The `smart-eqn` Package: Automatic Math Symbol Styling

Ziyue "Alan" Xiang
ziyue.alan.xiang@gmail.com

CTAN: http://www.ctan.org/pkg/smart-eqn

VC: https://github.com/xziyue/smart-eqn

August 6, 2021

Contents

1 Introduction

In LaTeX typesetting, it is usually the case that one needs to use different variants of a math symbol to clarify the meanings. For example, in linear algebra literature, it is common to use boldfaced symbols to represent vectors, and use normal symbols to represent scalars, which makes equations like $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ easier to understand. However, applying these variants by typing `\mathbf`, `\mathrm` commands manually can be daunting. The `smart-eqn` package aims to provide an automatic and customizable approach for math symbol styling, which eliminates the need to enter style commands repeatedly.

2 Usage

2.1 Style Configuration

`\smesetsym{`⟨*style csname*⟩`}{`⟨*symbols*⟩`}`

The ⟨*symbols*⟩ given in the form of comma separated list will be styled using ⟨*style csname*⟩. In Example 1, the four symbols A, v, Q, and Λ will be styled with the command `\symbfup` from the `unicode-math` package, which produces boldfaced symbols. If traditional LaTeX is used, one can use the `\mathbf` command instead.

`\smeclearsym`

Clear the style configuration associated with all symbols.

2.2 Inline Math

In traditional LaTeX, inline math is typeset with `$...$`. In `smart-eqn`, we use `@...@` instead. For inline math demonstrations, see Examples 2, 3, 4, 5.

`\makeatmath`

The command changes the category code of `@` so that it can be used as the inline math environment. To revert the change, one can use `\makeatletter` or `\makeatother`.

2.3 Display Math

In order to use display math environments, we need to declare smart math environments using `\smenewenv`. The names of new math environments will be prefixed with "sme". The way of passing arguments to the math environment (e.g., `alignat`) will also be slightly different. For display math demonstrations, see Examples 6, 7, 8, 9, 10.

`\smenewenv{`⟨*env name*⟩`}`

Declares a "smart" math environment based on the traditional math environment provided in ⟨*env name*⟩. The name of the new math environment will be prefixed with "sme". For example, if one calls `\smenewenv{align}`, then the smart math environment `smealign` will be available.

Passing arguments The arguments for math environments needed to be enclosed in square brackets. See Example 10.

2.4 Raw Math Content

`\smeraw{`⟨*content*⟩`}`

The ⟨*content*⟩ of `\smeraw` will not be styled. This can be useful when one needs to style symbols manually in `smart-eqn` environments. See Examples 5, 9.

3 Examples

The following code snippet is executed before running the examples:

```
\makeatmath
\smenewenv{align}
\smenewenv{gather}
\smenewenv{alignat}
```

EXAMPLE 1 (Setting styles for symbols)

```
\smesetsym{symbfup}{A, v, Q, \Lambda}
```

EXAMPLE 2 (Inline math example 1)

```
\smesetsym{symbfup}{A, v}
The eigenvalue @\lambda@ and eigenvector @v@ of matrix @A@ satisfy @Av=\
     lambda v@.
```

The eigenvalue λ and eigenvector $\mathbf{v}$ of matrix $\mathbf{A}$ satisfy $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$.

EXAMPLE 3 (Inline math example 2)

```
\smesetsym{symbfup}{Q, \Lambda, I}
A symmetric matrix @A@ can be orthogonally diagonalized, that is, @A=Q\
     Lambda Q^T@, where @Q^TQ=I@ and @\Lambda@ is diagonal.
```

A symmetric matrix $\mathbf{A}$ can be orthogonally diagonalized, that is, $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$, where $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and $\mathbf{\Lambda}$ is diagonal.

EXAMPLE 4 (Inline math example 3)

```
\smesetsym{mathcal}{S}
For a set @S@, the power set of @S@ (denoted by @2^S@) is the set of all
     subsets of @S@.
```

For a set $\mathcal{S}$, the power set of $\mathcal{S}$ (denoted by $2^{\mathcal{S}}$) is the set of all subsets of $\mathcal{S}$.

EXAMPLE 5 (Inline math example 4)

```
I just want to juxtapose @S \smeraw{S} \smeraw{\symbfup{S}} \smeraw{\
     symbfit{S}} \smeraw{\mathrm{S}}@.
```

I just want to juxtapose $\mathcal{S}S\mathbf{S}\mathbf{S}\mathrm{S}$.

EXAMPLE 6 (Display math example 1)

```
\begin{smealign}
Av_1&=\lambda_1 v_1\\
Av_2&=\lambda_2 v_2
\end{smealign}
```

$$\mathbf{A}\mathbf{v}_1 = \lambda_1\mathbf{v}_1 \tag{1}$$
$$\mathbf{A}\mathbf{v}_2 = \lambda_2\mathbf{v}_2 \tag{2}$$

EXAMPLE 7 (Display math example 2)

```
\begin{smegather}
k_1v_1 + k_2v_2 + \cdots + k_n v_n = \symbfup{0}\\
l_1v_1 + l_2v_2 + \cdots + k_{n-1} v_{n-1} = \symbfup{1}
\end{smegather}
```

$$k_1\mathbf{v}_1 + k_2\mathbf{v}_2 + \cdots + k_n\mathbf{v}_n = \mathbf{0} \tag{3}$$
$$l_1\mathbf{v}_1 + l_2\mathbf{v}_2 + \cdots + k_{n-1}\mathbf{v}_{n-1} = \mathbf{1} \tag{4}$$

EXAMPLE 8 (Display math example 3)

```
\begin{smealign}
e^{At} = I+\sum_{n=1}^{\infty} \frac{A^n t^n}{n!}
\end{smealign}
```

$$e^{\mathbf{A}t} = \mathbf{I} + \sum_{n=1}^{\infty} \frac{\mathbf{A}^n t^n}{n!} \tag{5}$$

EXAMPLE 9 (Display math example 4)

```
\begin{smealign}
S
\smeraw{S}
\smeraw{\symbfup{S}}
\smeraw{\symbfit{S}}
\smeraw{\mathrm{S}}
\end{smealign}
```

$$\mathscr{S}S\mathbf{S}\mathrm{S} \tag{6}$$

EXAMPLE 10 (Display math example 5)

```
\smesetsym{symbfup}{x,b}
\begin{smealignat}[{5}]
&&\phantom{\Rightarrow} && (c_1^2 + \cdots + c_n^2) \lambda_n^{-2} &\leq \
     lVert \delta x \rVert^2 &&\leq (c_1^2 + \cdots + c_n^2) \lambda_1
     ^{-2}\\
&&\Rightarrow &&\lVert \delta b \rVert^2\lambda_n^{-2} &\leq \lVert \delta
     x \rVert^2 &&\leq \lVert \delta b \rVert^2\\
&&\Rightarrow &&\lVert \delta b \rVert^2\lambda_n^{-2} &\leq \lVert \delta
     x \rVert^2 &&\leq \lVert \delta b \rVert^2
\end{smealignat}
```

$$(c_1^2 + \cdots + c_n^2)\lambda_n^{-2} \leq \|\delta\mathbf{x}\|^2 \leq (c_1^2 + \cdots + c_n^2)\lambda_1^{-2} \tag{7}$$

$$\Rightarrow \qquad \|\delta\mathbf{b}\|^2\lambda_n^{-2} \leq \|\delta\mathbf{x}\|^2 \leq \|\delta\mathbf{b}\|^2 \tag{8}$$

$$\Rightarrow \qquad \|\delta\mathbf{b}\|^2\lambda_n^{-2} \leq \|\delta\mathbf{x}\|^2 \leq \|\delta\mathbf{b}\|^2 \tag{9}$$

4 Implementation

```
\RequirePackage{fancyvrb}
\RequirePackage{expl3, xparse}

\makeatletter
\ExplSyntaxOn

\tl_new:N \g_sme_op_arg_tl
\ior_new:N \g_sme_tmpa_ior
\iow_new:N \g_sme_tmpa_iow
\prop_new:N \g_sme_symbols_prop
\clist_new:N \l_sme_tmpa_clist
\tl_new:N \l_sme_tmpa_tl
\tl_new:N \l_sme_tmpb_tl
\tl_new:N \l_sme_tmpc_tl
\tl_new:N \l_sme_cur_math_tl
```

\smeDefineVerbatimEnvironment

This is a modified version of `\DefineVerbatimEnvironment` from `fancyvrb`. In this function, `\smeFV@Environment` is used insetad of `\FV@Environment`.

```
\def\smeDefineVerbatimEnvironment#1#2#3{%
  \@namedef{#1}{\smeFV@Environment{#3}{#2}}%
  \@namedef{end#1}{\@nameuse{FVE@#2}}%
}
```

\smeFV@Environment

This is a modified version of `\FV@Environment` from `fancyvrb`. In this function, `\smeFV@GetKeyValues` is used instead of `\FV@GetKeyValues`.

```
\def\smeFV@Environment#1#2{%
  \def\FV@KeyValues{#1}%
  \catcode`\^^M=\active
  \tl_gclear:N \g_sme_op_arg_tl % clear optional arguments from previous calls
  \@ifnextchar[%
    {\catcode`\^^M=5 \smeFV@GetKeyValues{\@nameuse{FVB@#2}}}%
    {\catcode`\^^M=5 \@nameuse{FVB@#2}}}
```

\smeFV@GetKeyValues

This is a modified version of FV@GetKeyValues from fancyvrb. We directly save the optional parameters captured by fancyvrb to the variable \g_sme_op_arg_tl, which can be used later.

```
\def\smeFV@GetKeyValues#1[#2]{%
  \tl_gset:Nn \g_sme_op_arg_tl {#2}#1}
```

\sme_declare_math_env:n

#1: math environment name

This function declares a "smart" math environment. It is based on the VerbatimOut environment from fancyvrb, which writes its content out to the file \jobname-sme.vrb. The content stored in the external file will be read back and processed.

```
\cs_set:Npn \sme_declare_math_env:n #1 {
    \exp_args:Nc \def{sme#1}{\smeFV@Environment{}{sme#1}}

    % this is a modified version of VerbatimOut environment which does not take any argument
    \exp_args:Nc \def{FVB@sme#1}{%
      \@bsphack
      \begingroup
        \FV@UseKeyValues
        \FV@DefineWhiteSpace
        \def\FV@Space{\space}%
        \FV@DefineTabOut
        \def\FV@ProcessLine{\immediate\write\FV@OutFile}%
        \immediate\openout\FV@OutFile \jobname-sme.vrb\relax
        \let\FV@FontScanPrep\relax
    %% DG/SR modification begin - May. 18, 1998 (to avoid problems with ligatures)
        \let\@noligs\relax
    %% DG/SR modification end
        \FV@Scan}

    \exp_args:Nc\def{FVE@sme#1}{\immediate\closeout\FV@OutFile\endgroup\@esphack
        % call the function to process the content when the environment ends
        \sme_read_process_math:n {#1}
    }

    \smeDefineVerbatimEnvironment{sme#1}{sme#1}{}
}

% define the user function
\let\smenewenv\sme_declare_math_env:n
```

\sme_read_process_math:n

#1: math environment name

Read back and process the math content stored in the external file \jobname-sme.vrb. After processing, the new content will be stored in another file \jobname-sme-in.vrb. Eventually, the new content will be fed into LaTeX using \input.

```
\cs_set:Npn \sme_read_process_math:n #1 {
    \ior_open:Nn \g_sme_tmpa_ior {\jobname-sme.vrb}
    \iow_open:Nn \g_sme_tmpa_iow {\jobname-sme-in.vrb}

    \iow_now:Nx \g_sme_tmpa_iow {\c_backslash_str begin{#1} \g_sme_op_arg_tl}
    \ior_map_inline:Nn \g_sme_tmpa_ior {
        \sme_process_math:n {##1}
        \exp_args:NNV \iow_now:Nn \g_sme_tmpa_iow \l_sme_cur_math_tl
    }
    \iow_now:Nx \g_sme_tmpa_iow {\c_backslash_str end{#1}}
    \iow_close:N \g_sme_tmpa_iow
    \ior_close:N \g_sme_tmpa_ior
    \input{\jobname-sme-in.vrb}
}
```

\smesetsym

stores the style information in \g_sme_symbols_prop

```
\newcommand{\smesetsym}[2]{
    \clist_set:Nn \l_sme_tmpa_clist {#2}
    \clist_map_inline:Nn \l_sme_tmpa_clist {
        \prop_gput:Nnn \g_sme_symbols_prop {##1} {#1}
    }
}
```

\smeclearsym

clear the style information stored in \g_sme_symbols_prop

```
\newcommand{\smeclearsym}{
    \prop_gclear:N \g_sme_symbols_prop
}
```

\smeraw

used to apply custom styles in smart-eqn environments

```
\newcommand{\smeraw}[1]{#1}
```

__sme_grouped:n

```
\cs_set:Npn \__sme_grouped:n #1 {
    \exp_not:n {#1}
}
\cs_generate_variant:Nn \__sme_grouped:n {e}
\cs_generate_variant:Nn \__sme_grouped:n {V}
```

\sme_construct_bm:Nn

#1: result variable

#2: current token list

This function is the core of automatic math styling. It examines each token in ⟨*current token list*⟩ and determines if it needs special styling. The function runs recursively and stores the result in ⟨*result variable*⟩.

```
\cs_set:Npn \sme_construct_bm:Nn #1#2 {
    \tl_if_head_is_group:nTF {#2} {
        % if head is group, apply the algorithm recursively
        %\tl_put_right:Nx #1 {{\tl_head:n {#2}}}
        \tl_put_right:NV #1 \c_left_brace_str
        \exp_args:NNx \sme_construct_bm:Nn #1 {\tl_head:n {#2}}
        \tl_put_right:NV #1 \c_right_brace_str
        \tl_if_empty:nF {#2} {\exp_args:NNx \sme_construct_bm:Nn #1 {\tl_tail:n {#2}}}
    }{
        \tl_if_head_is_space:nTF {#2} {
            % ignore spaces
            \exp_args:NNx \sme_construct_bm:Nn #1 {\tl_trim_spaces:n {#2}}
        } {
            \tl_if_head_eq_meaning:nNTF {#2} \smeraw {
                % for \smeraw, ignore the next group
                \tl_put_right:Nn #1 {\smeraw}
                \tl_set:Nx \l_sme_tmpa_tl {\tl_tail:n {#2}}
                \tl_put_right:Nx #1 {\tl_head:N \l_sme_tmpa_tl}
                \tl_set:Nx \l_sme_tmpa_tl {\tl_tail:N \l_sme_tmpa_tl}
                \exp_args:NNV \sme_construct_bm:Nn #1 \l_sme_tmpa_tl
            } {
                % common case
                % extract head
             \exp_args:NNx \prop_get:NnNTF \g_sme_symbols_prop {\tl_head:n {#2}} \l_sme_tmpc_tl
{
```

```
                    % this symbol needs to be styled
                    \tl_set:No \l_sme_tmpa_tl {\csname\l_sme_tmpc_tl\endcsname}
                    \tl_put_right:Nx \l_sme_tmpa_tl {{\__sme_grouped:e {\tl_head:n {#2}}}}
                    \tl_put_right:Nx #1 {{\__sme_grouped:V \l_sme_tmpa_tl}}
                } {
                    % otherwise, use the original symbol
                    \tl_put_right:Nx #1 {\tl_head:n {#2}}
                }
                \tl_if_empty:nF {#2} {\exp_args:NNx \sme_construct_bm:Nn #1 {\tl_tail:n {#2}}}
            }
        }
    }
}
```

`\sme_process_math:n`

`#1`: inline content

used by inline math environment to process the content

```
\cs_set:Npn \sme_process_math:n #1 {
    \tl_clear:N \l_sme_cur_math_tl
    \sme_construct_bm:Nn \l_sme_cur_math_tl {#1}
}
```

`\makeatmath`

changes the catcode and definition of `@` so that we can use it for math typesetting

```
\begingroup
\catcode`@=\active
\gdef\makeatmath{%    note the global \gdef
  \catcode`@=\active
  \def@##1@{
    \sme_process_math:n{##1}
    $\exp_args:NnV \tl_rescan:nn {} \l_sme_cur_math_tl$
  }%
}
\endgroup
\makeatother
\ExplSyntaxOff
```