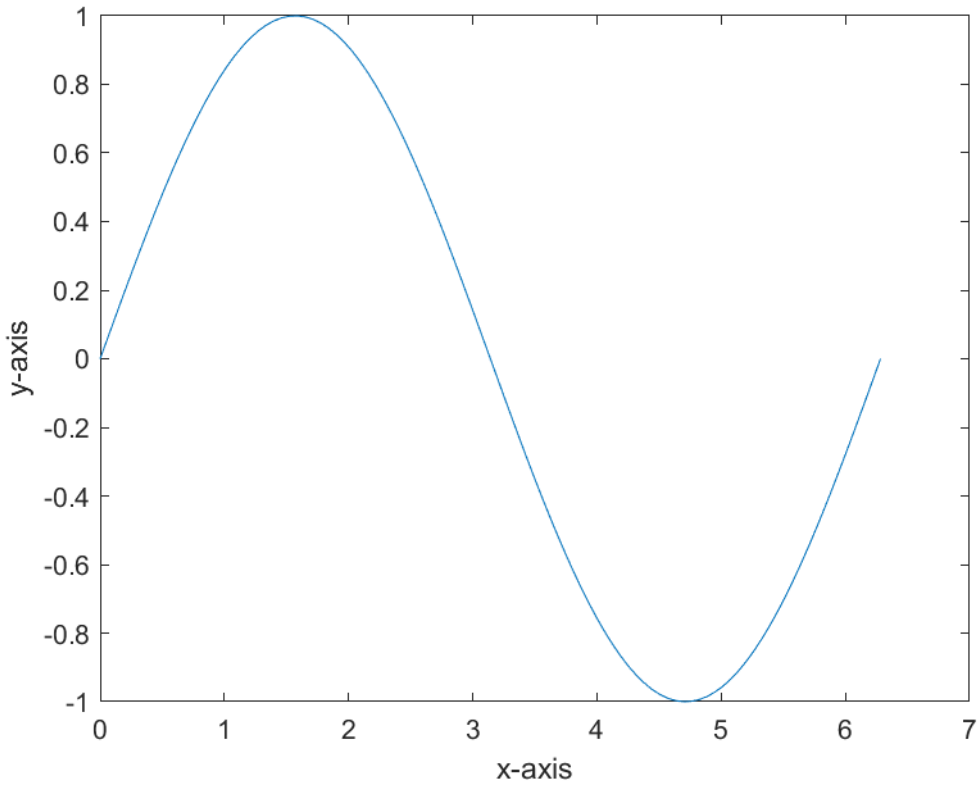


Figure 1: Caption