pgfmath–xfp

define pgfmath functions using xfp

Jonathan P. Spratte*

2021-05-19 v1.0

Abstract

pgfmath–xfp provides a small wrapper to define pgfmath functions which use the floating point unit of expl3 (of which the document-level interface is called xfp).

Contents

1 Documentation .............................. 1
  1.1 Document-Level Interface .............. 2
  1.2 Examples ................................ 2
  1.3 expl3-Level Interface .................. 3

2 Implementation ............................ 4
  2.1 General Package code ................ 4
  2.2 Document-Level Interface .......... 4
  2.3 expl3-Level Interface ................. 4
  2.4 Internals ................................ 5

Index ........................................ 7

1 Documentation

This package serves as a stopgap to allow the usage of xfp in pgfmath functions. It is only meant as a temporary fix to allow single functions using the expl3 fpu until a more sophisticated solution to allow broader support for it in \pgf is available.

The defined functions should work correctly independent of the surrounding pgfmath context. This is achieved by first parsing the arguments via \pgfmathsetmacro with \pgf settings applied locally to ensure that the resulting format is understandable by xfp’s fpu.

Though it has both pgfmath and xfp in its name, it only loads pgfmath as a dependency, the access to xfp’s fpu is done at the expl3 level.

It was created as a result of two questions on https://tex.stackexchange.com: expl3 cannot see declared functions and \pgf: “Dimension too large” in a function which fits into a graph, /pgf/fpu=true does not help.

*jspratte@yahoo.de
1.1 Document-Level Interface

Define a \texttt{pgfmath} function named \langle name \rangle that takes \langle arg-count \rangle arguments. The behaviour is different depending on whether the optional argument was used or not.

If it isn’t the \langle fp-expression \rangle can refer to the \langle arg-count \rangle arguments using \#1, \textit{etc}., and will get the arguments just like they are given to the function (translated to a format that \texttt{xfp} will understand by parsing them through \texttt{pgfmath} once).

If it is use \langle process-args \rangle to specify any number of processed arguments in a comma separated list. Inside this list you can specify up to nine processed arguments using \texttt{pgfmath} functions in which you can refer to the arguments passed to your new function using \#1, \textit{etc}. You can refer to these processed arguments inside \langle fp-expression \rangle using \#1, \textit{etc}. A result of this rule is that you have to explicitly use \#1 in \langle process-args \rangle to forward it unaltered to the underlying \texttt{xfp} expression.

1.2 Examples

The following are examples taken from the two questions responsible for this package.

\begin{verbatim}
\pgfmathdeclarefunction{lognormal}{3}{
  \exp(-((ln(#1) - #2)^2) / (2 * (#3)^2)) / (#1 * #3 * sqrt(2 * pi))}
\begin{tikzpicture}
\begin{axis} [domain=0.01:10, samples=100]
  \addplot {lognormal(x,ln(5),0.2)};
\end{axis}
\end{tikzpicture}
\end{verbatim}

Showing that a single \texttt{pgfmath} function argument can result in multiple arguments for the \texttt{xfp} expression. This example is suboptimal slow code, but could be educational.

\begin{verbatim}
\pgfmathdeclarefunction{fplog}{1}{ln(#1)}
\pgfmathdeclarefunction{nlogn}{1}{#1,fplog(#1)[#1 + #2]}
\end{verbatim}
\begin{tikzpicture}
\begin{axis}[
domain=0.01:10, samples=50
]\addplot[nlogn(x)];\end{axis}\end{tikzpicture}

1.3 \texttt{expl3}-Level Interface

\texttt{\textbackslash pgfmxfp\_declare:nnn} \hspace{1cm} \texttt{\textbackslash pgfmxfp\_declare:nnn (name) (arg-count) (fp-expression)}

Defines a \texttt{pgfmath} function named (\texttt{name}), that takes (\texttt{arg-count}) arguments. The function will evaluate the (\texttt{fp-expression}) using the \texttt{t3fp fpu} and store the result for \texttt{pgfmath}. The arguments can be referred using \#1, \textit{etc.}

\texttt{\textbackslash pgfmxfp\_declare\_processed\_args:nnnn} \hspace{1cm} \texttt{\textbackslash pgfmxfp\_declare\_processed\_args:nnnn (name) (arg-count) (processed-args) (fp-expression)}

Defines a \texttt{pgfmath} function named (\texttt{name}), that takes (\texttt{arg-count}) arguments. The arguments will be evaluated through \texttt{pgfmath} according to the comma separated list of (\texttt{processed-args}) (in which you can refer to the arguments using \#1, \textit{etc.}) and the results of which will be the arguments for the (\texttt{fp-expression}) (in which you can refer to the (\texttt{processed-args}) using \#1, \textit{etc.}).
2 Implementation

2.1 General Package code

Some code for versioning support might not be available in older \LaTeX\ releases.

\providecommand\DeclareRelease[3]{\quad}
\providecommand\DeclareCurrentRelease[2]{\quad}

Use these rollback functions to declare the current release.

\DeclareCurrentRelease{2021-05-19}

Make sure expl3 is available and load pgfmath and the PG\Fp\ fpu.

\IfUndefined{ExplFileDate}{\RequirePackage{expl3}}{}
\RequirePackage{pgfmath}
\usepgflibrary{fpu}

Store version and date in a macro.

\newcommand*{\pgfmxfpDate}{2021-05-19}
\newcommand*{\pgfmxfpVersion}{1.0}

(End definition for \pgfmxfpDate and \pgfmxfpVersion. These functions are documented on page ??.)

Provide the package.

\ProvidesExplPackage{pgfmath-xfp}{\pgfmxfpDate}{\pgfmxfpVersion}{define pgfmath functions using xfp}

2.2 Document-Level Interface

The document-level interface decides which of the two allocator functions are used.

\NewDocumentCommand{\pgfmxfpdeclarefunction}{m m o m}{
  \IfValueTF{#3}{\pgfmxfp_declare_processed_args:nnnn {#1} {#2} {#3} {#4}}{
    \pgfmxfp_declare:nnn {#1} {#2} {#3} {#4}}
}

(End definition for \pgfmxfpdeclarefunction. This function is documented on page 2.)

2.3 expl3-Level Interface

Start building the function body. First step is to initialize it with the common code. Then we add to the function body the input parsing step. For this we use a loop that will place \pgfmathsetmacro{tmp-cs}{\#(i)} in the function body. Afterwards we do the real definitions. This strange construct is used to normalize the input. Depending on the context in which these functions are used, the arguments might be in the internal format of PG\Fp’s fpu-library or something else that \L\Fp\ will not understand. The \pgfmathsetmacro calls will be done in a local context in which PG\Fp’s fpu-library will be activated and set up to output in a format \L\Fp\ understands.

\cs_new_protected:Npn \pgfmxfp_declare:nnn {#1#2#3}{
\@@_initialize_body:
\int_step_inline:nn {#2}
{
  \tl_put_right:Nx \l_@@_function_body_tl
  {
    \exp_not:n { \pgfmathsetmacro } \exp_not:c { \@@_arg#1 }
    { \exp_not:n {##1} }
  }
}
\@@_define_function:nnnn {#2} {#1} {#2} {#3}

(End definition for \pgfmxfp_declare:nnn. This function is documented on page 3.)

\pgfmxfp_declare_processed_args:nnnn
This works mostly like \pgfmxfp_declare:nnn, but instead of using an \int_step_inline:nn-loop this uses \clist_map_inline:nn to map over the processed arguments. Those will be stored in the function body as \pgfmathsetmacro \texttt{tmp-cs}_{i} \{ \texttt{expr}_{i}\}.
\cs_new_protected:Npn \pgfmxfp_declare_processed_args:nnnn #1#2#3#4
{
  \@@_initialize_body:
  \int_zero:N \l_@@_args_int
  \clist_map_inline:nn {#3}
  {
    \int_incr:N \l_@@_args_int
    \tl_put_right:Nx \l_@@_function_body_tl
    {
      \exp_not:n { \pgfmathsetmacro }
      \exp_not:c { \@@_arg \int_use:N \l_@@_args_int }
      { \exp_not:n {##1} }
    }
  }
  \exp_args:NV \@@_define_function:nnnn \l_@@_args_int {#1} {#2} {#4}
}

(End definition for \pgfmxfp_declare_processed_args:nnnn. This function is documented on page 3.)

2.4 Internals

\l_@@_function_body_tl
This token list will be used to build the function's top-level definition.
\tl_new:N \l_@@_function_body_tl

(End definition for \l_@@_function_body_tl. This variable is documented on page ??.)

\l_@@_args_int
In the case of \pgfmxfp_declare_processed_args:nnnn we'll have to count how many arguments the auxiliary function will take.
\int_new:N \l_@@_args_int

(End definition for \l_@@_args_int. This variable is documented on page ??.)

\@@_initialize_body:
Each function will have the same start setting up PGF's fpu.
\cs_new_protected:Npn \@@_initialize_body:
{
  \tl_set:Nn \l_@@_function_body_tl
  {
}}
First we define the internal function. Then add to the function body some code that’ll use \use:x to expand the temporary macros that store the input arguments and forward the results to the internal function.

\cs_new_protected:Npn \@@_define_function:nnnn \@@_define_function_aux:n

\@@_define_function:nnnn

\@@_define_function_aux:n

The auxiliary is just used to build the temporary macro names and prevent them from further expansion.

\cs_new:Npn \@@_define_internal_function:nnn \@@_define_internal_function_aux:n

\@@_define_internal_function:nnn

\@@_define_internal_function_aux:n

The internal function is pretty straight forward, the only difficult part is building the parameter list. For that we use some simple loop, a slow but simplistic solution.

\cs_new_protected:Npn \@@_define_internal_function_aux:n

\@@_define_internal_function_aux:n
# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

## Symbols

@@ commands:
- \l_@@_args_int .......... 37, 49, 44, 48, 51
- \@_define_function:n.nn .......... 32, 48, 60
- \@_define_function_aux:n .......... 60
- \@_define_internal_function:n.nn ................. 62, 75
- \@_define_internal_function_aux:n ................. 75
- \l_@@_function_body_tl .......... 26, 41, 50, 54, 63, 72
- \@_initialize_body: .......... 23, 36, 52

clist commands:
- \clist_map_inline:nn .......... 5, 38

cs commands:
- \cs_new:Npn ................. 74, 85
- \cs_new_protected:Npn 21, 34, 52, 60, 75
- \cs_set_protected:Npn ................. 78

D

\DeclareCurrentRelease ................. 2, 3
\DeclareRelease ......................... 1

E

exp commands:
- \exp_args:Nf ......................... 83
- \exp_args:Nln ......................... 71
- \exp_args:NV ......................... 48
- \exp_last_unbraced:Nx ................. 77
- \exp_not:N ......................... 28, 44, 67, 74
- \exp_not:n ................. 28, 29, 43, 45, 85

F

fp commands:
- \fp_eval:n ......................... 83

G

group commands:
- \group_begin: ......................... 56
- \group_end: ......................... 83

I

\IfValueTF ......................... 16

int commands:
- \int_incr:N ......................... 40
- \int_new:N ......................... 51
- \int_step_function:nN ................. 68, 81
- \int_step_inline:nn ......................... 5, 24
- \int_use:N ......................... 44
- \int_zero:N ......................... 37

N

\newcommand ................. 9, 10
\NewDocumentCommand ................. 14

P

\pgfkeys ................................ 57
\pgfmathdeclarefunction ................. 72
\pgfmathparse ......................... 83
\pgfmathsetmacro 1, 4, 5, 28, 43
\pgfmxfp commands:
- \pgfmxfp_declare:nnn ................. 3, 5, 18, 21
- \pgfmxfp_declare_processed_-_args:nnnn ................. 3, 5, 17, 34
- \pgfmxfpDate ......................... 9, 12
- \pgfmxfpdeclarefunction ................. 2, 14
- \pgfmxfpVersion ......................... 9, 13
- \providecommand ......................... 1, 2
- \ProvidesExplPackage ......................... 11

R

\RequirePackage ......................... 5, 7

T

\TeX and \LaTeX \epsilon commands:
- \@ifundefined ......................... 4
- \tlcommands:
- \tl_new:N ......................... 50
- \tl_put_right:Nn ................. 26, 41, 63
- \tl_set:Nn ......................... 54

U

\use commands:
- \use:n ......................... 6, 65
- \usepgflibrary ......................... 8