# Exporting vector graphics in Matlab with correct fonts

Juan M. Guerrero, guerrero@uniovi.es

April 25, 2017

## 1 Prerequisites

This tutorial is intended for versions 2015b and 2016b of Matlab, although most ideas can be easily extended to other versions.

It is assumed some external software is installed:

- *epstopdf* that can be found in some LaTex distributions as *MikTex*.
- *ghostscript* that is called by *epstopdf*. Also included in *MikTex*.
- Not required, but recommended to easily see the pdf results, Sumatra PDF.

## 2 Problem description

If you are reading this tutorial, probably you are more than aware of the problem when exporting vector graphics with the Matlab `print` command. If not, the following describes the problem.

Let's create a Matlab 18x10 cm figure to write some text and labels inside:

Listing 1: FontExample1.m: Create a figure with proper dimensions

```
% 1 - Create a 18x10 cm figure to illustrate the example
close all % Optional, in case you want to get rid of all previous figures
fig = figure;
fig.Units = 'centimeters';
width = 18; % cm
height = 10; % cm
fig.Position = [fig.Position(1) fig.Position(2) width height]; % 10x10 cm
ax=axes; % Create an axis for the font example
```

Now, we write some text and labels in the figure with different fonts. Additionally, we change the tick label fonts to "Elephant". I have randomly selected fonts installed in my Windows system. Replace them for your choice of installed fonts if you want. The output figure can be seen in Fig.1.

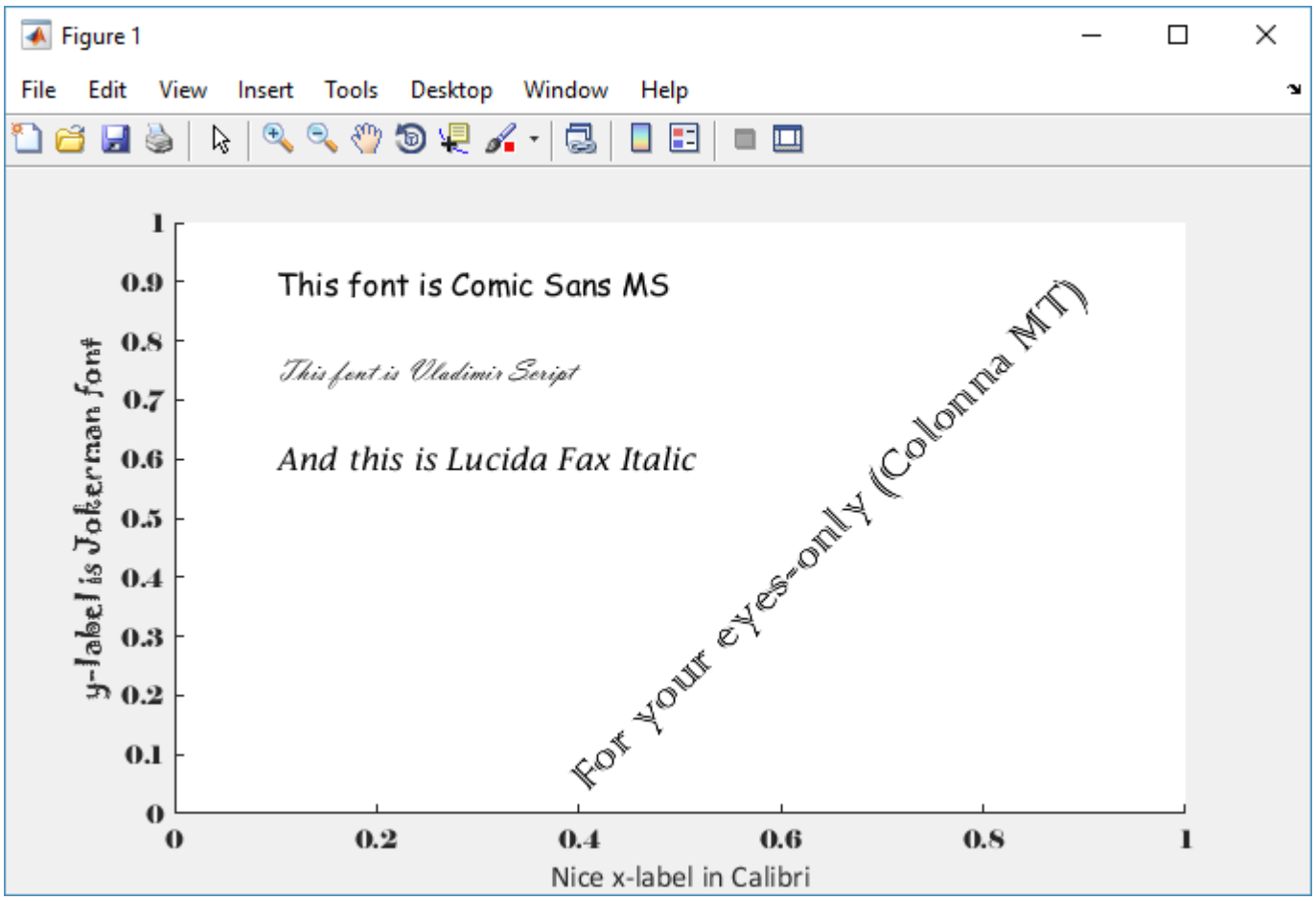


Figure 1: Figure obtained from FontExample1.m

Listing 2: FontExample1.m: Writing text and labels

```
% 2- Write some text and labels with different fonts(beware: select fonts installed
% in your system)
ax.FontName = 'Elephant';
text(.1,.9,'This font is Comic Sans MS','fontname','Comic Sans MS','fontsize',12);
text(.1,.75,'This font is Vladimir Script','fontname','Vladimir Script','fontsize',12);
text(.1,.6,'And this is Lucida Fax Italic','fontname','Lucida Fax','fontangle','italic','fontsize',12);
text(.4,.05,'For your eyes-only (Colonna MT)','fontname','Colonna MT','fontsize',20,'rotation',45);
xl = xlabel('Nice x-label in Calibri','fontname','Calibri','fontsize',12);
yl = ylabel('y-label is Jokerman font','fontname','Jokerman','fontsize',12);
```

And now, we want to obtain a *pdf* file from this figure. To do so, I first export the figure to *eps*, and then I use *epstopdf* to obtain the *pdf* file. The last step opens the pdf.

Listing 3: FontExample1.m: Obtaining the *pdf* in two steps

```
% 3- Obtain the eps file using print
file_name = 'Example1';
set(fig,'paperpositionmode','auto','units','centimeters','paperunits','centimeters','papersize',[width ...
    height]);
print(fig,'-depsc','-painters','-r600','-loose',strcat('.\',file_name));   % eps

%4- Obtain the pdf file using epstopdf (included for instance in MikTex)
system(sprintf('epstopdf --gsopt="-sFONTPATH=C:/Windows/Fonts  -dSubsetFonts=true  ...
    -dEmbedAllFonts=true -dPDFSETTINGS=/prepress" "%s.eps"',strcat('.\',file_name)));

%5- Open the pdf with the viewer of your choice (Sumatra in this case). Or remove this and open it ...
    manually
system(sprintf('"C:/Program Files/SumatraPDF/SumatraPDF.exe" -reuse-instance ...
    %s',strcat('.\',file_name,'.pdf &')));
```

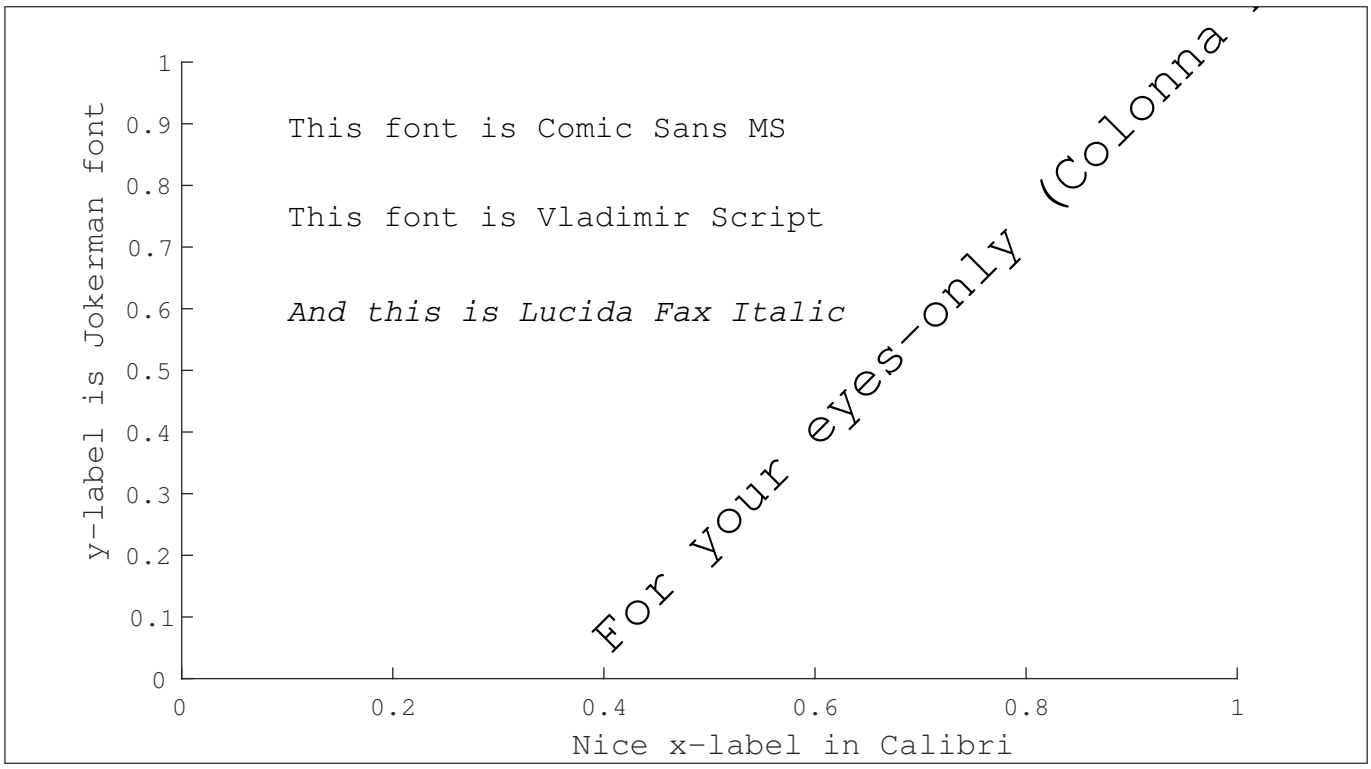


Figure 2: *Pdf* obtained from FontExample1.m

Fig. 2 shows the resulting pdf. As you can see the fonts have been replaced This is also the case if you export directly to pdf with the `'-dpdf'` option of `print`.

The problem is `print` is only able to handle the following fonts in Matlab 2015-16b, totalling 12 different fonts:

- **Helvetica Family**: Helvetica, Helvetica-Bold, Helvetica-Oblique, Helvetica-BoldOblique
- **Courier Family**: Courier, Courier-Bold, Courier-Oblique, Courier-BoldOblique
- **Times-Roman family**: Times-Roman, Times-BoldItalic, Times-Bold
- ZapfDingbats

Previous versions of Matlab were able to process some extra fonts, but now this is all. When a font in the figure does not corresponds to any in that subset, it is remapped in the *eps* file to one of that list, with the results shown in Fig. 2.

# 3 Solution

To solve for this the following strategy can be used:

1. Before converting to *eps* we are going to remap our figure fonts to the Matlab `print` built-in fonts. This means that we only can use 12 different fonts, including the font variations (italic, bold). First we have to obtain handles to the text objects and then we make the replacement as can be seen in listing 4. To access the bold, italic or bold-italic variations of the fonts we have to use the `fontangle` and `fontweight` text-object options, as can be seen in listing 4. In case of having `fontangle` and `fontweight` variations of the same font in the same figure, it is important to explicitily specify them in the `findall` command in **all** cases, as seen only for `font4` in this example.

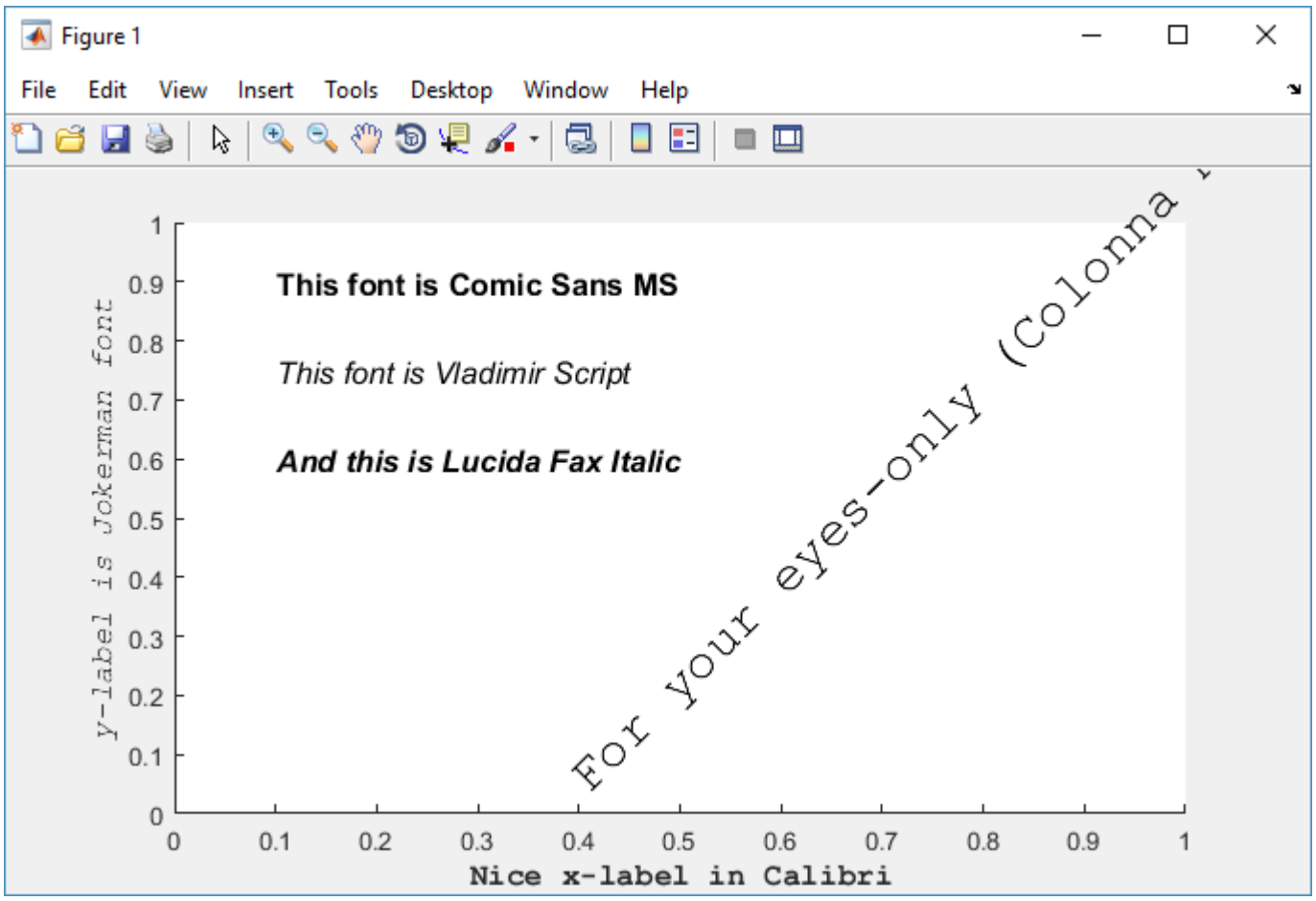


Figure 3: Figure obtained from FontExample2.m after font substitution

Listing 4: FontExample2.m: Font remapping

```
% 3- Preprocess. Remap the fonts to the Matlab subset
% Find font objects
font1 = findall(gcf,'fontname','Elephant');
font2 = findall(gcf,'fontname','Comic Sans MS');
font3 = findall(gcf,'fontname','Vladimir Script');
font4 = findall(gcf,'fontname','Lucida Fax','fontangle','italic');
font5 = findall(gcf,'fontname','Colonna MT');
font6 = findall(gcf,'fontname','Calibri');
font7 = findall(gcf,'fontname','Jokerman');

% Replace fonts
% Important Note: axes font must be remapped BEFORE the xlabel and ylabel
% remap (1, before 6 and 7 in this example)
set(font1,'fontname','Helvetica'); % Remap to Helvetica
set(font2,'fontname','Helvetica','fontweight','bold'); % Remap to Helvetica-Bold
set(font3,'fontname','Helvetica','fontangle','italic'); %  Remap to Helvetica-Oblique
set(font4,'fontname','Helvetica','fontangle','italic','fontweight','bold'); %  Remap to ...
    Helvetica-BoldOblique
set(font5,'fontname','courier'); %  Remap to Courier
set(font6,'fontname','courier','fontweight','bold'); %  Remap to Courier-Bold
set(font7,'fontname','courier','fontangle','italic'); %  Remap to Courier-Oblique
```

After this change, the Matlab figure seen in Fig. 1 becomes as shown in Fig. 3. The fonts have been temporary changed to some of the Matlab `print` built-in fonts.

2. Then we obtain the *eps* file as shown in step 3 of listing 3.

3. We can now recover (optional) the original fonts to leave the Matlab figure at it should be (Fig. 1). It is important to force the `fontangle` and `fontweight` to the original values provided they might have changed in the previous font substitution, as seen in listing 5.

Listing 5: FontExample2.m: Restoring the original fonts

```
% 5- Postprocess (recover the fonts in figure)
set(font1,'fontname','Elephant','fontangle','normal','fontweight','light');
set(font2,'fontname','Comic Sans MS','fontangle','normal','fontweight','light');
set(font3,'fontname','Vladimir Script','fontangle','normal','fontweight','light');
set(font4,'fontname','Lucida Fax','fontangle','italic','fontweight','light');
set(font5,'fontname','Colonna MT','fontangle','normal','fontweight','light');
set(font6,'fontname','Calibri','fontangle','normal','fontweight','light');
set(font7,'fontname','Jokerman','fontangle','normal','fontweight','light');
```

4. Finally, we obtain the *pdf* with *epstopdf*. The difference now is we are going to use a custom fontmap (*FontExample2.gs*). Important note: change in FontExample2.m (and in following examples) the path to the fontmap file (K:/Documentos archivados/M Files/Megaplot/) to the corresponding to your case.

Listing 6: FontExample2.m: Producing the *pdf* using a custom fontmap

```
%6- Obtain the pdf file using epstopdf (included for instance in MikTex)
system(sprintf('epstopdf --gsopt="-sFONTMAP=""K:/Documentos archivados/M ...
    Files/Megaplot/FontExample2.gs""  -sFONTPATH=C:/Windows/Fonts  -dSubsetFonts=true  ...
    -dEmbedAllFonts=true -dPDFSETTINGS=/prepress" "%s.eps"',strcat('.\',file_name)));


%7- Open the pdf with the viewer of your choice (Sumatra in this case). Or remove this and ...
    open it manually
system(sprintf('"C:/Program Files/SumatraPDF/SumatraPDF.exe" -reuse-instance ...
    %s',strcat('.\',file_name,'.pdf &')));
```

The custom fontmap contains the path to the fonts used in the *eps* file. We are going to mislead epstopdf (ghostscript) to replace the fonts in the *eps* for the fonts in our matlab script.

Listing 7: FontExample2.gs

```
% FontExample2.gs
% Ghostscript font map
%
% Author: Juan M. Guerrero
% Institution: University of Oviedo
% Date: 04/21/2017

% Helvetica Family
%/Helvetica                      (C:/WINDOWS/Fonts/arial.ttf)    ;   % Closest match in Windows
%/Helvetica-Bold                 (C:/WINDOWS/Fonts/arialbd.ttf)  ;   % Closest match in Windows
%/Helvetica-Oblique              (C:/WINDOWS/Fonts/ariali.ttf)   ;   % Closest match in Windows
%/Helvetica-BoldOblique          (C:/WINDOWS/Fonts/arialbi.ttf)  ;   % Closest match in Windows

/Helvetica                  (C:/WINDOWS/Fonts/ELEPHNT.TTF)    ; % Map to Elephant
/Helvetica-Bold             (C:/WINDOWS/Fonts/comic.ttf)      ; % Map to Comic Sans MS
/Helvetica-Oblique          (C:/WINDOWS/Fonts/VLADIMIR.TTF)   ; % Map to Vladimir Script
/Helvetica-BoldOblique      (C:/WINDOWS/Fonts/LFAXI.TTF)      ; % Map to Lucida Fax (Italic)

% Courier Family
%/Courier                     (C:/WINDOWS/Fonts/cour.ttf)    ; % Closest match in Windows
%/Courier-Bold                (C:/WINDOWS/Fonts/courbd.ttf)  ; % Closest match in Windows
%/Courier-Oblique             (C:/WINDOWS/Fonts/couri.ttf)   ; % Closest match in Windows
/Courier-BoldOblique        (C:/WINDOWS/Fonts/courbi.ttf)   ; % Closest match in Windows

/Courier                    (C:/WINDOWS/Fonts/COLONNA.TTF)   ; % Map to Colonna MT
/Courier-Bold               (C:/WINDOWS/Fonts/calibri.ttf)   ; % Map to Calibri
/Courier-Oblique            (C:/WINDOWS/Fonts/JOKERMAN.TTF)  ; % Map to Jokerman

% Times-Roman Family
/Times-Roman                (C:/WINDOWS/Fonts/times.ttf)    ; % Closest match in Windows
/Times-Bold                 (C:/WINDOWS/Fonts/timesbd.ttf)  ;   % Closest match in Windows
/Times-BoldItalic           (C:/WINDOWS/Fonts/timesbi.ttf)  ;   % Closest match in Windows

% ZapfDingbats
/ZapfDingbats               (C:/WINDOWS/Fonts/wingding.ttf)  ; % There is no match in my system ...
    (dummy map to Windows)
```

The file name of the fonts in the system can be found with the script *SearchFonts.m* by writing the beginning of the font name in the Matlab console. Alternatively, you can search for it by browsing in your file system.

Listing 8: SearchFont.m

```
% SearchFont.m
% Searches for font file names to include in fontmap.gs
%
% Author: Juan M. Guerrero
% Institution: Universidad de Oviedo (Spain)
% Date: 04/21/2017

clc;
fuente = input('Write the beginning of the name of the font (case sensitive) [e.g. Times, ...
    Lucida...] : ','s');
subs_char = '-';
espacios = 10;

system_fonts = fontinfo;

font_match = strfind(system_fonts,fuente);


index = false(1,numel(font_match));

for k = 1:numel(font_match)
    index(k) = ~isempty(font_match{k});
end

index = find(index ~=0);

if (isempty(index))
    disp('The font has not been found in your system');
    return;
end

fuentes = system_fonts(index);
fuentes_path = fuentes;

for k = 1:length(fuentes);
    fuentes_path{k} = fontinfo(char(fuentes{k}));

    % Create list for including in Ghostscript fontmap.gs
    index = strfind(fuentes{k},' ');
    fuentes{k}(index) = subs_char;

    % Change slash in Windows
    index = strfind(fuentes_path{k},'\');
    fuentes_path{k}(index) = '/';

    % Create line for fontmap.gs
    fuentes_path{k} =[blanks(espacios) char(['(' fuentes_path{k} ')    ;'])];
    fuentes{k} = ['/' fuentes{k}];
end

% Text to paste
[char(fuentes), char(fuentes_path)]
```

An usage example can be seen in Fig. 4. You have to copy and paste the path to the desired font (Lucida Fax Italic) in the substitution font, `/Helvetica-BoldOblique`, in the fontmap.gs (FontExample2.gs in this case). Note: *cursiva* stands for *italic* in Spanish language.

The resulting *pdf* file can be seen in Fig. 5. If you compare with the original Matlab figure in Fig. 1 is almost identical, at least it contains the correct fonts.

However, there are small differences. The x-label and y-label have been slightly displaced. This is due to the font replacements for the original label fonts occupy more space, as can be seen by comparison of x- and y-labels of Figs. 1 and 3. The *eps* file stores the starting position of the string so the labels are finally displaced when the *pdf* is produced.

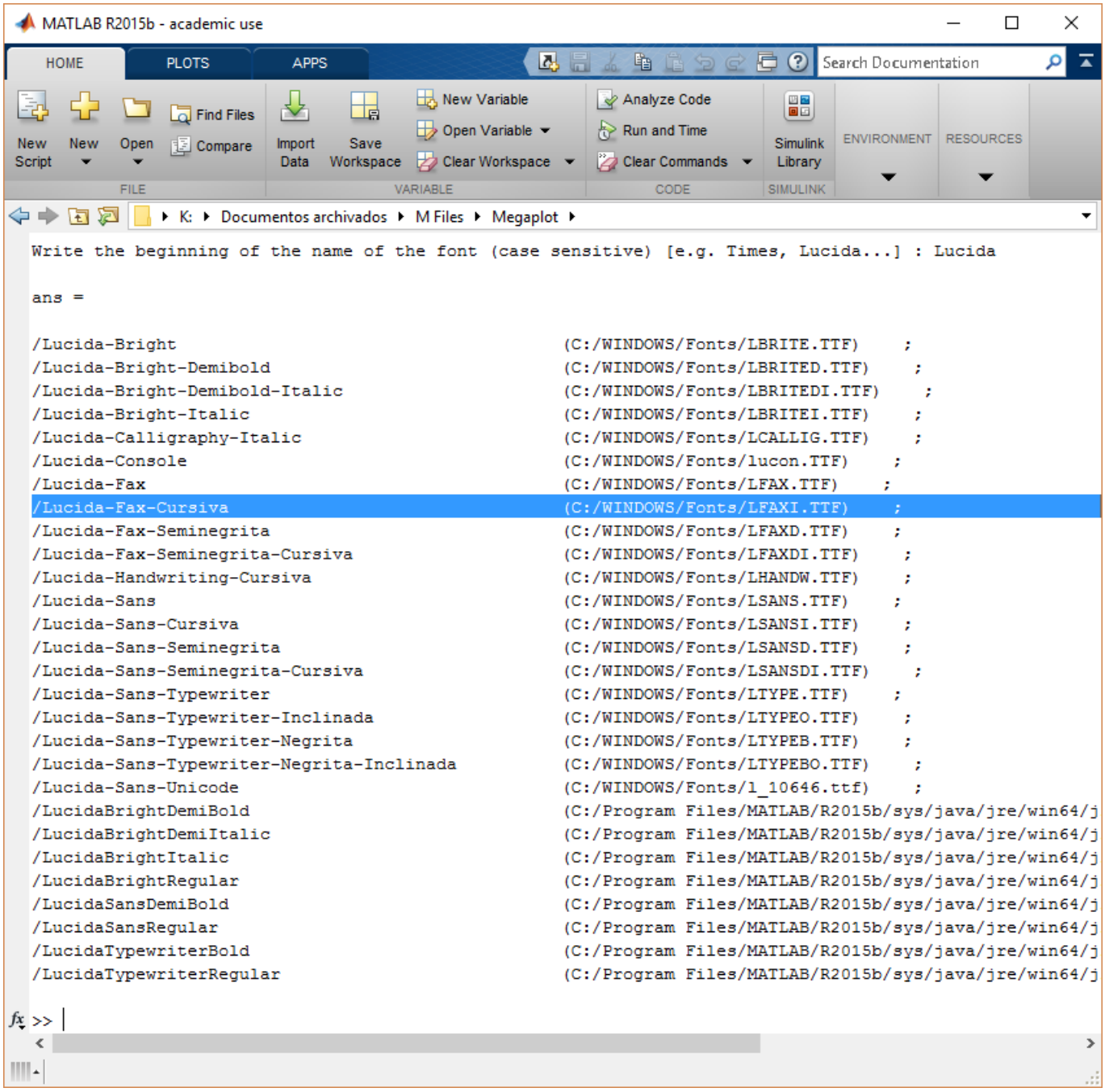


Figure 4: SearchFont.m usage example.

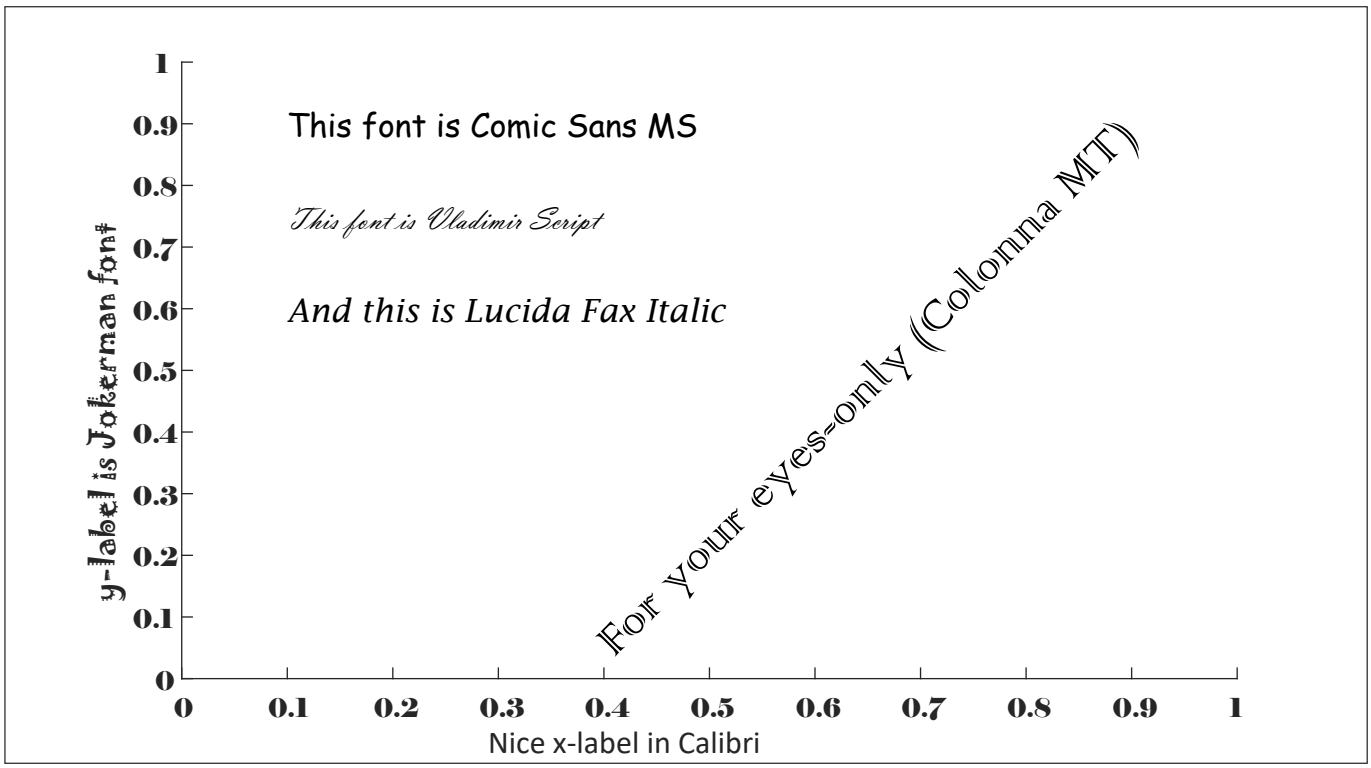


Figure 5: *Pdf* obtained from FontExample2.m

Another difference is the different tick labeling step due to the automatic tick space selection based on the font and axes dimensions, that changes during the font replacement. This can be easily avoided, if needed, by forcing the desired tick labels with `XTick` and `YTick` after the replacement.

To solve for the first problem we are going to add some code lines. First, before replacing the fonts in the figure, we store the label positions.

Listing 9: FontExample3.m: Storing the label positions

```
% 3- Calculate the original label position
xl.Units = 'pixels';
xl.Margin = 1;
xl_hpos = xl.Extent(1)+1;
yl.Units = 'pixels';
yl.Margin = 1;
yl_vpos = yl.Extent(2)+1;
```

And right before the *eps* to *pdf* conversion the postion is restored. I also force the tick labels to correspond with those of Fig.1.

Listing 10: FontExample3.m: Restoring the label positions

```
% Restore label left positions
xl.HorizontalAlignment = 'left';
xl.Position= [xl_hpos xl.Position(2) xl.Position(3)];

yl.HorizontalAlignment = 'left';
yl.Position= [yl.Position(1) yl_vpos yl.Position(3)];

ax.XTick=[0:.2:1]; % to force a particular tick labeling
```

This produce the resulting *pdf* seen in Fig. 6. Now, the axis labels are in their original positions (Fig. 1).

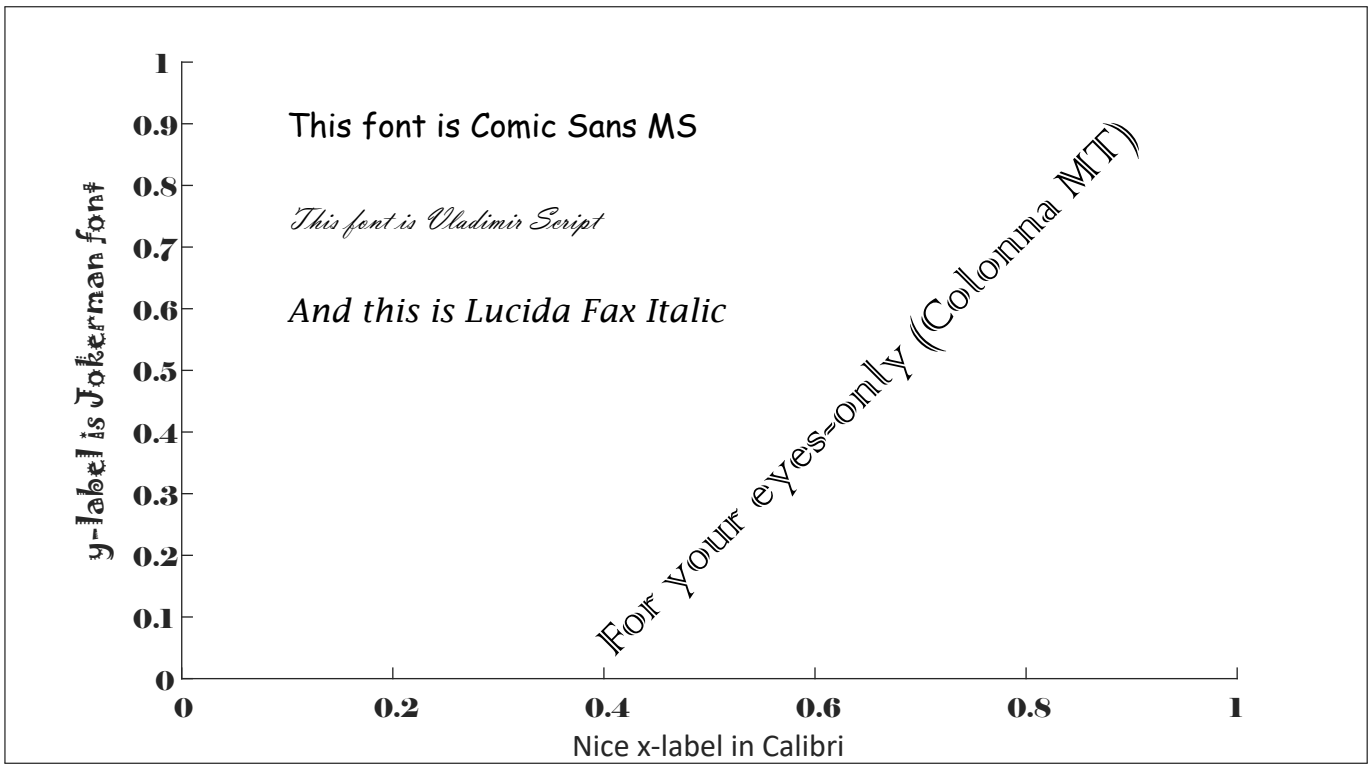


Figure 6: *Pdf* obtained from FontExample3.m

# 4 Labels using the LaTeXinterpreter

If you write technical documents or papers, you will often need to use the LaTeXinterpreter to write your labels and text comments in the Matlab plots. The conversion from figure to *pdf* when the LaTeXinterpreter is used does not require special processing. The following code produces an example graphic summarizing the different LaTeXfont names in Matlab 2015b.

Listing 11: FontExample4.m: Drawing LaTeXlabels and text

```
% FontExample4.m
% This example shows the text fonts of the LaTeX intrepreter in Matlab
% 2015b
%
% Author: Juan M. Guerrero
% Institution: Universidad de Oviedo (Spain)
% Date: 04/21/2017


% 1 - Create a 18x10 cm figure to illustrate the example
close all % Optional, in case you want to get rid of all previous figures
fig = figure;
fig.Units = 'centimeters';
width = 18; % cm
height = 10; % cm
fig.Position = [fig.Position(1) fig.Position(2) width height]; % 10x10 cm
ax=axes; % Create an axis for the font example


% 2- Write some text and labels with different fonts(beware: select fonts installed
% in your system)
ax.FontName = 'Times New Roman';
text(.05,1,'\textnormal{This is textnormal} (mwa\_cmr10)','interpreter','latex','fontsize',12);
text(.05,.9,'\textrm{This is textrm (mwa\_cmr10)}','interpreter','latex','fontsize',12);
text(.05,.8,'\textit{This is textit (mwa\_cmti10)}','interpreter','latex','fontsize',12);
text(.05,.7,'\textbf{This is textbf (mwa\_cmbx10)}','interpreter','latex','fontsize',12);
text(.05,.6,'\texttt{This is textt (mwa\_cmtt10)}','interpreter','latex','fontsize',12);
text(.05,.5,'\textsl{This is textsl (mwa\_cmsl10)}','interpreter','latex','fontsize',12);

```

```
text(.6,1,'$\mathnormal{This¬is¬mathnormal (mwa\_cmmi10)}$','interpreter','latex','fontsize',12);
text(.65,.9,'$\mathrm{This¬is¬mathrm (mwa\_cmr10)}$','interpreter','latex','fontsize',12);
text(.65,.8,'$\mathit{This¬is¬mathit (mwa\_cmti10)}$','interpreter','latex','fontsize',12);
text(.65,.7,'$\mathbf{This¬is¬mathbf (mwa\_cmbx10)}$','interpreter','latex','fontsize',12);
text(.65,.6,'$\mathtt{This¬is¬mathtt (mwa\_cmtt10)}$','interpreter','latex','fontsize',12);
text(.65,.5,'$\mathcal{This¬is¬mathcal (mwa\_cmsy10)}$','interpreter','latex','fontsize',12);
text(.65,.4,'Above: mathcal (mwa\_cmsy10)','interpreter','latex','fontsize',12);

text(.05,.2,'This mixes text and math: $ G(s)  =\frac{s^2 + 2 s +2}{s^3 + 2s +1}$ \bf and text ...
    again','interpreter','latex','fontsize',12);

xl = xlabel('Nice x-label in LaTeX $\lambda = \frac{\infty}{0}$','interpreter','latex','fontsize',12);
yl = ylabel('y-label $\mathbf{A} \frac{\mathbf{\alpha}}{\sum ...
    \gamma}$','interpreter','latex','fontsize',12);


% 4- Preprocess. Remap the fonts to the Matlab subset
% Find font objects
font1 = findall(gcf,'fontname','Times New Roman');

% Replace fonts
set(font1,'fontname','Times-Roman'); % Remap to Times-Roman


% 5- Obtain the eps file using print
file_name = 'Example4';
set(fig,'paperpositionmode','auto','units','centimeters','paperunits','centimeters','papersize',[width ...
    height]);
print(fig,'-depsc','-painters','-r600','-loose',strcat('.\',file_name));   % eps


% 6- Postprocess (recover the fonts in figure)
set(font1,'fontname','Times New Roman','fontangle','normal','fontweight','light');


% 7- Obtain the pdf file using epstopdf (included for instance in MikTex)
system(sprintf('epstopdf ––gsopt="-sFONTMAP=""K:/Documentos archivados/M ...
    Files/Megaplot/FontExample4.gs""  -sFONTPATH=C:/Windows/Fonts  -dSubsetFonts=true  ...
    -dEmbedAllFonts=true -dPDFSETTINGS=/prepress" "%s.eps"',strcat('.\',file_name))) %keeps colormap


% 8- Open the pdf with the viewer of your choice (Sumatra in this case). Or remove this and open it ...
    manually
system(sprintf('"C:/Program Files/SumatraPDF/SumatraPDF.exe" -reuse-instance ...
    %s',strcat('.\',file_name,'.pdf &')));
```

The resulting Matlab figure can be seen in Fig. 7 and the obtained *pdf* file in Fig. 8. The LaTeXtext font names defined in the *eps* file after printing the figure with Matlab 2015b are:

- mwa_cmr10
- mwa_cmti10
- mwa_cmbx10
- mwa_cmtt10
- mwa_cmsl10
- mwa_cmsy10

There are also additional fonts defined, starting mwb_ intended for greek letters and special signs. The Matlab LaTeXfonts are fully encoded in the *eps* file in Matlab 2015b and then used as another regular font. However, in Matlab 2016b each character is defined when it is used and the font set is not defined. Therefore, it cannot be modified.

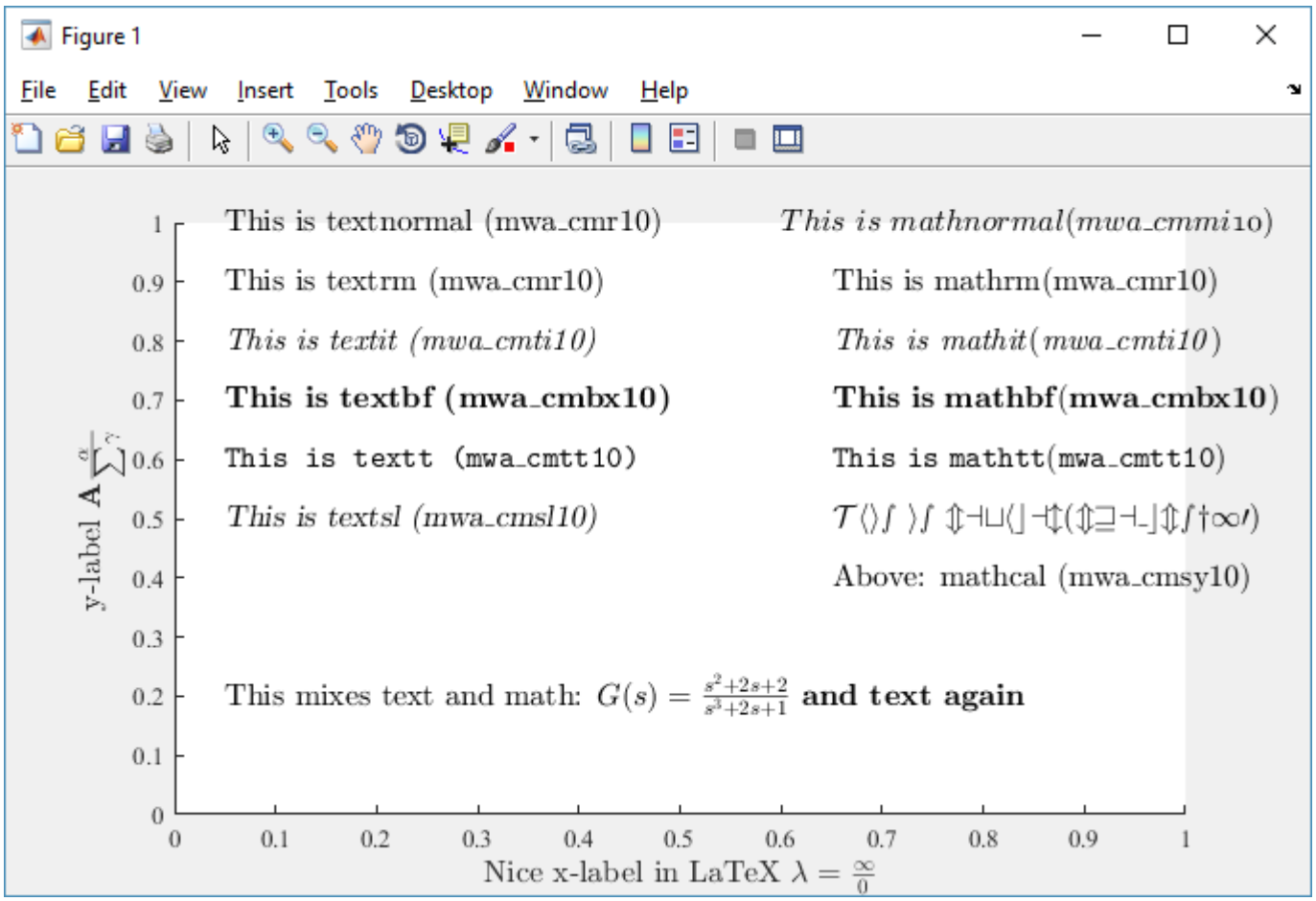


Figure 7: Matlab figure from FontExample4.m

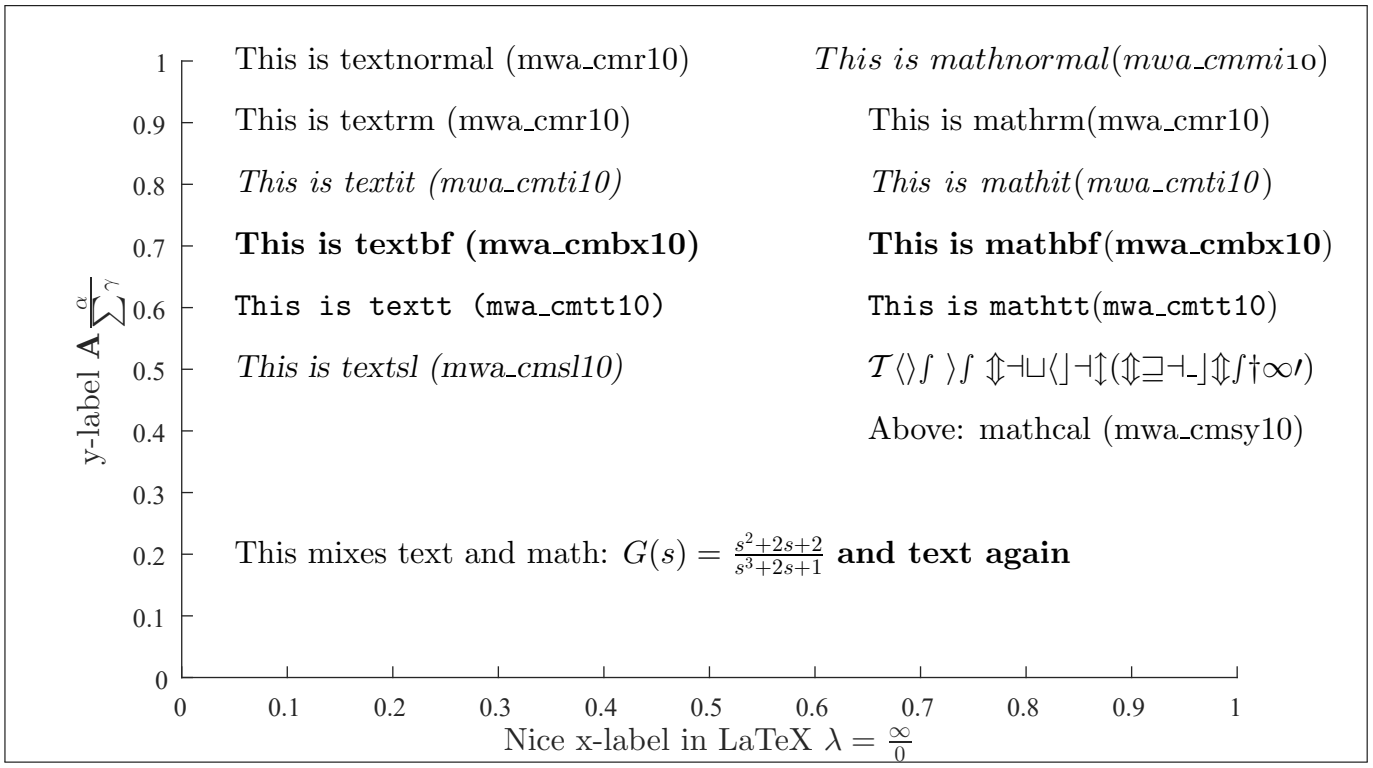


Figure 8: *Pdf* obtained from FontExample4.m

# 5 Modifying the LaTeXfonts (only Matlab 2015b)

The former discussion uses the default Tex interpreter for labels and text. Sometimes you need to produce a label containing complex math equations or special symbols easily obtained in LaTeX, but the journal you want to send your article for publication, or your thesis advisor, requires some particular font for the regular text and labels (e.g. Arial or Times New Roman). How can you change the LaTeXfonts?

The fonts cannot be changed in the Matlab figure, provided the fonts are internally defined when the LaTeXinterpreter is activated. However, we can still change the font in the exported *pdf* in Matlab 2015b.

As an example, we are going to replace the font mwa_cmr10 seen in Fig. 8 by the font Jokerman. The following code is added after the *eps* file generation, as seen in code listing 11. We read the *eps* file. Then, we replace all the occurrences, but the first, of `/mwa_cmr10` by one of the Matlab `print` fonts, in this case Helvetica. And then we save the updated *eps* file. In the custom fontmap, Helvetica is redefined to Jokerman.

Listing 12: FontExample5.m: Replacing LaTeXfonts

```
% 7- Search and replace latex fonts for Matlab fonts
fid = fopen(strcat('.\',file_name,'.eps'));
ff = char(fread(fid))';  % A large string with the file contents
fclose(fid);
%
index = min(strfind(ff,'/mwa_cmr10')); % Serach for the font you want to replace
ff  = strcat(ff(1:index+9),strrep(ff(index+10:end),'/mwa_cmr10','/Helvetica')); % Substitute for ...
    one of the Matlab fonts

% % open the file up and overwrite it
fid = fopen(strcat('.\',file_name,'.eps'),'w');
fprintf(fid,'%s',ff);
fclose(fid);
```

Listing 13: FontExample5.gs: Redefining Helvetica in the fontmap file

```
% FontExample2.gs
% Ghostscript font map
%
% Author: Juan M. Guerrero
% Institution: University of Oviedo
% Date: 04/21/2017

% Helvetica Family
%/Helvetica                    (C:/WINDOWS/Fonts/arial.ttf)    ;   % Closest match in Windows
/Helvetica-Bold             (C:/WINDOWS/Fonts/arialbd.ttf)    ; % Closest match in Windows
/Helvetica-Oblique          (C:/WINDOWS/Fonts/ariali.ttf)    ;  % Closest match in Windows
/Helvetica-BoldOblique      (C:/WINDOWS/Fonts/arialbi.ttf)    ; % Closest match in Windows

/Helvetica                  (C:/WINDOWS/Fonts/JOKERMAN.TTF)    ;   % Remap

% Courier Family
/Courier                    (C:/WINDOWS/Fonts/cour.ttf)    ;   % Closest match in Windows
/Courier-Bold               (C:/WINDOWS/Fonts/courbd.ttf)    ; % Closest match in Windows
/Courier-Oblique            (C:/WINDOWS/Fonts/couri.ttf)     ; % Closest match in Windows
/Courier-BoldOblique        (C:/WINDOWS/Fonts/courbi.ttf)    ; % Closest match in Windows

% Times-Roman Family
/Times-Roman                (C:/WINDOWS/Fonts/times.ttf)    ;   % Closest match in Windows
/Times-Bold                 (C:/WINDOWS/Fonts/timesbd.ttf)    ; % Closest match in Windows
/Times-BoldItalic           (C:/WINDOWS/Fonts/timesbi.ttf)    ; % Closest match in Windows

% ZapfDingbats
/ZapfDingbats               (C:/WINDOWS/Fonts/wingdings.TTF)    ; % There is no match in my system ...
    (dummy map to Wingdings)
```

The resulting *pdf* file can be seen in Fig. 9. Although this method works, it presents some limitations:

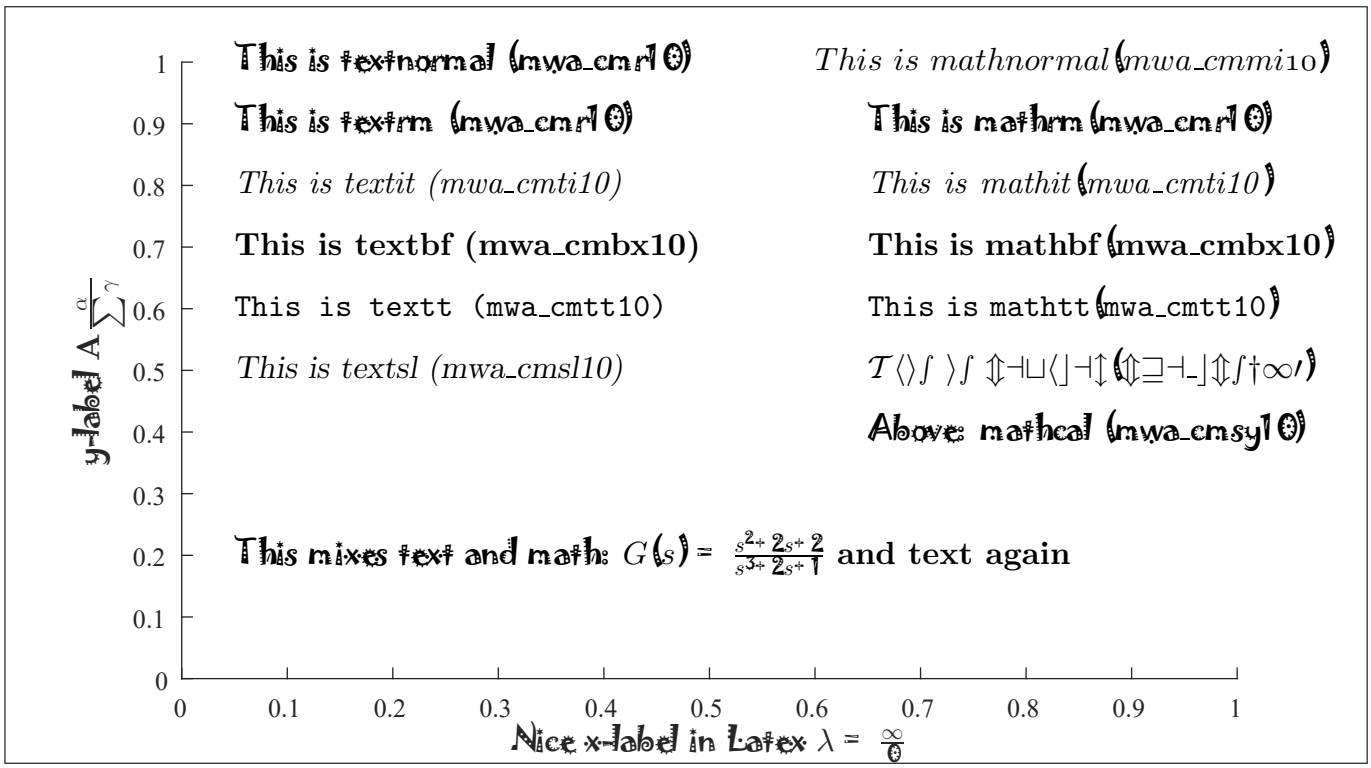


Figure 9: *Pdf* obtained from FontExample5.m

1. The fonts are replaced character by character. This means the character position is not modified. Depending on the size of the replacement font the characters can overlap or look too separated.
2. Some combinations of Matlab fonts with some font replacements can produce wrong symbols. For instance, if Comic Sans MS is used in the example, the letter 'n' is replaced by '-', even when in the *eps* file an 'n' is defined. I have no explanation for this, but it happens.
3. All the occurrences of the LaTeXfont are going to be replaced, even inside the equations, as can be seen in Fig. 9.
4. This method cannot be applied to Matlab 2016b.

# 6 Combining LaTeXand regular text

The solution for the aforementioned limitations is combine both the default interpreter with the LaTeXinterpreter. This can produce results as seen in Figs. 10 and 11.

## 6.1 Obtaining the x-label and y-labels

First we create the contents with the LaTeXinterpreter to obtain its position. Then we split in three (or more in a general) parts with the different texts with the desired fonts. We calculate the position for each text object based on the previous object. Finally we remove the original x-label.

Listing 14: FontExample6.m: x- and y-labels with mixed fonts

```
% A mixed latex and other fonts x-label
% Create the label in LaTex to get the position
xl=xlabel('{Time (s)} $\lambda_r = \frac{di}{dt}$ and so ...
    on','Interpreter','latex','fontsize',12,'margin',1,'Units','pixels');
```

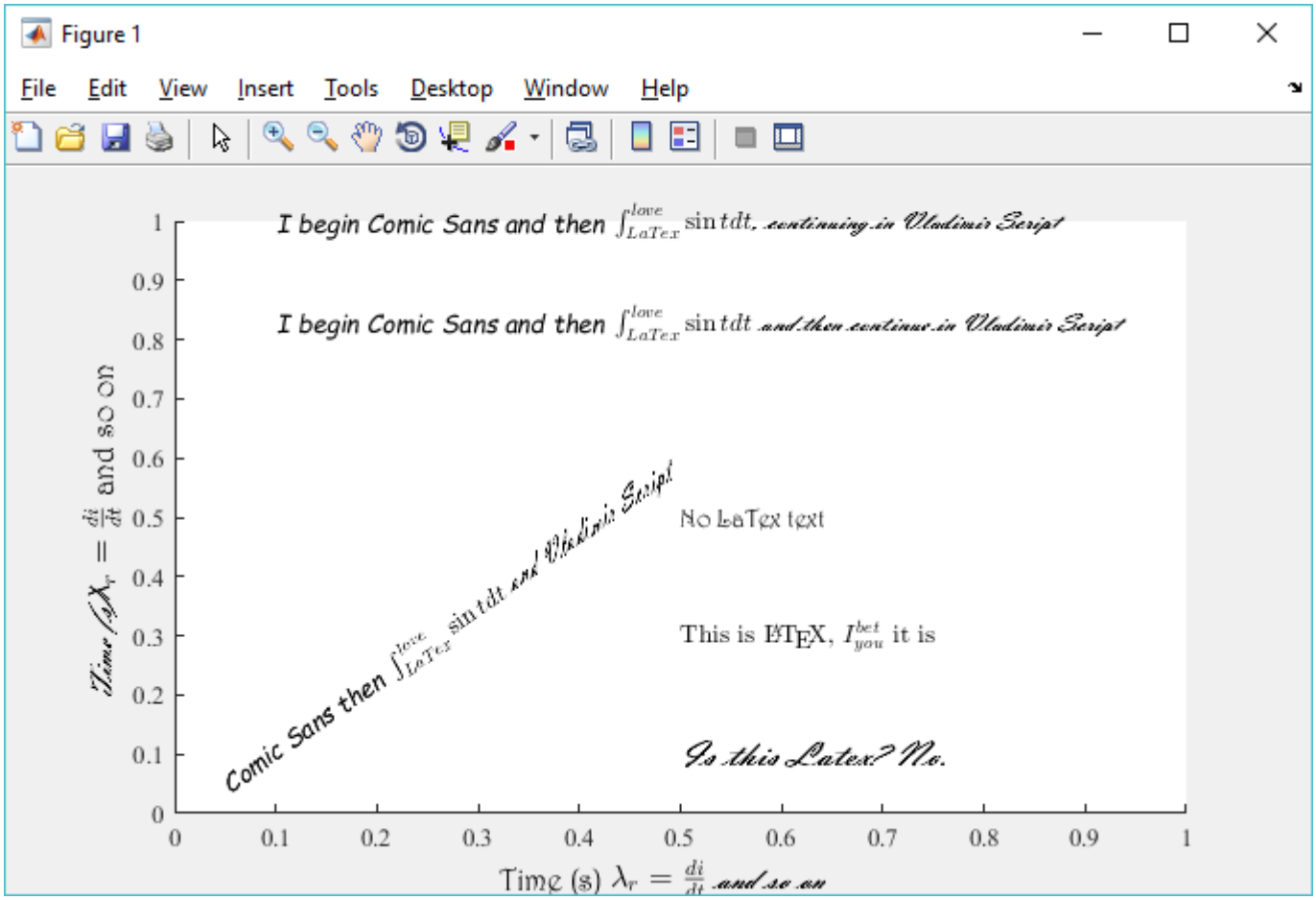


Figure 10: Figure obtained from FontExample6.m

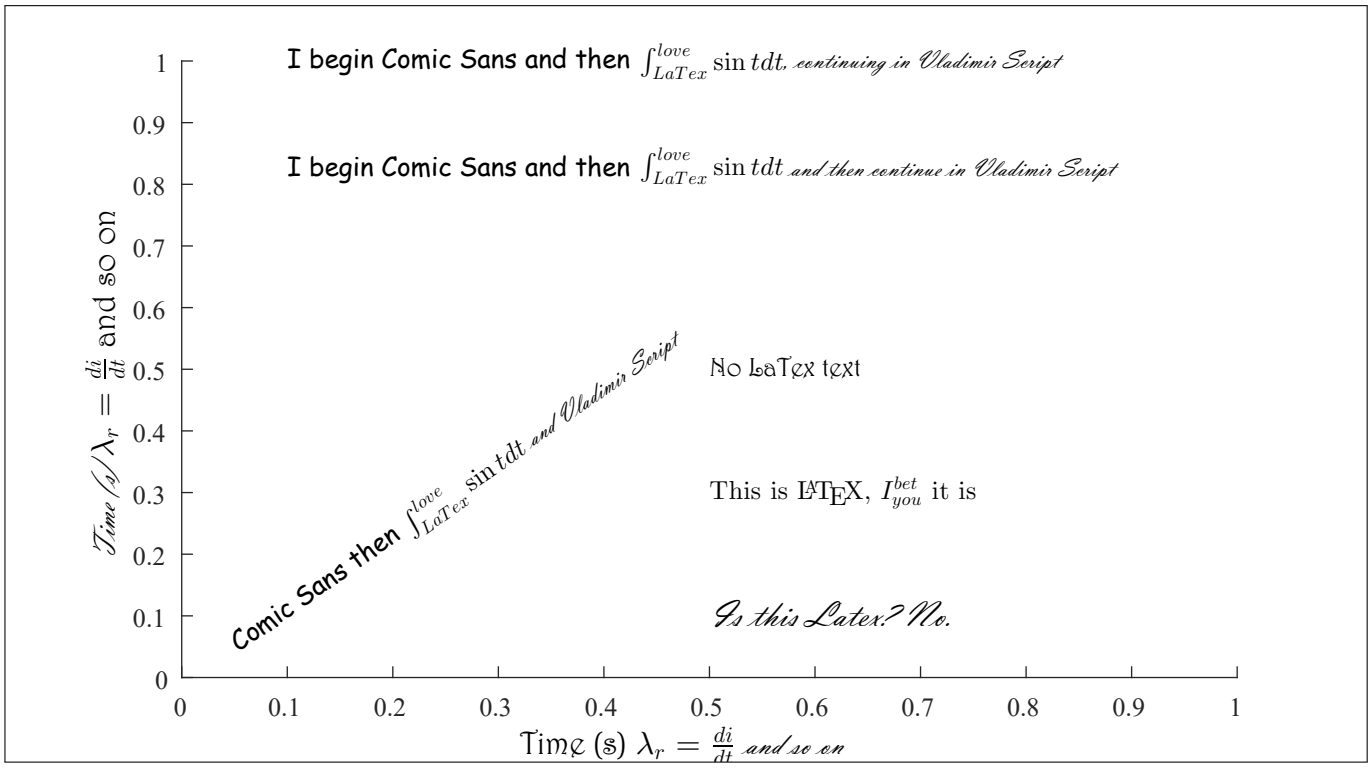


Figure 11: *Pdf* obtained from FontExample6.m

```
% Create the text replacements
t1= text(0,0,'Time (s) ','fontsize',12,'fontname','Harrington','Units','pixels');
t1.VerticalAlignment = xl. VerticalAlignment;
t1.Position = [xl.Extent(1)+1 xl.Position(2) 0];
t1.Units = 'data';
t2 = text(0,0,'$\lambda_r = \frac{di}{dt}$','Interpreter','latex','fontsize',12);
t2.VerticalAlignment = t1.VerticalAlignment;
t2.Position = [t1.Extent(1)+t1.Extent(3) t1.Position(2) 0];
t3 = text(0,0,' and so on','fontname','Vladimir Script','fontsize',12);
t3.VerticalAlignment = t2.VerticalAlignment;
t3.Position = [t2.Extent(1)+t2.Extent(3) t2.Position(2) 0];
xlabel(''); % Delete the x-label

% A mixed latex and other fonts y-label
% Create the label in LaTex to get the position
yl=ylabel('{Time (s)} $\lambda_r = \frac{di}{dt}$ and so ...
    on','Interpreter','latex','fontsize',12,'margin',1,'Units','pixels');
% Create the text replacements
t1= text(0,0,'Time (s) ','fontsize',12,'fontname','Vladimir Script','Units','pixels','rotation',90);
t1.VerticalAlignment = yl. VerticalAlignment;
t1.Position = [yl.Position(1) yl.Extent(2)+1 0];
t1.Units = 'data';
t2 = text(0,0,'$\lambda_r = \frac{di}{dt}$','Interpreter','latex','fontsize',12,'rotation',90);
t2.VerticalAlignment = t1.VerticalAlignment;
t2.Position = [t1.Position(1) t1.Extent(2)+t1.Extent(4) 0];
t3 = text(0,0,' and so on','fontname','Harrington','fontsize',12,'rotation',90);
t3.VerticalAlignment = t2.VerticalAlignment;
t3.Position = [t2.Position(1) t2.Extent(2)+t2.Extent(4) 0];
ylabel(''); % Delete the y-label
```

## 6.2 General text

We can do the same for general text objects inside the plot. The following code shows some examples. The code for producing rotating text is valid for the [-90-0-90] degree range. I leave you as an exercise to complete for the [0-360] range ;)

Listing 15: FontExample6.m: Mixing fonts in text objects

```
% A LaTex in the middle example with mixed fonts (middle aligned)
t1 = text(.1,1,'I begin Comic Sans and then ','fontname','Comic Sans MS','VerticalAlignment','middle');
t2 = text(0,0,'$\int^{love}_{LaTex}\sin{t} dt$','Interpreter','latex','VerticalAlignment','middle');
t2.Position = [t1.Extent(1)+t1.Extent(3) t1.Position(2) 0]; % Calculate position
t3 = text(0,0,', continuing in Vladimir Script','fontname','Vladimir ...
    Script','VerticalAlignment','middle');
t3.Position = [t2.Extent(1)+t2.Extent(3) t2.Position(2) 0]; % Calculate position

%% A LaTex in the middle example with mixed fonts (bottom aligned)
t1 = text(.1,.8,'I begin Comic Sans and then ','fontname','Comic Sans ...
    MS','VerticalAlignment','bottom');
t2 = text(0,0,'$\int^{love}_{LaTex}\sin{t} dt$','Interpreter','latex','VerticalAlignment','bottom');
t2.Position = [t1.Extent(1)+t1.Extent(3) t1.Position(2) 0];
t3 = text(0,0,' and then continue in Vladimir Script','fontname','Vladimir ...
    Script','VerticalAlignment','bottom');
t3.Position = [t2.Extent(1)+t2.Extent(3) t2.Position(2) 0];

%% A LaTex in the middle example with mixed fonts (middle aligned and rotated)
angle = 35; % rotation angle in degrees [-90 90]
t1 = text(.05,.05,'Comic Sans then ','fontname','Comic Sans ...
    MS','margin',1,'VerticalAlignment','middle');
t1.Units = 'centimeters'; % I change units after defining the text to write the position in data units
length = t1.Extent(3);
t1.Rotation = angle;
brx = t1.Position(1)+length*cosd(angle);
bry = t1.Position(2)+length*sind(angle);
t2 = text(brx,bry,'$\int^{love}_{LaTex}\sin{t} ...
    dt$','Interpreter','latex','margin',1,'VerticalAlignment','middle','Units','centimeters');
length = t2.Extent(3);
t2.Rotation = angle;
brx = t2.Position(1)+length*cosd(angle);
bry = t2.Position(2)+length*sind(angle);
```

```
28  t3 = text(brx,bry,' and Vladimir Script','fontname','Vladimir ...
        Script','rotation',angle,'margin',1,'VerticalAlignment','middle','Units','centimeters');
29  t3.Rotation = angle;
```

# 7 Final comment

I have spent some time in coming up with a solution for the font problem, but too little in writing this tutorial. Please, feel free to comment on it, and point out the several grammar and spelling errors it might contain.