The nccmath package[*]

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2006/01/20

The package extends the `amsmath` package adding some math commands from NCC-LaTeX. It also improves spacing control before display equations and fixes a bug of ignoring the `\displaybreak` in the `amsmath` version of the `equation` environment. All options are passed to the `amsmath` package.

Contents

1 Improvement to the amsmath

`eqnarray` In the `amsmath` package, the `eqnarray` environment leaves unchanged because alternative $\mathcal{AMS}$ environments exist. We redefine the `eqnarray` to work in the $\mathcal{AMS}$ style. The following improvements are done in it: an equation tag is prepared by the same manner as in $\mathcal{AMS}$ display formulas (`\tag` and `\tag*` are allowed); the `\displaybreak` command is allowed; the intercolumn distance is reduced to

---

[*] This file has version number v1.3, last revised 2006/01/20.

the distance between ordinary and relational math symbols; and the center field is prepared in the `\displaystyle` (the original version uses `\textstyle` here).

`\intertext` The `\intertext` command is improved here. It now has an optional parameter:

```
\intertext[⟨distance⟩]{⟨text⟩}
```

The ⟨*distance*⟩ parameter specifies a vertical space inserted before and after the text. If it is omitted, standard TEX's skips are inserted.

The following changes are made in display equations:

- The `\displaybreak` command now works within the `equation` environment (it is ignored in the `amsmath`);
- The $\mathcal{AMS}$ and LATEX display equations prepared in the vertical mode do not produce now an empty extra line before. Moreover, if a minipage starts from a display formula, the vertical skip before is suppressed.

2 Extra Macros

`fleqn` `ceqn` The following environments allow change the horizontal alignment of formulas inside them:

```
\begin{fleqn}[⟨margin⟩] ... \end{fleqn}
\begin{ceqn} ... \end{ceqn}
```

The `fleqn` environment prepares inner display equations in the flush left style. The ⟨*margin*⟩ parameter specifies the left margin value. If it is omitted, zero value is used. The `ceqn` environment prepares inner display equations in the centered style. They have no effect on formulas prepared with the low-level TEX command `$$`.

`darray` The `darray` environment produces an array of formulas in the `\displaystyle`. The distance between formulas is enlarged in just the same manner as in other multiline display equations. The `darray` environment has the same syntax as the `array`:

```
\begin{darray}[⟨pos⟩]{⟨columns⟩}
⟨body⟩
\end{darray}
```

The ⟨*pos*⟩ argument describes the vertical alignment of the array box (`t`, `b`, or `c`; default is `c`). The use of column specifications in the ⟨*columns*⟩ argument is restricted in comparison with `array`: it can contain the `l`, `c`, and `r` specifiers, `*` and `@` commands. The intercolumn separation is smaller than in the `array`: it is reduced to the distance between ordinary and relational math symbols. As in the `amsmath` package, the thin skip is inserted before `darray`. Skips before the first and after the last column of `darray` are not inserted. To insert them manually, use `@{...}` in the ⟨*columns*⟩ argument.

The `darray` environment is implemented independently on the `array` environment to avoid conflicts with the `array` package.

`\dmulticolumn` `\dmulticolumn{`⟨*count*⟩`}{`⟨*preamble*⟩`}{`⟨*formula*⟩`}` is used in `darray` instead of `\multicolumn`.

`\useshortskip` In TeX, two types of skips above display formulas are used: the normal skip defined in the `\abovedisplayskip` register and the short skip defined in the `\abovedisplayshortskip` register. When a display formula is typed out, TeX decides what skip to insert depending on the width of formula, its style (centered or flushed left, numbered left or right), and the width of the rest of text in the last line of the previous paragraph. But this algorithm works for ordinary formulas only. It does not work in multiline formulas prepared with `\halign` command. So, a manual replacement of the normal skip to the short skip is required in some cases. To provides this, the `\useshortskip` command is introduced. It forces the use of short skip in the next display formula but it has no effect on formulas prepared with the low-level TeX command `$$`.

`\nr` The vertical distance between lines of miltiline equations is frequently smaller than necessary. To increase it, the extra distance can be used as the optional parameter of the `\\[`⟨*dist*⟩`]` command. In most cases, it is enough to increase the distance on `0.5ex`. We introduce the `\nr` command here that is equivalent to the `\\[0.5ex]`. Its full sintax is just the same as for the `\\` command:

```
\nr*[⟨dist⟩]
```

This command can be used everywhere the command `\\` is allowed.

`\mrel` The `\mrel{`⟨*column*⟩`}` command composes a new math relation symbol from a one-column stack of math formulas described in the ⟨*column*⟩ parameter. For example, the command `$\mrel{<\\[-.7ex]>}$` produces $\lessgtr$.

`\underrel` The `\underrel{`⟨*base*⟩`}{`⟨*bottom*⟩`}` command is a twin to the `\overrel` command. For example, the command `$A\underrel{\longrightarrow}{x\to 0}B$` produces $A \underset{x\to 0}{\longrightarrow} B$.

3 Medium-Size Math Commands

Since version 1.2, a collection of medium-size math commands is introduced.

`\medmath` The `\medmath{`⟨*formula*⟩`}` command decreases a size of formula in 1.2 times and prepares it in the display style. An example:

```
$\medmath{\cfrac{1}{\sqrt 2 +\cfrac{1}{\sqrt 2 +\dotsb}}}$
\quad $\cfrac{1}{\sqrt 2 +\cfrac{1}{\sqrt 2 +\dotsb}}$
```

It produces:

$$\cfrac{1}{\sqrt 2 +\cfrac{1}{\sqrt 2 +\dotsb}} \quad \cfrac{1}{\sqrt 2 +\cfrac{1}{\sqrt 2 +\dotsb}}$$

`\medop` The `\medop{`⟨*operator*⟩`}` command prepares a medium-size operator with the required preference for limits. It can be use with `\sum` and others variable-size commands except integrals. An example:

```
$\sum_{i=1}^n \medop\sum_{i=1}^n \displaystyle
\sum\nolimits_{i=1}^n$\quad $\sum\limits_{i=1}^n
\displaystyle \medop\sum_{i=1}^n \sum_{i=1}^n$
```

It produces:

$$\textstyle\sum_{i=1}^n \sum_{i=1}^n \displaystyle\sum\nolimits_{i=1}^n \quad \textstyle\sum\limits_{i=1}^n \sum\limits_{i=1}^n \displaystyle\sum_{i=1}^n$$

`\medint` The `\medint{`⟨*operator*⟩`}` command prepares a medium-size integral with required preference for limits. It can be use with `\int`-family of commands and `\oint` command. An example:

```
$\int_a^b \medint\int_a^b \displaystyle\int_a^b$\quad
$\int\limits_a^b \medint\int_a^b\limits \displaystyle
\int_a\limits^b$\quad $\iint_a^b \medint\iiint_a^b
\displaystyle\iiiint_a^b$\quad $\iint\limits_X^Y
\medint\iiint_X\limits^Y \displaystyle \iiiint_X^Y\limits$
\quad $\medint\idotsint_X\limits \medint\oint_X^Y$
```

It produces:

$$\textstyle\int_a^b \int_a^b \displaystyle\int_a^b \quad \textstyle\int\limits_a^b \int\limits_a^b \displaystyle\int\limits_a^b \quad \textstyle\iint_a^b \iiint_a^b \displaystyle\iiiint_a^b \quad \textstyle\iint\limits_X^Y \iiint\limits_X^Y \displaystyle\iiiint\limits_X^Y \quad \int\cdots\int\limits_X \oint_X^Y$$

By the way, the original limits recognizing in `amsmath` multi-integrals is very restrictive: it allows only one `\limits`-like command right after the multi-integral. In this package, the recognizing is improved to work as TeX's one.

`\medintcorr` The `\medintcorr{`⟨*length*⟩`}` command specifies the value of italic correction for medium integrals. It controls a positioning indices in medium integrals and in multi-integrals. Its default value is `0.5em`.

`\mfrac` `\mbinom` Based on the medium size formulas, the `\mfrac` and `\mbinom` commands are introduced. They are similar to `\frac` and `\binom`. An example:

```
$\frac {x+y}{a-b} \mfrac {x+y}{a-b} \dfrac {x+y}{a-b}$\quad
$\binom {n}{k} \mbinom {n}{k} \dbinom {n}{k}$
```

It produces:

$$\tfrac{x+y}{a-b}\,\frac{x+y}{a-b}\,\dfrac{x+y}{a-b} \quad \tbinom{n}{k}\binom{n}{k}\dbinom{n}{k}$$

`medsize` `mmatrix` The `medsize` environment is introduced to prepare formulas and arrays in the medium size. It reduces the `\arraycolsep` value by 0.8 times. Basing on it, the `mmatrix` environment is introduced. It is specified as follows:

```
\begin{mmatrix} ... \end{mmatrix}  ≡
\begin{medsize}\begin{matrix} ... \end{matrix}\end{medsize}
```

An example:

```
$\bigl(\begin{smallmatrix} a&b\\c&d\end{smallmatrix}\bigr)$
$\Bigl(\begin{mmatrix} a&b\\c&d\end{mmatrix}\Bigr)$
$\begin{pmatrix}a&b\\c&d\end{pmatrix}$
```

It produces:

$$\bigl(\begin{smallmatrix} a&b\\c&d\end{smallmatrix}\bigr)\ \Bigl(\begin{matrix} a&b\\c&d\end{matrix}\Bigr)\ \begin{pmatrix}a&b\\c&d\end{pmatrix}$$

`mediummath` Finally, the `mediummath` option allows prepare all variable-size math elements in medium size. It redefines `\frac`, `\binom` and all math operators to the medium size. For `\frac` and `\binom`, the medium size is applied in the display and text styles. The `\dfrac`, `\tfrac`, `\dbinom`, and `\tbinom` commands have the old meaning.

4 NCC-LaTeX Equivalents to Display Formulas

The following NCC-LaTeX equivalents are provided with this package:

`\eq{`⟨*formula*⟩`}` = `\begin{equation}` ⟨*formula*⟩ `\end{equation}`.

`\eq*{`⟨*formula*⟩`}` = `\begin{equation*}` ⟨*formula*⟩ `\end{equation*}`.

`\eqs{`⟨*formulas*⟩`}` = `\begin{eqnarray}` ⟨*formulas*⟩ `\end{eqnarray}`.

`\eqs*{`⟨*formulas*⟩`}` = `\begin{eqnarray*}` ⟨*formulas*⟩ `\end{eqnarray*}`.

`\eqalign{`⟨*formulas*⟩`}` = `\begin{equation} \begin{darray}{rcl}` ⟨*formulas*⟩ `\end{darray} \end{equation}`.

`\eqalign*{`⟨*formulas*⟩`}` = `\begin{equation*} \begin{darray}{rcl}` ⟨*formulas*⟩ `\end{darray} \end{equation*}`.

The `\eqs` and `\eqs*` commands have an optional parameter specifying a distance between columns. For example, in the command

```
\eqs[0mm]{&& -\Delta u = f, \\ && u|_\Gamma = 0,}
```

the intercolumn distance is removed because only the 3rd column is used. The `eqnarray` environment has no optional parameter.

The `\eqalign` and `\eqalign*` commands also have an optional parameter. Its meaning is the column specification parameter: `\eqalign{`⟨*formulas*⟩`}` = `\eqalign[rcl]{`⟨*formulas*⟩`}`.

5 The Implementation

At first we load the `amsmath` package and pass all options to it except the `mediummath` option.

```
⟨*package⟩
```

```
\DeclareOption{mediummath}{\newcommand\NCC@op{}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{amsmath}}
\ProcessOptions\relax
\RequirePackage{amsmath}[2000/07/18]
```

5.1 Kernel

`\NCC@cr` Simplified version of `\\` used in some commands here. The low level command `\NCC@cr@@@{`⟨*skip*⟩`}` is defined if necessary to `\NCC@aligncr` or to something else. The `\new@ifnextchar` commands from the $\mathcal{AMS}$ does the same as `\@ifnextchar`, but disallows spaces before the tested symbol.

```
\newif\ifNCC@star
\def\NCC@cr{\relax\iffalse{\fi\ifnum0=`}\fi
  \@ifstar{\global\NCC@startrue\NCC@cr@}{\global\NCC@starfalse\NCC@cr@}%
}
\def\NCC@cr@{\new@ifnextchar[\NCC@cr@@{\NCC@cr@@[\z@]}}
\def\NCC@cr@@[#1]{\ifnum0=`{\fi \iffalse}\fi\NCC@cr@@@{#1}}
\def\NCC@aligncr#1{\cr\noalign{\vskip #1\relax}}
```

`\NCC@default@cr` This command sets defaults for the `\\` command.

```
\def\NCC@default@cr{\let\\\NCC@cr \let\NCC@cr@@@\NCC@aligncr}
```

`\nr` The `\nr` command has just the same syntax as `\\` but adds 0.5ex extra vertical space between lines. It can work anywhere the `\\` command is allowed. We temporary change in it the value of `\NCC@cr@@@` to `\NCC@nr` and restore it later.

```
\newcommand{\nr}{%
  \let\NCC@temp\NCC@cr@@@
  \let\NCC@cr@@@\NCC@nr
  \NCC@cr
}
\def\NCC@nr#1{%
  \let\NCC@cr@@@\NCC@temp
  \setlength\@tempskipa{#1}\advance\@tempskipa .5ex
  \ifNCC@star
    \edef\@tempa{\noexpand\\*[\the\@tempskipa]}%
  \else
    \edef\@tempa{\noexpand\\[\the\@tempskipa]}%
  \fi
  \@tempa
}
```

5.2 Additional Math Commands

`\mrel` The `\mrel{`⟨*column*⟩`}` command composes a new math relation and vertically centers it with respect to the math line.

```
\newcommand{\mrel}{\mathpalette\NCC@rel}
\def\NCC@rel#1#2{\mathrel{\vcenter{\NCC@default@cr
  \offinterlineskip \ialign{\hfil$\m@th#1##$\hfil\cr#2\crcr}}}}
```

`\underrel` The `\underrel{`⟨*base*⟩`}{`⟨*bottom*⟩`}` command is a twin to `\overrel`.

```
\newcommand{\underrel}[2]{\mathrel{\mathop{#1}\limits_{#2}}}
```

5.3 Medium-Size Math Commands

`\NCC@select@msize` The `\NCC@select@msize` command prepares dimensions for medium-size math:

- In `\NCC@fracrulewidth` — a rule width in fractions;
- In `@tempdima` — a raising value; and
- In `@tempdimb` — a font size to be used in medium fractions and matrices.

```
\newdimen\NCC@fracrulewidth
\def\NCC@select@msize{\relax
```

`\@tempdima` contains the current font size

```
  \@tempdima \f@size\p@
```

Calculate in `\@tempdimb` a text font size in medium fraction

```
  \ifdim\@tempdima>11.5\p@
    \@tempdimb .83\@tempdima
  \else
    \@tempdimb .8\@tempdima
    \ifdim\@tempdimb<5\p@ \@tempdimb 5\p@\fi
  \fi
```

Calculate in `\NCC@fracrulewidth` the rule width and in `\@tempdima` — the raising value

```
  \NCC@fracrulewidth .04\@tempdima
  \@tempdima 1.25\NCC@fracrulewidth
  \ifdim\NCC@fracrulewidth>.45\p@ \else
    \ifdim\NCC@fracrulewidth>.34\p@ \NCC@fracrulewidth .4\p@
    \else \NCC@fracrulewidth .3\p@
    \fi
  \fi
}
```

`\NCC@innerfrac`
`\NCC@innerbinom`
The `\NCC@innerfrac{`⟨*style*⟩`}` prepares a fraction with a special width in the given style:

```
\def\NCC@innerfrac#1{\genfrac{}{}\NCC@fracrulewidth{#1}}
```

`\NCC@prepare@msize` Select a font by rounding its pt-size to the nearest integer and redefine fractions to have the given rule width. The `\binom` command is redefined also to its original value because it can be changed when the `mediummath` option is applied.

```
\def\NCC@prepare@msize{%
  \@tempdima 1.2\@tempdimb
  \advance\@tempdimb .5\p@
  \edef\@tempa{\strip@pt\@tempdimb}%
  \expandafter\NCC@floor\expandafter\@tempa\@tempa.\@nil
  \fontsize\@tempa\@tempdima\selectfont
  \def\frac{\protect\NCC@innerfrac{}}%
  \def\dfrac{\NCC@innerfrac\z@}%
  \def\tfrac{\NCC@innerfrac\@ne}%
  \def\binom{\protect\genfrac()\z@{}}%
}
\def\NCC@floor#1#2.#3\@nil{\def#1{#2}}
```

`\NCC@op@prepare`

`\NCC@op@prepare{`⟨*integral*⟩`}` command prepares an integral. It looks forward, extracts indices and limits-change commands, and puts the integral with required kerning of indices. The `\NCC@op@print` driver is a command to print the integral. Its default value is `\NCC@op@printm`. The driver uses the following hooks: `\NCC@op` contains an integral command, `\NCC@op@lim` contains the selected limits-style, `\NCC@op@sb` contains a subscript, `\NCC@op@sp` contains a superscript, `NCC@op@kern` contains the kerning value for medium-size integrals. If subscript or superscript is omitted, the corresponding hook is equal to `\relax`.

```
\DeclareRobustCommand*\NCC@op@prepare[1]{%
  \def\NCC@op{#1}%
  \let\NCC@op@print\NCC@op@printm
  \NCC@op@prepare@
}
\def\NCC@op@prepare@{%
  \let\NCC@op@lim\ilimits@
  \let\NCC@op@sp\relax
  \let\NCC@op@sb\relax
  \NCC@op@next
}
\def\NCC@op@next{\futurelet\@let@token\NCC@op@getnext}
```

Test the next token and get it if necessary:

```
\def\NCC@op@getnext{%
  \let\@tempa\NCC@op@skip
  \ifx\@let@token\limits
   \let\NCC@op@lim\limits \else
    \ifx\@let@token\nolimits
     \let\NCC@op@lim\nolimits \else
      \ifx\@let@token\displaylimits
       \let\NCC@op@lim\displaylimits \else
        \ifx\@let@token\sp
         \NCC@op@test\NCC@op@sp
         \def\@tempa{\NCC@op@get\NCC@op@sp}\else
          \ifx\@let@token\sb
           \NCC@op@test\NCC@op@sb
           \def\@tempa{\NCC@op@get\NCC@op@sb}\else
            \ifx\@let@token\@sptoken
             \let\@tempa\NCC@op@skipsp \else
             \let\@tempa\NCC@op@print
            \fi
          \fi
        \fi
      \fi
    \fi
  \fi
  \@tempa
}
```

Skip `\limits`-like token:

```
\def\NCC@op@skip#1{\NCC@op@next}
```

Skip a space token. A space token is skipped within `\@ifnextchar` before comparing it with the first parameter. So, it does not important what char to test for:

```
\def\NCC@op@skipsp{%
  \@ifnextchar0{\NCC@op@next}{\NCC@op@next}%
}
```

Test subscript or superscript to be already defined:

```
\def\NCC@op@test#1{%
  \ifx#1\relax \else
    \PackageError{nccmath}{Double index in math operator}{}
  \fi
}
```

Get a subscript or superscript:

```
\def\NCC@op@get#1#2#3{\def#1{#3}\NCC@op@next}
```

`\NCC@op@printm` Driver for printing the medium-size integral with indices:

```
\def\NCC@op@printm{%
  \ifx\NCC@op@lim\nolimits \NCC@op@printm@\@ne \else
    \ifx\NCC@op@lim\limits \NCC@op@printm@\z@ \else
      \mathchoice{\displaystyle\NCC@op@printm@\z@}%
                 {\textstyle\NCC@op@printm@\@ne}%
                 {\scriptstyle\NCC@op@printm@\@ne}%
                 {\scriptscriptstyle\NCC@op@printm@\@ne}%
    \fi
  \fi
}
\def\NCC@op@printm@{\NCC@op@print@\NCC@op\NCC@op@kern}
```

`\NCC@op@print@` `\NCC@op@print@{`⟨*integral*⟩`}{`⟨*kern*⟩`}{`⟨*level*⟩`}` command prints an ⟨*integral*⟩ using the specified ⟨*kern*⟩ in indices. If ⟨*level*⟩ = 0 use `\limits` else use `\nolimits`.

```
\def\NCC@op@print@#1#2#3{\mathop{#1}%
  \setlength\@tempdima{#2}%
  \@tempswatrue
  \ifx\NCC@op@sb\relax \else \ifnum#3>\z@ \@tempswafalse \fi \fi
  \ifx\NCC@op@sp\relax \else \ifnum#3>\z@ \@tempswafalse \fi \fi
  \edef\@tempa{%
    \ifnum#3=\z@ \noexpand\limits \else \noexpand\nolimits \fi
    \ifx\NCC@op@sb\relax \else
      \noexpand\sb{%
        \ifnum#3=\z@ \kern -\@tempdima\else \kern -.8\@tempdima \fi
        \noexpand\NCC@op@sb}%
    \fi
    \ifx\NCC@op@sp\relax \else
      \noexpand\sp{\ifnum#3=\z@ \kern \@tempdima\fi
        \noexpand\NCC@op@sp}%
    \fi
    \if@tempswa \kern -.2\@tempdima \fi
  }%
  \@tempa
}
```

`\medmath` The `\medmath{`⟨*formula*⟩`}` prepares a medium-size formula in display style:

```
\DeclareRobustCommand*\medmath[1]{\NCC@select@msize
  \mathord{\raise\@tempdima\hbox{\NCC@prepare@msize
    $\displaystyle#1$}}%
}
```

`\medop` The `\medop{`⟨*operator*⟩`}` prepares an operator in the medium size:

```
\newcommand*\medop[1]{\DOTSB\mathop{\medmath{#1}}\slimits@}
```

`\medintcorr` The `\medintcorr{`⟨*length*⟩`}` specifies an italic correction for a medium integral:

```
\newcommand*\medintcorr[1]{\def\NCC@op@kern{#1}}
\medintcorr{.5em}
```

`\medint` The `\medint{`⟨*integral*⟩`}` command prepares a medium integral:

```
\newcommand*\medint[1]{\DOTSI\NCC@op@prepare{\medmath{#1}}}
```

`\mfac` The `\mfrac{`⟨*numerator*⟩`}{`⟨*denominator*⟩`}` prepares a medium-size fraction:

```
\DeclareRobustCommand*\mfrac[2]{\medmath{\frac{#1}{#2}}}
```

`\mbinom` The `\mbinom{`⟨*numerator*⟩`}{`⟨*denominator*⟩`}` prepares a medium-size binomial expression:

```
\DeclareRobustCommand*\mbinom[2]{%
  \Bigl(\medmath{\genfrac{}{}{\z@}{}{#1}{#2}}\Bigr)%
}
```

`medsize` The `medsize` environment is useful for preparing medium-size arrays:

```
\newenvironment{medsize}{\NCC@select@msize
  \mathord\bgroup
    \raise\@tempdima\hbox\bgroup\NCC@prepare@msize
      \arraycolsep .8\arraycolsep $}{$\egroup\egroup}
```

`mmatrix` The `mmatrix` environment prepares a medium-size matrix:

```
\newenvironment{mmatrix}{\medsize\begin{matrix}}{\end{matrix}\endmedsize}
```

5.4 Patches to amsmath

`\MultiIntegral` Improve the `\MultiIntegral` kerning method on the base of `\NCC@op@prepare@` hook. The original method from `amsmath` works bad if a multi-integral is an argument of the `\medint` command.

```
\renewcommand*{\MultiIntegral}[1]{%
  \edef\NCC@op{\noexpand\intop
    \ifnum#1=\z@\noexpand\intdots@\else\noexpand\intkern@\fi
    \ifnum#1>\tw@\noexpand\intop\noexpand\intkern@\fi
    \ifnum#1>\thr@@\noexpand\intop\noexpand\intkern@\fi
    \noexpand\intop
  }%
  \let\NCC@op@print\NCC@op@printd
  \NCC@op@prepare@
}
\def\NCC@op@printd{%
  \setlength\@tempdima{\NCC@op@kern}%
  \ifx\NCC@op@lim\nolimits \@tempcnta\@ne \else
    \ifx\NCC@op@lim\limits \@tempcnta\z@ \else
      \@tempcnta\m@ne
    \fi
  \fi
  \mathchoice{\NCC@op@printd@{\displaystyle}{1.2\@tempdima}}%
             {\NCC@op@printd@{\textstyle}{.8\@tempdima}}%
             {\NCC@op@printd@{\scriptstyle}{.8\@tempdima}}%
             {\NCC@op@printd@{\scriptscriptstyle}{.8\@tempdima}}%
}
\def\NCC@op@printd@#1#2{#1%
  \ifnum\@tempcnta>\m@ne
    \NCC@op@print@{\hbox{$#1\NCC@op$}}{#2}\@tempcnta
  \else
    \ifx#1\displaystyle
      \NCC@op@print@{\hbox{$#1\NCC@op$}}{#2}\z@
    \else
      \NCC@op@print@{\hbox{$#1\NCC@op$}}{#2}\@ne
    \fi
  \fi
}
```

`\endmathdisplay@a` Fix the bug in the `\endmathdisplay@a` command from the `amsmath` package. The `\displaybreak` has no effect in it if a tag is specified. This is because the change of `\postdisplaypenalty` is done after the `\eqno` command. But the rest of display formula after `\eqno` up to the `$$` command belongs to the tag. It is prepared in the horizontal mode and the mentioned penalty is ignored. Fixed version of this command at first changes the `\postdisplaypenalty` and after that prints a tag.

To be sure, that the required command does not fixed yet, we prepare its bug version in the `\@tempa` command

```
\def\@tempa{%
  \if@eqnsw \gdef\df@tag{\tagform@\theequation}\fi
  \if@fleqn \@xp\endmathdisplay@fleqn
  \else \ifx\df@tag\@empty \else \veqno \alt@tag \df@tag \fi
    \ifx\df@label\@empty \else \@xp\ltx@label\@xp{\df@label}\fi
  \fi
  \ifnum\dspbrk@lvl>\m@ne
    \postdisplaypenalty -\@getpen\dspbrk@lvl
    \global\dspbrk@lvl\m@ne
  \fi
}
```

and compare it with the current value of `\endmathdisplay@a`. If they are identic, we fix the last command. Otherwise, print a warning and do nothing.

```
\ifx\@tempa\endmathdisplay@a
  \def\endmathdisplay@a{%
    \ifnum\dspbrk@lvl>\m@ne
      \postdisplaypenalty -\@getpen\dspbrk@lvl
      \global\dspbrk@lvl\m@ne
    \fi
    \if@eqnsw \gdef\df@tag{\tagform@\theequation}\fi
    \if@fleqn \@xp\endmathdisplay@fleqn
    \else \ifx\df@tag\@empty \else \veqno \alt@tag \df@tag \fi
      \ifx\df@label\@empty \else \@xp\ltx@label\@xp{\df@label}\fi
    \fi
  }
\else
  \PackageWarning{nccmath}%
    {The \string\endmathdisplay@a\ command differs from\MessageBreak
     waited value in this version of amsmath package.\MessageBreak
     We don’t fix it!}
\fi
```

`\intertext`

Redefine $\mathcal{AMS}$'s `\intertext{`⟨*text*⟩`}` to `\intertext[`⟨*skip*⟩`]{`⟨*text*⟩`}`. Optional ⟨*skip*⟩ means the vertical space inserted below and after the text. If it is omitted, the default `\belowdisplayskip` and `\abovedisplayskip` spaces are inserted.

We need to redefine its default value used out of display equations:

```
\renewcommand*{\intertext}[1][]{\@amsmath@err{\Invalid@@\intertext}\@eha}
```

and also must redefine the `\intertext@` hook that changes the value of `\intertext` within display equations. Its new definition differs from the original one in the conditional inserting of skips before and after the text. The optional parameter is scanned inside the `\noalign` command. We use the ordinary trick with the `\ifnum0` to close the open brace in the next macro.

```
\def\intertext@{%
  \def\intertext{%
    \ifvmode\else\\\@empty\fi
    \noalign{\ifnum0=`}\fi
      \@ifnextchar[{\NCC@intertext}{\NCC@intertext[]}%
  }%
}
\def\NCC@intertext[#1]#2{%
```

```
  \penalty\postdisplaypenalty
  \@ifempty{#1}{\vskip\belowdisplayskip}{\vskip#1\relax}%
  \vbox{\normalbaselines
    \ifdim\linewidth=\columnwidth
    \else \parshape\@ne \@totalleftmargin \linewidth
    \fi
    \noindent#2\par}%
  \penalty\predisplaypenalty
  \@ifempty{#1}{\vskip\abovedisplayskip}{\vskip#1\relax}%
  \ifnum0=`{\fi}%
}
```

`\useshortskip` The `\useshortskip` command changes an above skip for nearest display formula to `\abovedisplayshortskip`. Really, it sets the value of inner if-macro to true and the actual changes are applied in the `\NCC@ignorepar` hook.

```
\newif\ifNCC@shortskip \NCC@shortskipfalse
\newcommand{\useshortskip}{\global\NCC@shortskiptrue}
```

`\NCC@ignorepar` This command removes extra vertical space before display formula if it starts from a new paragraph and changes the before-skip to `\abovedisplayshortskip` if the `\useshortskip` command was applied.

```
\def\NCC@ignorepar{\relax
  \ifNCC@shortskip
    \abovedisplayskip\abovedisplayshortskip
    \global\NCC@shortskipfalse
  \fi
  \ifmmode \else \ifvmode
```

If a display equation starts in the vertical mode, we insert the vertical space with the `\addvspace` (this space will be ignored at the beginning of minipage) and set above display skips to zero. The below display skips are made equal. Then we put the `\noindent` command that prevents insertion an empty paragraph.

```
    \addvspace{\abovedisplayskip}%
    \abovedisplayskip\z@skip
    \abovedisplayshortskip\z@skip
    \belowdisplayshortskip\belowdisplayskip
    \noindent
  \fi\fi
}
```

Now we insert the `\NCC@ignorepar` command at the beginning of all LaTeX and AMS-LaTeX display equations except `eqnarray`. We need to correct four AMS commands only:

```
\let\NCC@startgather\start@gather
\let\NCC@startalign\start@align
\let\NCC@startmultline\start@multline
\let\NCC@startdisplay\mathdisplay
\def\start@gather{\NCC@ignorepar\NCC@startgather}
\def\start@align{\ifingather@\else\NCC@ignorepar\fi\NCC@startalign}
```

```
\def\start@multline{\NCC@ignorepar\NCC@startmultline}
\def\mathdisplay{\NCC@ignorepar\NCC@startdisplay}
```

5.5 The darray Environment

`darray` The implementation of `darray` is a hybrid of the `\start@aligned` command from the `amsmath` package and the `\array` command.

```
\newenvironment{darray}[2][c]{%
  \null\,%
  \if #1t\vtop \else \if#1b \vbox \else \vcenter \fi \fi
  \bgroup
    \NCC@default@cr
    \spread@equation
    \NCC@mkpream{#2}%
    \edef\@preamble{\ialign \bgroup \strut@ \@preamble \tabskip\z@skip \cr}%
    \let\par\@empty \let\@sharp##%
    \set@typeset@protect
    \tabskip\z@skip
    \@preamble
}{%
    \crcr\egroup\egroup
}
```

`\dmulticolumn` To produce multi-columns in `darray`, the `\dmulticolumn` command is used.

```
\newcommand\dmulticolumn[3]{\multispan{#1}%
  \begingroup
    \NCC@mkpream{#2}%
    \def\@sharp{#3}\set@typeset@protect
    \@preamble
  \endgroup
  \ignorespaces
}
```

`\NCC@mkpream` The `darray` environment is independent from `array` to avoid conflicts with packages customizing the `array` environment. So, we need to implement an independent preamble maker.

The following classes can appear in the preamble:

0 `lcr`
1 `@`-argument
2 `@`

The implementation of preamble maker is very similar to the LaTeX's version.

```
\def\NCC@mkpream#1{%
  \@lastchclass\@ne \@firstamptrue
```

Specify the default distance between columns in the `\alignsep@` register from `amsmath`.

```
  \settowidth\alignsep@{$\m@th\mskip\thickmuskip$}%
```

```
 \let\@sharp\relax
 \let\@preamble\@empty
```

The `\@xexpast` command expands the argument replacing all instances of `*{`⟨*N*⟩`}{`⟨*string*⟩`}` by ⟨*N*⟩ copies of ⟨*string*⟩. The result is saved in the `\reserved@a` macro. But this command is let to `\relax` in the `array` package. So, we use its original definition prepared in the `\NCC@xexpast` macro to avoid conflicts with other packages.

```
 \let\protect\@unexpandable@protect
 \NCC@xexpast #1*0x\@@
```

Now we make the preamble collecting it in the `\@preamble` hook. The code is very similar to the LaTeX's `\@mkpream` command.

```
 \expandafter \@tfor \expandafter \@nextchar
  \expandafter :\expandafter =\reserved@a \do
 {\@chclass
  \ifnum \@lastchclass=\tw@ \@ne \else
   \z@
   \edef\@nextchar{\expandafter\string\@nextchar}%
   \if \@nextchar @\@chclass \tw@ \else
    \@chnum
    \if \@nextchar c\z@ \else
     \if \@nextchar l\@ne \else
      \if \@nextchar r\tw@ \else
       \z@ \@preamerr \z@
      \fi
     \fi
    \fi
   \fi
  \fi
  \ifcase \@chclass
   \ifnum \@lastchclass=\z@ \@addtopreamble{\hskip \alignsep@}\fi
   \@addamp
   \@addtopreamble{%
     \ifcase \@chnum \hfil$\displaystyle{\@sharp}$\hfil
     \or             $\displaystyle{\@sharp}$\hfil
     \or             \hfil$\displaystyle{\@sharp}$%
     \fi
   }%
  \or
   \@addtopreamble{$\@nextchar$}%
  \fi
  \@lastchclass\@chclass
 }%
 \ifnum\@lastchclass=\tw@ \@preamerr\@ne \fi
}
```

`\NCC@xexpast` The standard LaTeX's `\@xexpast` macro is saved here:

```
\def\NCC@xexpast#1*#2#3#4\@@{%
 \edef\reserved@a{#1}%
```

```
  \@tempcnta#2\relax
  \ifnum\@tempcnta>\z@
    \@whilenum\@tempcnta>\z@\do
      {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
    \let\reserved@b\NCC@xexpast
  \else
    \let\reserved@b\NCC@xexnoop
  \fi
  \expandafter\reserved@b\reserved@a #4\@@
}
\def\NCC@xexnoop #1\@@{}
```

5.6 NCC Equations

`fleqn` `ceqn` The implementation of these environments is streightforward: change the `\if@fleqn` flag and the `\@mathmargin` value:

```
\newenvironment*{fleqn}[1][\z@]{\@fleqntrue
  \setlength\@mathmargin{#1}\ignorespaces
}{%
  \ignorespacesafterend
}
\newenvironment{ceqn}{\@fleqnfalse
  \@mathmargin\@centering \ignorespaces
}{%
  \ignorespacesafterend
}
```

`\eq` The implementation of the NCC-LaTeX's `\eq` command is quite simple:

```
\newcommand{\eq}{\@ifstar{\NCC@eqx}{\NCC@eq}}
\def\NCC@eqx#1{\begin{equation*}#1\end{equation*}}
\def\NCC@eq#1{\begin{equation}#1\end{equation}}
```

`\eqalign` The `\eqalign` command is based on the `equation` and `darray` environments:

```
\newcommand{\eqalign}{%
  \@ifstar{\let\@tempa\NCC@eqx \NCC@eqa}%
          {\let\@tempa\NCC@eq \NCC@eqa}%
}
\newcommand*{\NCC@eqa}[2][rcl]{%
  \@tempa{\begin{darray}{#1}#2\end{darray}}%
}
```

`\eqs` `eqnarray` The difference between the `\eqs` command and the `eqnarray` environment consists in optional length parameters allowed in `\eqs`. All these commands are based on `\NCC@beqs` and `\NCC@eeqs` macros.

```
\newcommand{\eqs}{\@ifstar{\st@rredtrue\NCC@eqs}{\st@rredfalse \NCC@eqs}}
\newcommand*{\NCC@eqs}[2][]{%
  \begingroup\NCC@beqs{#1}#2\NCC@eeqs\endgroup\ignorespaces
}
\renewenvironment{eqnarray}{\st@rredfalse\NCC@beqs{}}
```

```
                           {\NCC@eeqs\ignorespacesafterend}
\renewenvironment{eqnarray*}{\st@rredtrue\NCC@beqs{}}
                            {\NCC@eeqs\ignorespacesafterend}
```

`\NCC@beqs` The `\NCC@beqs{`⟨*skip*⟩`}` starts eqnarray-like equations. The ⟨*skip*⟩ parameter specifies a skip inserted between columns. If it is empty, the default value of this skip is used. It equals to the thick skip appearing in relations. The implementation of this macro uses hooks from the `amsmath` package.

```
\def\NCC@beqs#1{%
  \NCC@ignorepar$$
  \inalign@true \intertext@ \displ@y@ \Let@
  \chardef\dspbrk@context\z@
  \let\math@cr@@@\NCC@eqcr \let\tag\tag@in@align
  \let\label\label@in@display \let\split\insplit@
  \ifst@rred\else \global\@eqnswtrue \fi
  \tabskip\@mathmargin
  \@ifempty{#1}{\settowidth\alignsep@{$\m@th\mskip\thickmuskip$}}%
               {\setlength\alignsep@{#1}}%
  \halign to \displaywidth\bgroup
    \strut@ \global\column@\z@ \hfil$\displaystyle{##}$\tabskip\z@skip
   &\column@plus \hskip\alignsep@ \hfil$\displaystyle{##}$\hfil
   &\column@plus \hskip\alignsep@ $\displaystyle{##{}}$\hfil
    \tabskip\@centering
   &\column@plus \llap{##}\tabskip\z@skip\cr
}
```

`\NCC@eqcr` The `\NCC@eqcr` hook is called at the end of line of the `eqnarray`. It is originated on LaTeX's `\@eqncr` command, but uses commands from `amsmath` to prepare a tag in the $\mathcal{AMS}$ style.

```
\def\NCC@eqcr{%
  \let\@tempa\relax
  \ifcase\column@ \def\@tempa{&&&}\or \def\@tempa{&&}\or\def\@tempa{&}%
  \else
    \let\@tempa\@empty
    \@latex@error{Too many columns in eqnarray environment}\@ehc
  \fi
  \@tempa
  \ifst@rred\nonumber\fi
  \if@eqnsw \global\tag@true \fi
  \iftag@ \@lign\strut@
    \iftagsleft@ \rlap{\hskip -\displaywidth\make@display@tag}%
    \else \make@display@tag \fi
  \fi
  \ifst@rred\else\global\@eqnswtrue\fi
  \cr
}
```

`\NCC@eeqs` This macro finishes eqnarray-like equations.

```
\def\NCC@eeqs{\math@cr\egroup$$}
```

5.7 Math with medium fractions and operators

Finally, we process the `mediummath` option. It is recognized by the `\NCC@op` command to be specified.

```
\@ifundefined{NCC@op}{\endinput}{}
```

Redifine fractions and binoms.

```
\DeclareRobustCommand\frac{\NCC@op@select\mfrac{\genfrac{}{}{}{}}}
\DeclareRobustCommand\binom{\NCC@op@select\mbinom{\genfrac()\z@{}}}
\def\NCC@op@select#1#2#3#4{%
  \mathchoice{#1{#3}{#4}}{#1{#3}{#4}}%
             {\scriptstyle#2{#3}{#4}}{\scriptscriptstyle#2{#3}{#4}}%
}
```

Redefine all math operators except integrals:

```
\def\@tempa#1#2{%
  \ifx#2\@undefined \let#2#1\fi
  \def#1{\DOTSB\medop{#2}}%
}
\@tempa \coprod    \coprod@
\@tempa \bigvee    \bigvee@
\@tempa \bigwedge  \bigwedge@
\@tempa \biguplus  \biguplus@
\@tempa \bigcap    \bigcap@
\@tempa \bigcup    \bigcup@
\@tempa \prod      \prod@
\@tempa \sum       \sum@
\@tempa \bigotimes \bigotimes@
\@tempa \bigoplus  \bigoplus@
\@tempa \bigodot   \bigodot@
\@tempa \bigsqcup  \bigsqcup@
```

Redefine integrals:

```
\def\@tempa#1#2#3{\let#3#2%
  \DeclareRobustCommand#2{\mathop{\medmath{#3}}}%
  \def#1{\DOTSI\NCC@op@prepare{#2}}%
}
\@tempa\int  \intop  \NCC@op@int
\@tempa\oint \ointop \NCC@op@oint
\let\@tempa\relax
```

Redefine multiple integrals:

```
\renewcommand*{\MultiIntegral}[1]{%
  \edef\NCC@op{\noexpand\intop
    \ifnum#1=\z@\noexpand\intdots@\else\noexpand\intkern@\fi
    \ifnum#1>\tw@\noexpand\intop\noexpand\intkern@\fi
    \ifnum#1>\thr@@\noexpand\intop\noexpand\intkern@\fi
    \noexpand\intop
  }%
  \let\NCC@op@print\NCC@op@printm
  \NCC@op@prepare@
}
\def\intkern@{\kern-\NCC@op@kern}
\def\intdots@{\setlength\@tempdima{\NCC@op@kern}%
  \kern-.4\@tempdima{\cdotp}\mkern1.5mu{\cdotp}%
  \mkern1.5mu{\cdotp}\kern-.4\@tempdima}
⟨/package⟩
```