MatLaTex: Embedding Matlab Results in LaTeX Documents

version 2022.02.05

Athanasios Kehagias
kehagiat@ece.auth.gr

February 7, 2022

1 Introduction

`MatLaTex.m` is a *very simple* script intended to enable the user to *programmatically* include the results of Matlab computations in LaTeX documents; the focus is on *symbolic* computations but numerics and figures can also be used. Hence it is similar in purpose to the various "sweave" packages that exist for both Matlab and other languages, e.g. SageMath, R, Python etc. But the emphasis of MatLaTex is on *simplicity.* [1]

2 Installation

Simply unzip the file `MatLatex.zip` into some folder. You must have installed Matlab, the Matlab Symbolic Math Toolbox and a TeX distribution. I have tested MatLatex with Matlab R2018b and TeX Live 2019. It works on Windows 7, 10, 11; since it only depends on Matlab and LaTeX it should also work on the Linux and Apple operating systems (but I have not tested this).

3 QuickStart

In the Matlab environment, run the script `MatLatex.m`. You will get the files `MatLatex.tex` and `MatLatex.pdf` (this document). Open `MatLatex.m` to see the code which produces the results of this document (the contents of `MatLatex.m` will be explained in a later section).

4 Usage

The MatLaTex workflow is as follows.

1. Write a *single* Matlab script, e.g., `foo.m`, which contains both Matlab commands and LaTeX code (embedded as comments); the LaTeX code can include the control words `latex (a)` and `num2str(b)` where `a` and `b` are Matlab variables (details will be given in the following sections).
2. The file `foo.m` must be created in the same folder which contains `MatLatex.m`.
3. Change the second line of `MatLatex.m` from `fn='MatLatexDoc'` to `fn='foo'`.
4. Run `MatLatex.m` (i.e., type `MatLatex` in the Matlab command line).
5. When execution of `MatLatex.m` is completed you have the following files.
   (a) `foo.tex`: your LaTeX code with Matlab results having replaced the `latex (a)` and `num2str(b)` control words.

[1] Further discussion of this point appears in the Postscript.

(b) `foo.pdf`: the output of `foo.tex` as compiled by pdflatex.

To use MatLaTex follow the above workflow. The rules for writing MatLaTex files are as follows.

1. Each line contains either *only* Matlab code or *only* LaTeX code.
2. Matlab code is written as usual.
3. LaTeX code is also written as usual with the following exceptions.
   (a) Every line of LaTeX code is preceded by the characters `%%` (so, as far as Matlab is concerned, these are comment lines).
   (b) You can use the additional control word `latex ( )` (**without** being preceded by a backslash!). Every occurrence of `a` in the LaTeX part of your code will be replaced by the LaTeX expression for `a`, where it is assumed that `a` has been declared (in the Matlab part of your code) as a symbolic variable.
   (c) You can use the additional control word `num2str( )` (**without** being preceded by a backslash!). Every occurrence of `num2str(a)` in the LaTeX part of your code will be replaced by the `num2str` expression for `a`, where it is assumed that `a` has been declared / computed (in the Matlab part of your code) as a 1×1 double variable.

   Nota Bene: This means that you can only use `num2str ( )` to render *scalar* doubles. If you want to render a double matrix `A`, you must define a symbolic variable `Z` by `B=sym(A)` and then use `latex(Z)`. This works well when the entries of `A` are integers or simple fractions; but if `A` has entries with many decimals, `Z` will be represented by fractions with large integer numerators and denominators. So this is an issue which I hope to fix at a later version of MatLaTex.

5 Some Examples and Explanations

Let us now look at some parts of `MatLatexDoc.m`.

1. The file starts with the lines

```
%% \documentclass{article}
%% \usepackage{amsmath}
%% \usepackage{graphicx}
```

   and continues like this with typical LaTeX preamble commands. Note that, since these are LaTeX commands, they are preceded by `%%`.

2. After a while we have

```
%% \begin{document}
%% \maketitle
%% \section{Introduction}
%% \texttt{MatLaTex.m} is a \emph{very simple} script intended to
%% implement \emph{literate programming} in \textsf{Matlab}.
```

   and so on, where we write our LaTeX content as usual, but always using the `%%` line prefix.

3. Things get more interesting when we introduce symbolic computations. So for example the code

```
syms x
syms f(x)
f(x)=x*exp(x);
F(x)=int(f,x);
%% Let us write the integral of \(f(x)=latex (f)\),
%% i.e., \( \int latex (f) dx = latex (F)\).
```

produces the following results.

Let us write the integral of $f(x) = x\,\mathrm{e}^x$, i.e., $\int x\,\mathrm{e}^x dx = \mathrm{e}^x\ (x-1)$.

Similarly, the code

```
syms n
syms g(n)
g(n)=1/n^2;
G=symsum(g,n,1,inf);
G0=double(G);
%% Let us write the sum of \(g(n)=latex (g)\), i.e.,
%% \(\sum_{n=1}^\infty latex (g)  = latex (G)\).
%% This evaluates to \( latex (G) = num2str (G0)\).
```

produces the following results.

Let us write the sum of $g(n) = \frac{1}{n^2}$, i.e., $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. This evaluates to $\frac{\pi^2}{6} = 1.6449$.

4. We can also include plots. For example, the following code

```
figure(1); plot([0:0.1:4*pi],sin(2*[0:0.1:4*pi])); axis([0 4*pi -1.1 1.1]);
print('FIG001.pdf','-dpdf','-r600')
system(['pdfcrop␣FIG001.pdf␣FIG001.pdf']);
%% \begin{figure}[H]
%% \centering
%% \includegraphics[scale=0.75]{FIG001}
%% \caption{A plot of \(f(x)=\sin(2*x)\).}
%% \end{figure}
```

produces the plot:

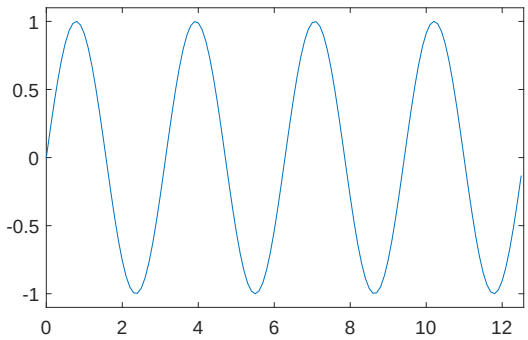

Figure 1: A plot of $f(x) = \sin(2 * x)$.

The idea is the following. The first three lines of the above fragment generate a "regular" Matlab plot, print it to file `FIG001.pdf` and then crop white space. The final five lines are regular LaTeX code which "graphics -includes" the previously produced file.

5. Let us now present some additional symbolic results; to see the code which generates the following lines open MatLatexDoc.m and look at around lines 180-260.

   (a) The Hessian of $f(x, y, z) = 2\,z\,x^2 + x\,y^2$ is

$$H = \begin{pmatrix} 4\,z & 2\,y & 4\,x \\ 2\,y & 2\,x & 0 \\ 4\,x & 0 & 0 \end{pmatrix}$$

(b) The Jacobian of $f(x, y, z) = 2\,z\,x^2 + x\,y^2$ is

$$J = \begin{pmatrix} y^2 + 4\,x\,z & 2\,x\,y & 2\,x^2 \end{pmatrix}$$

(c) An algebraic simplification

$$\frac{x^3 - y^3}{x - y} = x^2 + x\,y + y^2$$

(d) Let us solve the equation $x^2 + x + 1 = 0$. It has the roots

$$r_1 = -\frac{1}{2} - \frac{\sqrt{3}\,1\mathrm{i}}{2}, \qquad r_2 = -\frac{1}{2} + \frac{\sqrt{3}\,1\mathrm{i}}{2}.$$

(e) Let us solve the differential equation $\frac{\partial}{\partial t} u\,(t) = t\,u\,(t)$. It has the family of solutions

$$u(t) = C_1\,\mathrm{e}^{\frac{t^2}{2}}.$$

(f) We can write a matrix and compute its inverse

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad \text{and} \qquad A^{-1} = \begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$$

(g) The same thing with a matrix which includes symbols

$$B = \begin{pmatrix} 1 & 2 \\ a & b \end{pmatrix} \qquad \text{and} \qquad B^{-1} = \begin{pmatrix} -\frac{b}{2\,a-b} & \frac{2}{2\,a-b} \\ \frac{a}{2\,a-b} & -\frac{1}{2\,a-b} \end{pmatrix}$$

(h) Here is a Fourier transform.

$$\mathcal{F}(a\,\cos\left(t\,w_0\right)) = \pi\,a\,\left(\delta\left(t - w\right) + \delta\left(t + w\right)\right).$$

(i) Here is a Laplace transform.

$$\mathcal{L}(a\,\cos\left(t\,w_0\right)) = \frac{a\,s}{s^2 + {w_0}^2}.$$

Here are two additional plots (look up the corresponding code in `MatLatexDoc.m`).

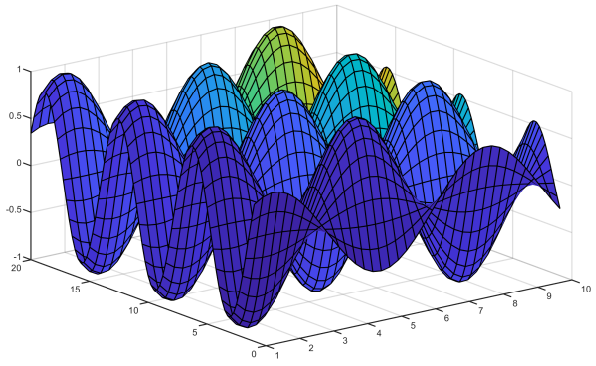

Figure 2: A plot of $f(x, y) = \sin(x)\cos(y)$.

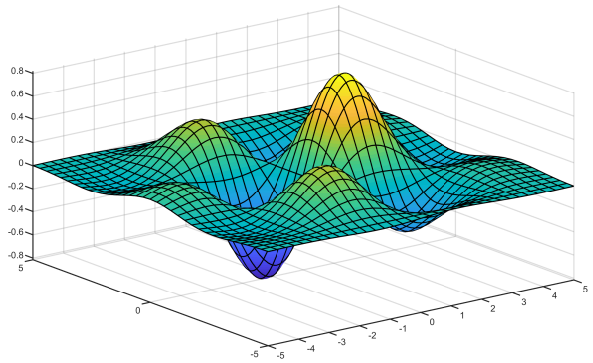

Figure 3: A plot of $f(x,y) = \sin(x)\cos(y)e^{-\frac{x^2+y^2}{10}}$.

6 Postscript

My motivation for writing MatLaTex was the need for a simple system to create problem sets and exams (and answers!) for my math classes. I have been using several computer algebra systems: the symbolic math toolboxes of Matlab and Octave, Maple, SageMath, SymPy and so on. Not wanting to reinvent the wheel I looked at what was available. I started with SageTex; alas, I was never able to properly install and run it. I tried several additional packages and I found each one either too confusing to set up, or not providing the functionalities I wanted, or both.

Consequently I decided to write my own package. I am a simple guy and MatLaTex is a simple hack. I am certain that any sufficiently interested serious programmer can produce a much better version and I will be very happy if someone does. In the meantime, MatLaTex works right out of the box and does what I want it to do: programmatically incorporate symbolic and numeric Matlab results into my LaTeX documents.

In conclusion, there are a few improvements / extensions on which I hope to work in the future. I list them in order of decreasing priority.

1. In the current implementation, variable names cannot be "reused". If a variable `a` appears several times in the LaTeX commands, it will always be replaced by its last computed value. I would like to be able to replace each occurrence of `a` with its value as computed just before this appearance.

2. Double matrices should be handled better.

Finally, let me mention that I am also working on OctLatex and MapleLatex which are Octave and Maple versions of the MatLatex idea.