exampleA.m : file for publish_mpl showing extra options

This file will be used to demonstrate the possibilies of the new function `publish_mpl` . The new function expands the possibilities of the standard `publish` function with regard to the LaTeX format. Motivation for creating this new function is that I want more control over the output than the pdf and html format can offer. So LaTeX is the obvious choice but at the same time I want to avoid manual editing of the tex file handle as much as possible. By using an adapted xsl file, the package matlab-prettifier created by Julien Cretel and using additional `publish` options we can achieve the following:

1. determine the documentclass and layout of the document
2. show MATLAB code (and also listings of mfiles) in a nice layout
3. specify hyperref options that determine the pdf attributes
4. determine how the header of the document is presented (titel, author, list of figures and listings)
5. include captions and references

Contents

Acknowledgement

This file is adapted from the `fourier_demo2.m` file that is included in MATLAB and can be copied in the current directory with


```
copyfile(fullfile(matlabroot,'help','techdoc',...
'matlab_env','examples','fourier_demo2.m'),'.','f')
```

Square Waves from Sine Waves

The Fourier series expansion for a square-wave is made up of a sum of odd harmonics, as shown here by the plots in figure 1 on page 2 (1 harmonic), figure 2 on page 3 (5 harmonics) and figure 3 on page 4 (9 harmonics).

```
if exist('avalue','var')
    fprintf('print the value passed to this script: %f\n',avalue)
else
    fprintf('no value passed to this script\n')
end
```

```
print the value passed to this script: 2.000000
```

Add an Odd Harmonic and Plot It

```
t    = 0:.1:pi*4;
k    = 1 ;
y    = sin(k*t)/k;
figure(k)
plot(t,y);
title(sprintf('MATLAB caption: plot when k=%.0f',k))
```

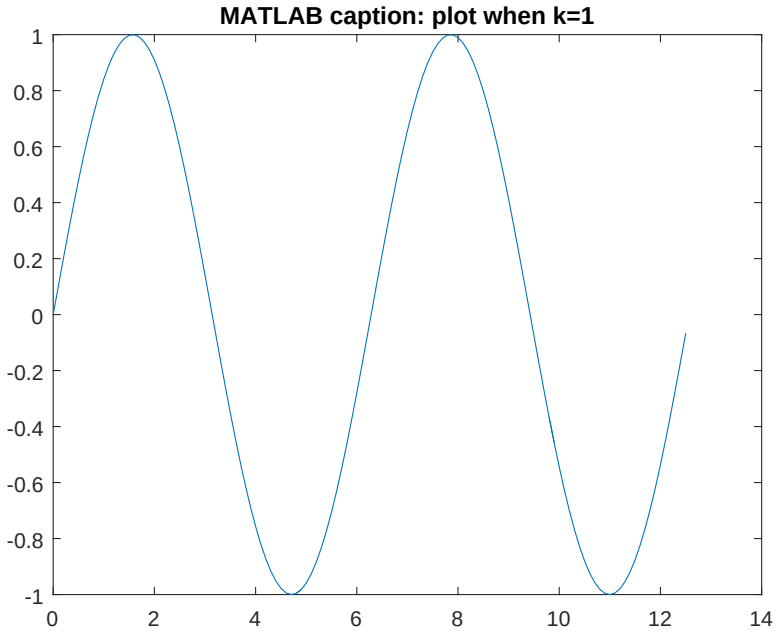

Figure 1: first harmonic

In each iteration of the for loop add an odd harmonic to y. As k increases, the output approximates a square wave with increasing accuracy.

Perform the following mathematical operation at each iteration:

$$y = y + \frac{\sin kt}{k}$$

Display some of the plots:

```
for k = 3:2:9
    y = y + sin(k*t)/k;
    if mod(k,4)==1
        figure(k)
        plot(t,y)
        title(sprintf('MATLAB caption: plot when k=%.0f',k))
    end
end
```

Note About Gibbs Phenomenon

Even though the approximations are constantly improving, they will never be exact because of the Gibbs phenomenon, or ringing.

Listing of this script

```
%% exampleA.m : file for publish_mpl showing extra options
% This file will be used to demonstrate the possibilies
% of the new function |publish_mpl| . The new function
% expands the possibilities of the standard |publish| function
% with regard to the LaTeX format. Motivation for creating
% this new function is that I want more control over the output than
   the
```

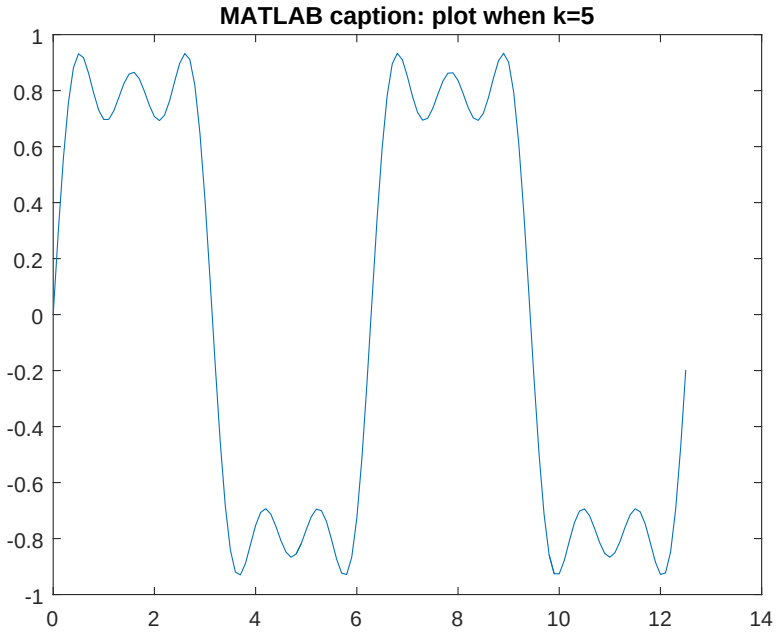

Figure 2: sum of first 5 harmonics

```
% pdf and html format can offer. So LaTeX is the obvious choice
% but at the same time I want to avoid manual editing of the
% tex file handle as much as possible. By using an adapted
% xsl file, the package matlab-prettifier created by Julien Cretel
% and using additional |publish| options we can achieve the following:
%
% # determine the documentclass and layout of the document
% # show MATLAB code (and also listings of mfiles) in a nice layout
% # specify hyperref options that determine the pdf attributes
% # determine how the header of the document is presented (titel,
   author, list of figures and listings)
% # include captions and references
%% Acknowledgement
% This file is adapted from the |fourier_demo2.m| file
% that is included in MATLAB and can be copied in
% the current directory with
%%
%
%  copyfile(fullfile(matlabroot,'help','techdoc',...
%  'matlab_env','examples','fourier_demo2.m'),'.','f')
%%
%% Square Waves from Sine Waves
% <latex>
% % The actual function to publish starts now.
% % This text block is changed to a latex block to show the caption and
    reference capabilities
% %
% % the following statements insert the references to the plots:
% The Fourier series expansion for a square-wave is
% made up of a sum of odd harmonics, as shown here
% by the plots in figure \ref{exampleA_01.eps} on page \pageref{
   exampleA_01.eps} (1 harmonic),
% figure \ref{exampleA_02.eps} on page \pageref{exampleA_02.eps} (5
   harmonics) and
```

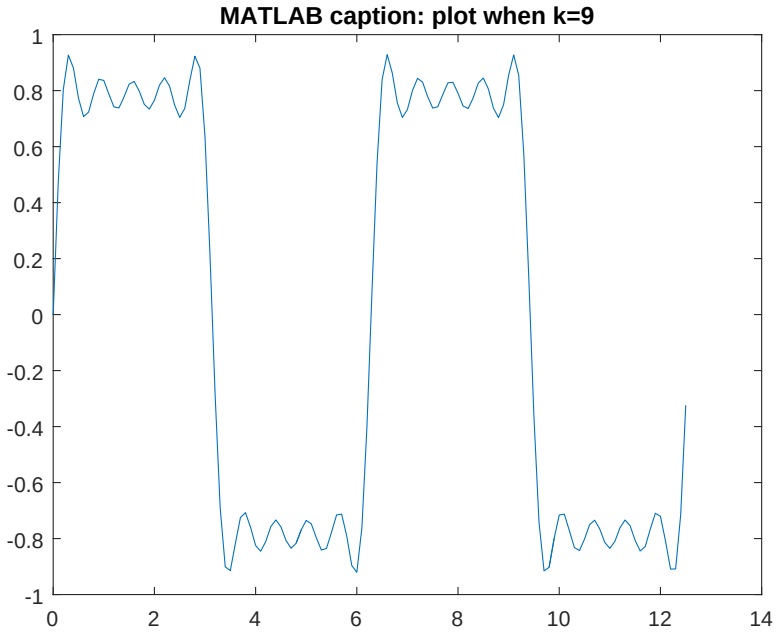

Figure 3: sum of first 9 harmonics

```
% figure \ref{exampleA_03.eps} on page \pageref{exampleA_03.eps} (9
   harmonics).
% %
% % the following statements define the captions of the plots:
% \global\def\captionA{first harmonic}
% \global\def\captionB{sum of first 5 harmonics}
% \global\def\captionC{sum of first 9 harmonics}
% </latex>
if exist('avalue','var')
    fprintf('print the value passed to this script: %f\n',avalue)
else
    fprintf('no value passed to this script\n')
end
%% Add an Odd Harmonic and Plot It
t    = 0:.1:pi*4;
k    = 1 ;
y    = sin(k*t)/k;
figure(k)
plot(t,y);
title(sprintf('MATLAB caption: plot when k=%.0f',k))

%%
% In each iteration of the for loop add an odd
% harmonic to y. As _k_ increases, the output
% approximates a square wave with increasing accuracy.
%
% Perform the following mathematical operation
% at each iteration:
%
% $$ y = y + \frac{\sin kt}{k} $$
%
% Display some of the plots:
%
```

```
for k = 3:2:9
    y = y + sin(k*t)/k;
    if mod(k,4)==1
        figure(k)
        plot(t,y)
        title(sprintf('MATLAB caption: plot when k=%.0f',k))
    end
end

%% Note About Gibbs Phenomenon
% Even though the approximations are constantly
% improving, they will never be exact because of the
% Gibbs phenomenon, or ringing.
%% Listing of this script
% <latex>
% % assuming m-file in directory one level higher than tex dir (using
   the standard html subdirectory)
% % assuming numbers and framed are not set in \usepackage and they are
    wanted
% % \lstinputlisting[frame=single,numbers=left]{../exampleA.m}
% % assuming numbers and framed are set in \usepackage and they are not
    wanted
% % \lstinputlisting[frame=none,numbers=none]{../exampleA.m}
% % assuming numbers and framed are set in \usepackage are set and
   wanted
% \lstinputlisting{../exampleA.m}
% </latex>
%% Listing of publish_mpl_examples.m
% <latex>
% \lstinputlisting{../publish_mpl_examples.m}
% </latex>
```

Listing of publish_mpl_examples.m

```
addpath('../code')
%% example1: -> pdf
% Use the function to create pdf-file.
% This is the same as using the publish user interface.
mycode = { ...                                          % example of
   code to execute (two lines)
               'avalue = 2;'   ...
           'exampleA'   ...
           } ;
pstruct = struct( ...                                   % publish
   options
    'format' , 'pdf' , ...                              % output format
    'call' , {mycode} , ...                             % code to
       execute (defined above)
    'newname' , 'exampleA1.pdf' );                       % new name of
       output file
newname = publish_mpl('exampleA', pstruct) ;            % produce the
   output file (pdf)

%% example2: -> latex
% Use the function to create tex-file
% with as much as possible the same layout
% as the original tex file but with references, captions
% and listings
```

```
mycode = { ...                                          % example of
    code to execute (one line)
             'exampleA'   ...
             } ;
pstruct = struct( ...                                   % publish
    options
     'format' , 'latex' , ...                           % output format
        latex using the new xsl file
     'call' , {mycode} , ...                            % code to
        execute (defined above)
     'orientation', 'portrait', ...                     % overwrite
        orientation (default 'landscape')
     'newname' , 'exampleA2.tex' , ...                  % new name of
        output file
     'prettifier_options' , '' ) ;                       % overwrite
        prettify options (default 'framed,numbered')

newname = publish_mpl('exampleA', pstruct) ;            % produce the
    output file (tex)

%% example3: -> latex
% Same as example2 but the layout is landscape and
% the MATLAB code will be in frames with numbers.
mycode = { ...                                          % example of
    code to execute (one line)
             'exampleA'   ...
             } ;
pstruct = struct( ...                                   % publish
    options
     'format' , 'latex' , ...                           % output format
        latex using the new xsl file
     'call' , {mycode} , ...                            % code to
        execute (defined above)
     'newname' , 'exampleA3.tex');                       % new name of
        output file
 %   'orientation', 'landscape', ...                     % use default
     orientation ('landscape')
 %   'prettifier_options', 'framed,numbered' , ...       % use default
     prettify options ('framed,numbered')

newname = publish_mpl('exampleA', pstruct) ;            % produce the
    output file (tex)
%% example4: -> latex
% same as example3 but listings have their own
% caption in exampleB and they are listed by
% setting 'makelstlistoflistings' to true
mycode = { ...                                          % example of
    code to execute (one line)
             'exampleB'   ...
             } ;
pstruct = struct( ...                                   % publish
    options
     'format' , 'latex' , ...                           % output format
        latex using the new xsl file
     'call' , {mycode} , ...                            % code to
        execute (defined above)
     'newname' , 'exampleB1.tex' , ...                  % new name of
        output file
```

```
    'pdfauthor', 'han@hanoostdijk.nl' , ...                % insert a
       pdf option
    'makelstlistoflistings', true);                        % create
       lstlistoflistings
newname = publish_mpl('exampleB', pstruct) ;               % produce the
   output file (tex)

%% example5: -> latex
% same as example4 but now with a regular LaTeX contents
% by setting 'maketableofcontents' to true
mycode = { ...                                             % example of
   code to execute (one line)
            'exampleB'   ...
            } ;
pstruct = struct( ...                                      % publish
   options
    'format' , 'latex' , ...                               % output format
       latex using the new xsl file
    'call' , {mycode} , ...                                % code to
       execute (defined above)
    'newname' , 'exampleB2.tex' , ...                      % new name of
       output file
    'pdfauthor', 'han@hanoostdijk.nl' , ...                % insert a
       pdf option
    'maketableofcontents', true , ...                      % create
       tableofcontents
    'makelstlistoflistings', true);                        % create
       lstlistoflistings
newname = publish_mpl('exampleB', pstruct) ;               % produce the
   output file (tex)

%% example6: -> xml
% same as example3 but now to xml format
mycode = { ...                                             % example of
   code to execute (one line)
            'exampleB'   ...
            } ;
pstruct = struct( ...                                      % publish
   options
    'format' , 'xml' , ...                                % output format
       latex using the new xsl file
    'call' , {mycode} , ...                                % code to
       execute (defined above)
    'newname' , 'exampleB3.xml' , ...                      % new name of
       output file
    'pdfauthor', 'han@hanoostdijk.nl' , ...                % insert a
       pdf option
    'makelstlistoflistings', true);                        % create
       lstlistoflistings
newname = publish_mpl('exampleB', pstruct) ;               % produce the
   output file (tex)
```