mglTEX package example

Diego Sejas Viscarra, Alexey Balakin

June 15, 2016

The LaTeX package mglTeX (was made by Diego Sejas Viscarra) allows one to make figures directly from MGL scripts located in LaTeX file.

For using this package you need to specify `--shell-escape` option for *latex/pdflatex* or manually run *mglconv* tool on produced MGL scripts for generation of images. Don't forget to run *latex/pdflatex* a second time to insert the generated images into the output document.

The package may have following options: `draft`, `final` — the same as in the *graphicx* package; `on`, `off` — to activate/deactivate the creation of scripts and graphics; `comments`, `nocomments` — to make visible/invisible commentaries contained inside `mglcomment` environments; `jpg`, `jpeg`, `png` — to export graphics as JPEG/PNG images; `eps`, `epsz` — to export to uncompressed/compressed EPS format as primitives; `bps`, `bpsz` — to export to uncompressed/compressed EPS format as bitmap (doesn't work with *pdflatex*); `pdf` — to export to 3D PDF; `tex` — to export to LaTeX/*tikz* document.

The package defines the following environments:

mgl It writes its contents to a general script which has the same name as the LaTeX document, but its extension is *.mgl*. The code in this environment is compiled and the image produced is included. It takes exactly the same optional arguments as the `\includegraphics` command, plus an additional argument *imgext*, which specifies the extension to save the image.

mgladdon It adds its contents to the general script, without producing any image. It useful to set some global properties (like size of the images) at beginning of the document.

mglcode Is exactly the same as `mgl`, but it writes its contents verbatim to its own file, whose name is specified as a mandatory argument.

mglscript Is exactly the same as `mglcode`, but it doesn't produce any image, nor accepts optional arguments. It is useful, for example, to create a MGL script, which can later be post processed by another package like "listings".

mglblock It writes its contents verbatim to a file, specified as a mandatory argument, and to the LaTeX document, and numerates each line of code.

mglverbatim Exactly the same as `mglblock`, but it doesn't write to a file. This environment doesn't have arguments.

mglfunc Is used to define MGL functions. It takes one mandatory argument, which is the name of the function, plus one additional argument, which specifies the number of arguments of the function. The environment needs to contain only the body of the function, since the first and last lines are appended automatically, and the resulting code is written at the end of the general script, which is also written automatically. The warning is produced if 2 or more function with the same name is defined.

mglsignature Used to defined a commentary that will be added to every script. It is useful to include signature text or license text. Observe this is a verbatim-like environment, so no LaTeX command will be executed inside it, but will be copied as is.

As an alternative to this method of declaring signatures, the user can manually redefine the signature macro `\mgltexsignature`, according to the following rules:

- The positions of the comment signs for the MGL language have to be manually specified in the signature using the `\mglcomm` macro.
- The new-line character is declared as "`^^J`".
- A percent sign (`%`) has to be added at the end of every physical line of `\mgltexsignature`, otherwise an inelegant space at the beginning of every line will appear.
- Any LaTeX command can be used in this case.

For example, the default signature:

```
<------------------ MGL comment ------------------>
                \begin{quote}\small
                    \mglcomm\\
\mglcomm\ This script was generated from $<$document$>$.mgl on date
                    $<$today$>$\\
                      \mglcomm
                     \end{quote}
<------------------ MGL comment ------------------>
```

can be achieved with

```
\def\mgltexsignature{%
  \mglcomm^^J%
  \mglcomm\ This script was generated from \jobname.mgl on date \today^^J%
  \mglcomm%
}
```

mglcomment Used to contain multiline commentaries. This commentaries will be visible/invisible in the output document, depending on the use of the package options `comments` and `nocomments` (see above), or the and commands (see bellow).

When, visible, the comment will appear like this:

```
<------------------ MGL comment ------------------>
                    <Commentary>
<------------------ MGL comment ------------------>
```

mglsetup If many scripts with the same code are to be written, the repetitive code can be written inside this environment only once, then this code will be used automatically every time the `\mglplot` command is used (see below). It takes one optional argument, which is a name to be associated to the corresponding contents of the environment; this name can be passed to the `\mglplot` command to use the corresponding block of code automatically (see below).

The package also defines the following commands:

\mglplot It takes one mandatory argument, which is MGL instructions separated by the symbol ':' this argument can be more than one line long. It takes the same optional arguments as the `mgl` environment, plus an additional argument *settings*, which indicates the name associated to a block of code inside a `mglsetup` environment. The code inside the mandatory argument will be appended to the block of code specified, and the resulting code will be written to the general script.

\mglgraphics This command takes the same optional arguments as the `mgl` environment, and one mandatory argument, which is the name of a MGL script. This command will compile the corresponding script and include the resulting image. It is useful when you have a script outside the LaTeX document, and you want to include the image, but you don't want to type the script again.

\mglinclude This is like `\mglgraphics` but, instead of creating/including the corresponding image, it writes the contents of the MGL script to the LaTeX document, and numerates the lines.

\mgldir This command can be used in the preamble of the document to specify a directory where LaTeX will save the MGL scripts and generate the corresponding images. This directory is also where `\mglgraphics` and `\mglinclude` will look for scripts.

\mglquality Can be used to adjust the quality of the MGL graphics produced. The following table shows the available qualities:

Quality	Description
0	No face drawing (fastest)
1	No color interpolation (fast)
2	High quality (normal)
3	High quality with 3d primitives (not implemented yet)
4	No face drawing, direct bitmap drawing (low memory usage)
5	No color interpolation, direct bitmap drawing (low memory usage)
6	High quality, direct bitmap drawing (low memory usage)
7	High quality with 3d primitives, direct bitmap drawing (not implemented yet)
8	Draw dots instead of primitives (extremely fast)

\mgltexon, \mgltexoff To activate/deactivate the creation of MGL scripts and images. Notice these commands have local behavior in the sense that their effect is from the point they are called on.

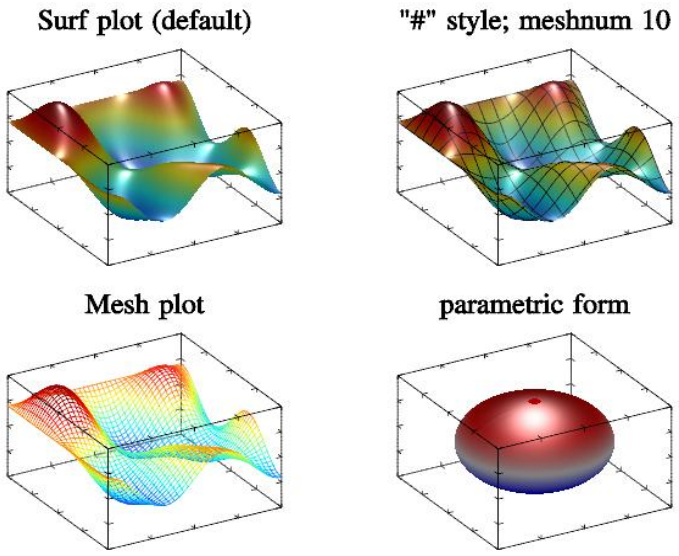
\mglcomment, \mglnocomment To make visible/invisible the contents of the `mglcomment` environments. These commands have local effect too.

\mglTeX It just pretty prints the name of the package "mglTEX".

An example of usage of `mgl` and `mglfunc` environments would be:

```
\begin{mglfunc}{prepare2d}
  new a 50 40 '0.6*sin(pi*(x+1))*sin(1.5*pi*(y+1))+0.4*cos(0.75*pi*(x+1)*(y+1))'
  new b 50 40 '0.6*cos(pi*(x+1))*cos(1.5*pi*(y+1))+0.4*cos(0.75*pi*(x+1)*(y+1))'
\end{mglfunc}

\begin{figure}[!ht]
  \centering
  \begin{mgl}[width=0.85\textwidth,height=7.5cm]
    fog 0.5
    call 'prepare2d'
    subplot 2 2 0:title 'Surf plot (default)':rotate 50 60:light on:box:surf a

    subplot 2 2 1:title '"\#" style; meshnum 10':rotate 50 60:box
    surf a '#'; meshnum 10

    subplot 2 2 2 : title 'Mesh plot' : rotate 50 60 : box
    mesh a

    new x 50 40 '0.8*sin(pi*x)*sin(pi*(y+1)/2)'
    new y 50 40 '0.8*cos(pi*x)*sin(pi*(y+1)/2)'
    new z 50 40 '0.8*cos(pi*(y+1)/2)'
    subplot 2 2 3 : title 'parametric form' : rotate 50 60 : box
    surf x y z 'BbwrR'
  \end{mgl}
\end{figure}
```

Note, that `mglfunc` environment(s) can be located at any position (at the beginning, at the end, or somewhere else) of LaTeX document.



Following example show the usage of `mglscript` environment

```
\begin{mglscript}{Vectorial}
call 'prepare2v'
subplot 3 2 0 '' : title 'lolo' : box
vect a b
subplot 3 2 1 '' : title '"." style; "=" style' : box
vect a b '.='
subplot 3 2 2 '' : title '"f" style' : box
vect a b 'f'
subplot 3 2 3 '' : title '">" style' : box
vect a b '>'
subplot 3 2 4 '' : title '"<" style' : box
vect a b '<'
call 'prepare3v'
subplot 3 2 5 : title '3d variant' : rotate 50 60 : box
vect ex ey ez

stop

func 'prepare2v'
  new a 20 30 '0.6*sin(pi*(x+1))*sin(1.5*pi*(y+1))+0.4*cos(0.75*pi*(x+1)*(y+1))'
  new b 20 30 '0.6*cos(pi*(x+1))*cos(1.5*pi*(y+1))+0.4*cos(0.75*pi*(x+1)*(y+1))'
return
```

```
func 'prepare3v'
  define $1 pow(x*x+y*y+(z-0.3)*(z-0.3)+0.03,1.5)
  define $2 pow(x*x+y*y+(z+0.3)*(z+0.3)+0.03,1.5)
  new ex 10 10 10 '0.2*x/$1-0.2*x/$2'
  new ey 10 10 10 '0.2*y/$1-0.2*y/$2'
  new ez 10 10 10 '0.2*(z-0.3)/$1-0.2*(z+0.3)/$2'
return
\end{mglscript}
```

You should use `\mglgraphics` command to display its contents

```
\begin{figure}[!ht]
  \centering
  \mglgraphics[width=40em,height=20em]{Vectorial}
  \caption{A beautiful example}
\end{figure}
```

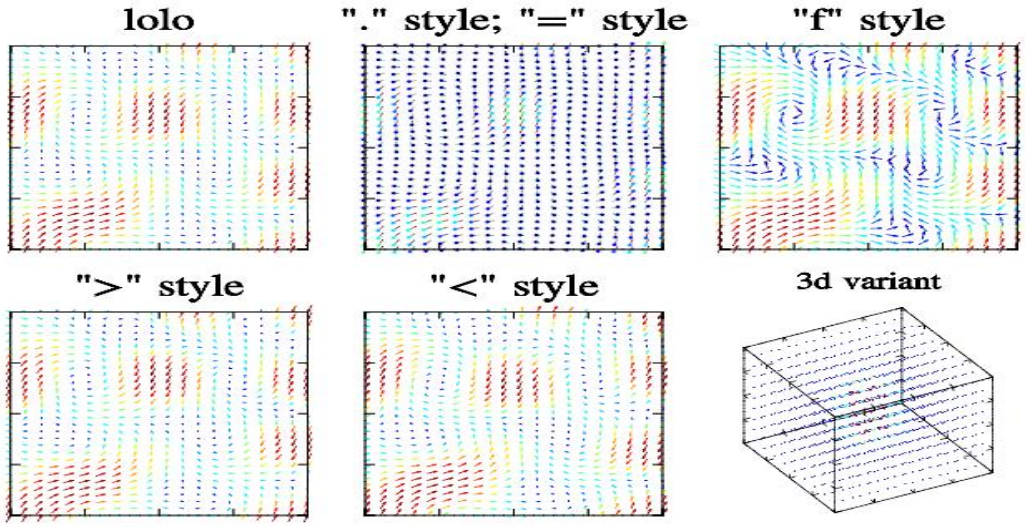


Figure 1: A beautiful example

Alternatively, you can display the contents of the script in parallel to saving to a file, if you are using `mglblock` environment

```
\begin{mglblock}{Axis_projection}
  ranges 0 1 0 1 0 1
  new x 50 '0.25*(1+cos(2*pi*x))'
  new y 50 '0.25*(1+sin(2*pi*x))'
  new z 50 'x'
  new a 20 30 '30*x*y*(1-x-y)^2*(x+y<1)'
  new rx 10 'rnd':new ry 10:fill ry '(1-v)*rnd' rx
```

```
  light on

  title 'Projection sample':ternary 4:rotate 50 60
  box:axis:grid
  plot x y z 'r2':surf a '#'
  xlabel 'X':ylabel 'Y':zlabel 'Z'
\end{mglblock}
\begin{figure}[!ht]
  \centering
  \mglgraphics[scale=0.5]{Axis_projection}
  \caption{The image from Axis\_projection.mgl script}
\end{figure}
```

```
#   This scripts was generated on date June 15, 2016.

 ranges 0 1 0 1 0 1
  new x 50 '0.25*(1+cos(2*pi*x))'
  new y 50 '0.25*(1+sin(2*pi*x))'
  new z 50 'x'
  new a 20 30 '30*x*y*(1-x-y)^2*(x+y<1)'
  new rx 10 'rnd':new ry 10:fill ry '(1-v)*rnd' rx
  light on

  title 'Projection sample':ternary 4:rotate 50 60
  box:axis:grid
  plot x y z 'r2':surf a '#'
  xlabel 'X':ylabel 'Y':zlabel 'Z'
```

Finally, you can just show MGL script itself

```
\begin{mglverbatim}
  ranges 0 1 0 1 0 1
  new x 50 '0.25*(1+cos(2*pi*x))'
  new y 50 '0.25*(1+sin(2*pi*x))'
  new z 50 'x'
  new a 20 30 '30*x*y*(1-x-y)^2*(x+y<1)'
  new rx 10 'rnd':new ry 10:fill ry '(1-v)*rnd' rx
  light on

  title 'Projection sample':ternary 4:rotate 50 60
  box:axis:grid
  plot x y z 'r2':surf a '#'
  xlabel 'X':ylabel 'Y':zlabel 'Z'
\end{mglverbatim}
```

```
 ranges 0 1 0 1 0 1
  new x 50 '0.25*(1+cos(2*pi*x))'
```

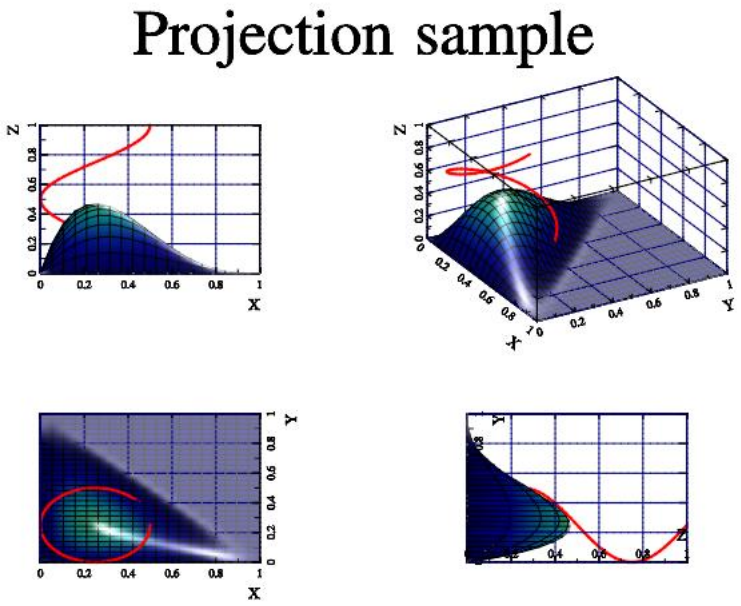


Figure 2: The image from Axis_projection.mgl script

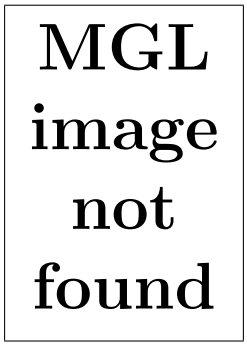
```
3.    new y 50 '0.25*(1+sin(2*pi*x))'
4.    new z 50 'x'
5.    new a 20 30 '30*x*y*(1-x-y)^2*(x+y<1)'
6.    new rx 10 'rnd':new ry 10:fill ry '(1-v)*rnd' rx
7.    light on
8.
9.    title 'Projection sample':ternary 4:rotate 50 60
10.   box:axis:grid
11.   plot x y z 'r2':surf a '#'
12.   xlabel 'X':ylabel 'Y':zlabel 'Z'
```

An example of usage of \mglplot command would be:

```
\begin{mglsetup}
  box '@{W9}' : axis
\end{mglsetup}
\begin{mglsetup}[2d]
  box : axis
  grid 'xy' ';k'
\end{mglsetup}
\begin{mglsetup}[3d]
  rotate 50 60
  box : axis : grid 'xyz' ';k'
```

```
\end{mglsetup}
\begin{figure}[!ht]
  \centering
  \mglplot[scale=0.5]{new a 200 'sin(pi*x)':plot a '2B'}
\end{figure}
\begin{figure}[!ht]
  \centering
  \mglplot[scale=0.5,settings=2d]{
    fplot 'sin(pi*x)' '2B' :
    fplot 'cos(pi*x^2)' '2R'
  }
\end{figure}
\begin{figure}[!ht]
  \centering
  \mglplot[width=0.5 \textwidth, settings=3d]
  {fsurf 'sin(pi*x)+cos(pi*y)'}
\end{figure}
```

As an additional feature, when an image is not found or cannot be included, instead of issuing an error, `mgltex` prints a box with the word *'MGL image not found'* in the LaTeX document.



Let's display the content of the MGL file using `\mglinclude` command:

```
1.  #   This scripts was generated on date June 15, 2016.
2.
3.  call 'prepare2v'
4.  subplot 3 2 0 '' :  title 'lolo' :  box
5.  vect a b
```

```
6.  subplot 3 2 1 '' :  title '"." style; "=" style' :  box
7.  vect a b '.='
8.  subplot 3 2 2 '' :  title '"f" style' :  box
9.  vect a b 'f'
10. subplot 3 2 3 '' :  title '">" style' :  box
11. vect a b '>'
12. subplot 3 2 4 '' :  title '"<" style' :  box
13. vect a b '<'
14. call 'prepare3v'
15. subplot 3 2 5 :  title '3d variant' :  rotate 50 60 :  box
16. vect ex ey ez
17.
18. stop
19.
20. func 'prepare2v'
21.   new a 20 30 '0.6*sin(pi*(x+1))*sin(1.5*pi*(y+1))+0.4*cos(0.75*pi*(x+1)*(y+1))'
22.   new b 20 30 '0.6*cos(pi*(x+1))*cos(1.5*pi*(y+1))+0.4*cos(0.75*pi*(x+1)*(y+1))'
23. return
24.
25. func 'prepare3v'
26.   define $1 pow(x*x+y*y+(z-0.3)*(z-0.3)+0.03,1.5)
27.   define $2 pow(x*x+y*y+(z+0.3)*(z+0.3)+0.03,1.5)
28.   new ex 10 10 10 '0.2*x/$1-0.2*x/$2'
29.   new ey 10 10 10 '0.2*y/$1-0.2*y/$2'
30.   new ez 10 10 10 '0.2*(z-0.3)/$1-0.2*(z+0.3)/$2'
31. return
```

The following commentary will be visible, since mglTEX has been called with the `comments` option.

```
\begin{mglcomment}
  This is a visible commentary
  that can have multiple lines
\end{mglcomment}
```

The result is:

```
<------------------ MGL comment ------------------>
          This is a visible commentary
          that can have multiple lines
<------------------ MGL comment ------------------>
```

The following commentary won't be visible, since it is wrapped by `\mglnocomments` and `\mglcomments`.

```
\mglnocomments
\begin{mglcomment}
  This is an invisible commentary
```

```
  that can have multiple lines
\end{mglcomment}
\mglcomments
```

The last example is the use of the \mgltexon and \mgltexoff commands. For example, the following image won't be generated:

```
\mgltexoff
\begin{figure}[!ht]
  \centering
  \begin{mgl}
    box : axis
    fplot 'sin(pi*x)' '2B'
  \end{mgl}
\end{figure}
\mgltexon
```

The result is: