# The **moremath** package[*]

Marcel Ilg[†]

Released 2024-08-20

**Abstract**

The **moremath** package provides several document level commands to ease the typesetting and LaTeX code readability of certain mathematical constructs. It provides complementary commands to all operators defined by **amsmath**, the commands typeset delimiters that may be automatically, manually or not scaled at all. The commands also accept optional sub- and superscripts to the operators.

Furthermore it provides several commands to typeset gradient, divergence, curl, and Laplace operators, for which there are also versions with delimiters. Those commands also accept an optional subscript and their appearance can be modified using key-value options.

Additionally commands are provided for producing row and column vectors, as well as (anti-)diagonal matrices, utilizing **mathtools** `matrix*` family of environments.

Most of the document level commands defined by this package can also be disabled using a package load-time option to avoid clashes with commands defined by other packages.

## Note: This Package is Still in its Initial Development Phase

Do not expect a stable interface until version 1.0.0 has been reached! While I try to keep the interface stable and backwards compatible, I am unable to promise this.

## Contents

---

[*] This document coresponds to **moremath** v0.5.0, dated 2024-08-20.

[†] `mister01x (at) web (dot) de`

## Part I

# Document Author Documentation

## 1 Introduction

When typesetting mathematics in LaTeX it is very common to often encounter, code patterns such as `\sin \left( \frac{x}{2} \right)`. While this above code works as expected its not especially easy for human readers of this code to immediately understand what that code is going to do.

This package tries to ease readability of such commonly used constructs by providing commands that (hopefully) increase readability, as well as speeding up writing math in LaTeX. Using moremath the above code can be simplified to `\psin*{\frac{x}{2}}`, which is shorter and better "human readable". The package also provides the possibility for users to define such delimited operators, on their own.

There are also other things that can be improved when typesetting math. One prominent example is the typesetting of row and column vectors, which require the use of a `matrix` environment. For this case moremath provides commands which accept a comma separated list of vector entries, obliterating the need for a `matrix` environment to typeset vectors.

The same goes for (anti-)diagonal matrices, only the elements of the diagonal are of importance here. Therefore the production of those matrices may also be simplified into a single command, which again improves readability of the code compared to a mostly empty `matrix` environment.

Another feature of this package is the definition of several (delimited) vector calculus operators such as gradient, divergence and curl operators, this does not only improve the semantics of the math code (`\mathop{\nabla} f` vs. `\grad f`), but also tries to provide the correct spacing between the operator and its arguments.

The moremath package is written using LaTeX3 and provides a small LaTeX3 interface for class and package writers.

The source code for moremath is hosted on GitHub at

`https://github.com/Mister00X/moremath`

If you have any issues or found a bug, feel free to register an issue there.

## 2 Dependencies

The main dependency of this package is mathtools [Høg+24]. Optionally the bm [CMT23] may be loaded.

If the option `no-vector` has *not* been given as a package load time option, this package also loads the amssymb [The] package.

# 3 Package Options

## 3.1 Package Load Time Options

The options described in this section *must* be given at package load time i.e. as package options. All options which are not described below will be passed to mathtools [Høg+24], see its documentation for more information.

`bm` The `bm` option loads the bm [CMT23] package which provides a better version of the `\boldsymbol` command.

`no-vector` `no-abs-shorthands` `no-operators` `no-crvector` `no-matrix` `nopredef` The options `no-vector`, `no-abs-shorthands`, `no-operators`, `no-crvector`, and `no-matrix` disable the definition of the predefined commands described in sections 5, 8, 4, 6, and 7 respectively.

The option `nopredef` achieves the same as the three `no-`⟨*functionality*⟩ options above. It accepts multiple values, valid option values are: `vector`, `abs`, `operators`, `crvector`, `matrix`, and `all`. The values `abs`, `operators`, and `vector` disable the predefined document level commands for delimited operators, vector calculus and the shorthands for absolute value and norm respectively. The values `crvector` and `matrix` disable the shorthand commands for row and column vectors, and simple matrices respectively. The special value `all` disables all of them.

The option accepts multiple values which can be given as a comma separated list, or as multiple key-value options, like in the examples below:

```
\usepackage[nopredef={vector,abs,operators}]{moremath}
```

This is equivalent to

```
\usepackage[nopredef=all]{moremath}
```

and to:

```
\usepackage[nopredef=vector,nopredef=abs,nopredef=operators]{moremath}
```

The command `\NewDelimitedOperator` is not affected by any of the above settings.

## 3.2 General Options

The options described in this section *must not* be given as package options, instead they should be set using `\moremathsetup` or given as optional argument to the commands described later.

`\moremathsetup` `\moremathsetup{`⟨*kv list*⟩`}`

Updated: 2024-07-15

Sets the options specified in the ⟨*key-value list*⟩, the assignment is local to the current group. If a ⟨*value*⟩ contains a comma it needs to be wrapped in braces. This command may be used anywhere in the document after moremath has been loaded.

### 3.2.1 Options Affecting Vector Calculus Operators

`nabla` The option `nabla` sets the symbol to use by the document level commands described in section 5 to use for the nabla. It accepts a list of ⟨*tokens*⟩. Its default value is `\nabla`.

`arrownabla` The option `arrownabla` puts a small arrow over the gradient operator symbol. Its default value is `false`.

`boldnabla` The option `boldnabla` makes the nabla symbol bold. If the `bm` package option has been given the `\boldsymbol` command from the bm package is used for the bold symbol, otherwise the amsmath [The23] version is used.

`grad-op` The option `grad-op` may be used to overwrite, the built in version of the gradient operator, it accepts a ⟨*token list*⟩. Use at your own responsibility.

`laplacian-symb` The option `laplacian-symb` sets the symbol to use by the document level commands described in section 5 to use for the Laplace operator. It accepts a list of ⟨*tokens*⟩.

`delta-laplace` The option `delta-laplace` replaces the Laplace operator symbol (by default $\nabla^2$) with a uppercase delta ($\Delta$). Its default value is `false`.

`arrowlaplace` The option `arrowlaplace` if set to `true` makes the Laplace operator look like this: $\vec{\nabla}^2$.

`laplacian` Like the option `grad-op` above the option `laplacian` may be used to overwrite the built-in version of the Laplace operator. Use at your own responsibility.

`dalembert-symb` Like the option `nabla` this sets the symbol to use by the document level commands described in section 5.5 to use as symbol for the d'Alembert operator. It accepts a list of ⟨*tokens*⟩. It's default value is `\square`.

`vcenter` The option `vcenter` controls if certain mathematical symbols of the operators described in section 5 should be vertically centered along the math-axis. The default value of this option is `true`.

#### 3.2.2 Options Affecting Matrices and Vectors

The options in this section only affect the commands described in sections 6 and 7. To set them with `\moremathsetup` it is necessary to add the prefix `matrix /` to these options, so that the resulting command looks like `\moremathsetup{matrix /` ⟨*option*⟩`}`. When using these options inside the optional argument of the commands described in sections 6 and 7, the prefix `matrix /` must be omitted.

`delimiter` The option `delimiter` determines the delimiters used for the matrices, valid values are `p` for parenthesis, `b` for brackets, `B` for braces, `v` for single vertical lines ("|"), `V` for double vertical lines ("‖") or empty for no delimiters. The default value is `{}` (empty).

`fill` The fill option determines the values an (anti-)diagonal matrix is filled with, outside the diagonal. The default is again empty.

`align` This option determines the alignment of the numbers inside the matrix. The value of this option gets passed to the optional argument of the `matrix*` or `smallmatrix*` family of environments defined by mathtools [Høg+24]. Valid values for both types of those environments are `l` for left alignment, `r` for right alignment and `c` for centered alignment. The default is `c`.

**TEXhackers note:** The non-`small` versions of the commands described in the sections 6 and 7, accept "[...] any column type valid in the usual `array` environment." [Høg+24]

# 4 Delimited Operators

## 4.1 Delimited Operators Predefined by moremath

`no-operators` If the package load time option `no-operators` is not given this package defines several delimited mathematical operators.

Table 1: Operator commands defined by the amsmath [The23] package.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| `\arccos` | arccos | `\deg` | deg | `\lg` | lg | `\projlim` | proj lim |
| `\arcsin` | arcsin | `\det` | det | `\lim` | lim | `\sec` | sec |
| `\arctan` | arctan | `\dim` | dim | `\liminf` | lim inf | `\sin` | sin |
| `\arg` | arg | `\exp` | exp | `\limsup` | lim sup | `\sinh` | sinh |
| `\cos` | cos | `\gcd` | gcd | `\ln` | ln | `\sup` | sup |
| `\cosh` | cosh | `\hom` | hom | `\log` | log | `\tan` | tan |
| `\cot` | cot | `\inf` | inf | `\max` | max | `\tanh` | tanh |
| `\coth` | coth | `\injlim` | inj lim | `\min` | min | | |
| `\csc` | csc | `\ker` | ker | `\Pr` | Pr | | |

| | | | |
|---|---|---|---|
| `\varinjlim` | $\varinjlim$ | `\varliminf` | $\varliminf$ |
| `\varprojlim` | $\varprojlim$ | `\varlimsup` | $\varlimsup$ |

```
\parccos
\barccos
\Barccos
\varccos
\Varccos
```

```
\parccos [⟨size cmd⟩] {⟨contents⟩}
\parccos [⟨size cmd⟩] ^{⟨superscript⟩} {⟨contents⟩}
\parccos [⟨size cmd⟩] _{⟨subscript⟩} {⟨contents⟩}
\parccos [⟨size cmd⟩] ^{⟨superscript⟩} _{⟨subscript⟩} {⟨contents⟩}
\parccos* {⟨contents⟩}
\parccos* ^{⟨superscript⟩} {⟨contents⟩}
\parccos* _{⟨subscript⟩} {⟨contents⟩}
\parccos* ^{⟨superscript⟩} _{⟨subscript⟩} {⟨contents⟩}
\⟨prefix⟩⟨op name⟩ [⟨size cmd⟩] {⟨contents⟩}
\⟨prefix⟩⟨op name⟩ [⟨size cmd⟩] ^{⟨superscript⟩}  {⟨contents⟩}
\⟨prefix⟩⟨op name⟩ [⟨size cmd⟩]  _{⟨subscript⟩} {⟨contents⟩}
\⟨prefix⟩⟨op name⟩ [⟨size cmd⟩] ^{⟨superscript⟩} _{⟨subscript⟩} {⟨contents⟩}
\⟨prefix⟩⟨op name⟩* {⟨contents⟩}
\⟨prefix⟩⟨op name⟩* ^{⟨superscript⟩}  {⟨contents⟩}
\⟨prefix⟩⟨op name⟩* _{⟨subscript⟩} {⟨contents⟩}
\⟨prefix⟩⟨op name⟩* ^{⟨superscript⟩} _{⟨subscript⟩} {⟨contents⟩}
```

For all of the operators predefined by amsmath [The23], which are shown in table 1, moremath declares delimited versions. The name of those commands follows the scheme `\⟨prefix⟩⟨op⟩`, where `⟨prefix⟩` is one of `p`, `b`, `B`, `v`, or `V` and `⟨op⟩` is the name of one of the operators shown in table 1.

The `⟨prefix⟩`es `p`, `b`, `B`, `v`, and `V` stand for parenthesis, brackets, braces, single vertical lines ("|"), and double vertical lines ("‖") respectively.

The commands accept a `⟨size cmd⟩` optional argument, which is usually one of `\big`, `\Big`, `\bigg` and `\Bigg`. These `⟨size cmd⟩`s are used to change the size of the delimiters.

The commands also accept sub- and superscripts, which have to be issued *after* the optional argument (if present), but before the mandatory argument `⟨contents⟩`.

The starred variant uses automatic scaling for the delimiters depending on the height of its contents.

**Examples** The following examples showcase the use of those predefined delimited operators:

1. Different delimited operators without any scaling:

```
\[
    \pcos{x} \times \bcos{y}
    \times \Bcos{z} \times \vcos{a}
    \times \Vcos{b}
  \]
```

$$\cos(x) \times \cos[y] \times \cos\{z\} \times \cos|a| \times \cos\|b\|$$

2. Delimited operator with automatic scaling:

```
\[
  \pcos*{\frac{x^{2}}{2}}
\]
```

$$\cos\left(\frac{x^{2}}{2}\right)$$

3. Delimited operator with manual scaling:

```
\[
  \pcos[\Big]{\frac{x^{2}}{2}}
\]
```

$$\cos\Big(\frac{x^{2}}{2}\Big)$$

4. Delimited operator with subscript:

```
\[
  \plog_{10}{1+x}
\]
```

$$\log_{10}(1+x)$$

5. Delimited operator with superscript:

```
\[
  \pcos^{2}{x}
\]
```

$$\cos^{2}(x)$$

6. Delimited operator with both sub- and superscript and manual scaling:

```
\[
  \pcos[\Big]^{2}_{x}{\frac{x}{2}}
\]
```

$$\cos^{2}_{x}\Big(\frac{x}{2}\Big)$$

## 4.2 Declaring New Delimited Operators

`\DeclareDelimitedOperator`

`\DeclareDelimitedOperator{⟨new op⟩}{⟨op⟩}{⟨delim⟩}`

Creates a new delimited operator, the name of the new command will be ⟨*new op*⟩. The ⟨*op*⟩ is the command of the operator to use, which is usually a command declared with `\DeclareMathOperator`. ⟨*delim*⟩ is the command to use as paired delimiter, it is expected to behave like a paired delimiter declared by mathtools [Høg+24] `\DeclarePairedDelimiter`.

**Example: Creating a New Delimited Operator** The following code creates a new operator and a paired delimiter and uses it afterwards to declare a paired operator.

```
\documentclass{scrartcl}

\DeclareMathOperator{\glorb}{glorb}
\DeclarePairedDelimiter{\inparen}{\lparen}{\rparen}
\DeclarePairedOperator{\pglorb}{\glorb}{\inparen}

\begin{document}
```

```
\[
  \pglorb{a}
\]
\end{document}
```

The result then looks like this:

$$\mathrm{glorb}(a)$$

# 5 Vector Calculus Operators

`no-vector` The commands in this section are only declared if the option `no-vector` has not been given to the package as a load time option.

`vcenter` The option `vcenter` controls if the symbols for the operators described below should be vertically centered along the math axis. Its default value is `true`.

This option only shows its effects if other options like `arrownabla`, `arrowlaplace`, or `boldnabla` are set to `true`. Like in the example below:

```
\begin{gather*}
  \grad f(x) \quad \grad[arrownabla,vcenter=false] f(x)
  \quad \grad[arrownabla,vcenter=true] f(x)\\
  \laplacian f(x) \quad \laplacian[arrowlaplace,vcenter=false] f(x)
  \quad \laplacian[arrowlaplace,vcenter=true] f(x)
\end{gather*}
```

$$\begin{gathered}\nabla f(x) \quad \vec{\nabla} f(x) \quad \vec{\nabla} f(x) \\ \nabla^2 f(x) \quad \vec{\nabla}^2 f(x) \quad \vec{\nabla}^2 f(x)\end{gathered}$$

All of the commands described in this section take key-value options as optional argument, which are described in section 3.2.1

## 5.1 Gradient Operator Commands

### 5.1.1 Standalone Operator Command

`\grad`
Updated: 2024-07-08

```
\grad [⟨kv opts⟩]
\grad [⟨kv opts⟩] _{⟨subscript⟩}
```

The `\grad` command produces a gradient operator looking like this “$\nabla$” by default. The optional argument ⟨*kv opts*⟩ accepts the key-value options described in section 3.2.1, whitespace between the command name and `[`⟨*kv opts*⟩`]` is *not allowed*. An optional subscript using `_` may be given after the optional argument.

**Examples of Use**

1. Standalone gradient operator (with and without subscript):

```
\[
  \grad f(x), \quad \grad_{x} f(x)
\]
```

$$\nabla f(x), \quad \nabla_x f(x)$$

2. Bold version of the gradient operator:

```
\[
  \grad[boldnabla] f(x)
\]
```

$$\boldsymbol{\nabla} f(x)$$

3. Gradient operator with an arrow:

```
\[
  \grad[arrownabla] f(x)
\]
```

$$\vec{\nabla} f(x)$$

### 5.1.2 Operators with Delimiters

```
\pgrad \⟨delim⟩grad [⟨size cmd⟩] {⟨content⟩}
\bgrad \⟨delim⟩grad [⟨kv opts⟩] {⟨content⟩}
\Bgrad \⟨delim⟩grad [⟨size cmd⟩] _{⟨subscript⟩} {⟨content⟩}
\vgrad \⟨delim⟩grad [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
\Vgrad \⟨delim⟩grad* [⟨kv opts⟩] {⟨content⟩}
       \⟨delim⟩grad* [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
```

The `\⟨delim⟩grad` family of commands produces gradient operator which is followed by ⟨*contents*⟩ inside delimiters. The delimiter is determined by the first letter of the command ⟨*delim*⟩, which may be `p` for parenthesis, `b` for brackets, `B` for braces, `v` for a single vertical line ("|"), or `V` for a double vertical line ("‖").

The commands accept either a ⟨*size command*⟩ as optional argument, which gets passed to mathtools [Høg+24] paired delimiter or a list of ⟨*key-value option*⟩s, the ⟨*kv opts*⟩ *must* be given using the complete syntax, i.e. ⟨*key*⟩=⟨*value*⟩, shorthands for options with an implicit default value (`arrownabla`), will not work here. The ⟨*size command*⟩ is usually one of `\big`, `\Big`, `\bigg` and `\Bigg`. Valid ⟨*kv opts*⟩ are all options described in section 3.2.1 and the key `scale` which accepts a ⟨*size cmd*⟩.

> **Note:**
> Do not mix ⟨*kv opts*⟩ and ⟨*size cmd*⟩, use either `\pgrad[\big]{f(x)}` or `\pgrad[arrownabla=true,scale=\big]{f(x)}`.

An optional ⟨*subscript*⟩ may be given between the optional argument, and ⟨*contents*⟩. One use case for this subscript is to write formulae using the so called Feynman-notation, where the gradient operator acts only on one variable.

The starred version of the commands automatically scale the delimiters with its contents.

**Examples:**

1. Gradient operator with non-scaled delimiters:

```
\[
  \pgrad{1+\vec{x}}
\]
```

$$\nabla(1+\vec{x})$$

2. Bold version:

```
\[
  \pgrad[boldnabla=true]{1+\vec{x}}
\]
```

$$\boldsymbol{\nabla}(1+\vec{x})$$

3. Gradient operator with automatically scaled delimiters:

```
\[
  \pgrad*{\frac{1}{x}}
\]
```

$$\nabla\left(\frac{1}{x}\right)$$

4. Manually scaled version:

```
\begin{gather*}
  \pgrad[\Big]{\frac{1}{x}}\\[.5ex]
  \pgrad[scale=\Big]{\frac{1}{x}}
\end{gather*}
```

$$\nabla\Big(\frac{1}{x}\Big)$$
$$\nabla\Big(\frac{1}{x}\Big)$$

5. Feynman-notation:

```
\[
  \pgrad_{x}{x+y+z}
\]
```

$$\nabla_x(x+y+z)$$

## 5.2 Divergence Operator Commands

### 5.2.1 Standalone Operator Command

`\divergence`

Updated: 2024-07-08

`\divergence [⟨kv opts⟩]`
`\divergence [⟨kv opts⟩] _{⟨subscript⟩}`

The `\divergence` command produces the divergence operator “∇·”, its usage is analogous to the use of the `\grad` command, which is described in section 5.1.1.

**Examples**

1. Standalone divergence operator

```
\[
  \divergence f(x)
\]
```

$$\nabla\cdot f$$

2. Standalone divergence operator with an arrow over the gradient operator

```
\[
  \divergence[arrownabla] f(x)
\]
```

$$\vec{\nabla}\cdot f(x)$$

3. Standalone divergence operator with subscript

```
\[
  \divergence_{x} f(x)
\]
```

$$\nabla_x\cdot f(x)$$

### 5.2.2 Operators with Delimiters

```
\pdiv \⟨delim⟩div [⟨size cmd⟩] {⟨content⟩}
\bdiv \⟨delim⟩div [⟨kv opts⟩] {⟨content⟩}
\Bdiv \⟨delim⟩div [⟨size cmd⟩] _{⟨subscript⟩} {⟨content⟩}
\vdiv \⟨delim⟩div [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
\Vdiv \⟨delim⟩div* [⟨kv opts⟩] {⟨content⟩}
      \⟨delim⟩div* [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
```

The `\⟨delim⟩div` family of commands produces the divergence operator with its arguments placed inside delimiters. The usage of these commands is analogous to the `\⟨delim⟩grad` family of commands described in section 5.1.2.

**Examples**

1. Divergence operator with parenthesis and no scaling

```
\[
  \pdiv{1+x}
\]
```

$$\nabla\cdot(1+x)$$

2. Bold version with manual scaling and subscript

```
\[
  \pdiv[boldnabla=true,scale=\Big]_{x}{1 + \frac{1}{x}}
\]
```

$$\boldsymbol{\nabla}_x\cdot\Big(1+\frac{1}{x}\Big)$$

3. Automatic scaling

```
\[
  \pdiv*{1 + \frac{1}{x}}
\]
```

$$\nabla\cdot\left(1+\frac{1}{x}\right)$$

## 5.3 Curl Operator Commands

### 5.3.1 Standalone Operator Command

```
\curl   \curl [⟨kv opts⟩]
        \curl [⟨kv opts⟩] _{⟨subscript⟩}
```

Updated: 2024-07-08

The `\curl` command produces the curl operator "$\nabla\times$", its usage is analogous to the use of the `\grad` command described in section 5.1.1.

**Examples**

1. Standalone curl operator

```
\[
  \curl f(x)
\]
```

$$\nabla\times f(x)$$

2. Standalone curl operator with an arrow over the gradient operator

```
\[
  \curl[arrownabla] f(x)
\]
```

$$\vec{\nabla} \times f(x)$$

3. Standalone curl operator with subscript

```
\[
  \curl_{x} f(x,y)
\]
```

$$\nabla_x \times f(x,y)$$

### 5.3.2 Operators with Delimiters

```
\pcurl  \⟨delim⟩curl [⟨size cmd⟩] {⟨content⟩}
\bcurl  \⟨delim⟩curl [⟨kv opts⟩] {⟨content⟩}
\Bcurl  \⟨delim⟩curl [⟨size cmd⟩] _{⟨subscript⟩} {⟨content⟩}
\vcurl  \⟨delim⟩curl [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
\Vcurl  \⟨delim⟩curl* [⟨kv opts⟩] {⟨content⟩}
        \⟨delim⟩curl* [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
```

The `\⟨delim⟩curl` family of commands produce the curl operator with its arguments placed inside delimiters. The usage of these commands is analogous to the `\⟨delim⟩grad` family of commmands described in section 5.1.2.

#### Examples

1. Curl operator with parenthesis without scaling

```
\[
  \pcurl{1+x}
\]
```

$$\nabla \times (1 + x)$$

2. Bold version with manual scaling and subscript

```
\[
  \pcurl[boldnabla=true,scale=\Big]_{x}{1 + \frac{1}{x}}
\]
```

$$\boldsymbol{\nabla}_x \times \Big(1 + \frac{1}{x}\Big)$$

3. Automatic scaling

```
\[
  \pcurl*{1 + \frac{1}{x}}
\]
```

$$\nabla \times \left(1 + \frac{1}{x}\right)$$

## 5.4 Laplace Operator Commands

This section describes commands which can be used to typeset a Laplace operator.

Like the commands described in sections 5.1, 5.2, and 5.3 the commands in this section accept key-value options via an optional argument. There is some deviation from the options compared to the above commands: The `arrownabla` option is ignored, instead the `arrowlaplace` option produces an arrow over the operator. The `boldnabla` option on the other hand is not ignored. Finally the `delta-laplace` option replaces the symbol used for the operator from $\nabla^2$ to $\Delta$

### 5.4.1 Standalone Operator Command

`\laplacian`

Updated: 2024-07-08

```
\laplacian [⟨kv opts⟩]
\laplacian [⟨kv opts⟩] _{⟨subscript⟩}
```

The `\laplacian` command produces a Laplace operator, which looks by default like this: $\nabla^2$.

Its interface is analogous to the `\grad`, `\divergence`, and `\curl` commands described above, with the difference in key-value options described at the start of this subsection.

### 5.4.2 Operators with Delimiters

`\plaplacian`
`\blaplacian`
`\Blaplacian`
`\vlaplacian`
`\Vlaplacian`

```
\⟨delim⟩laplacian [⟨size cmd⟩] {⟨content⟩}
\⟨delim⟩laplacian [⟨kv opts⟩] {⟨content⟩}
\⟨delim⟩laplacian [⟨size cmd⟩] _{⟨subscript⟩} {⟨content⟩}
\⟨delim⟩laplacian [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
\⟨delim⟩laplacian* [⟨kv opts⟩] {⟨content⟩}
\⟨delim⟩laplacian* [⟨kv opts⟩] _{⟨subscript⟩} {⟨content⟩}
```

**Examples**

1. Laplace operator delimited by parenthesis without scaling

```
\[
  \plaplacian{1+x}
\]
```

$$\nabla^2(1+x)$$

2. Version with arrow, manual scaling and subscript

```
\[
  \plaplacian[arrowlaplace=true,scale=\Big]_{x}{1 + \frac{1}{x}}
\]
```

$$\vec{\nabla}^2_x\Big(1+\frac{1}{x}\Big)$$

3. Version with automatic scaling

```
\[
  \plaplacian*{1 + \frac{1}{x}}
\]
```

$$\nabla^2\left(1+\frac{1}{x}\right)$$

4. Using a delta as symbol for the Laplacian

```
\[
  \plaplacian[delta-laplace=true]{1+x}
\]
```

$$\Delta(1+x)$$

## 5.5 Commands Producing a d'Alembert operator

### 5.5.1 Standalone Operator Command

`\quabla`

New: 2024-07-04
Updated: 2024-07-08

```
\quabla [⟨kv opts⟩]
\quabla [⟨kv opts⟩] _{⟨subscript⟩}
```

The `\quabla` command produces the d'Alembert operator "□". This command accepts an optional subscript.

The command is called `\quabla` because thats shorter and easier to type than `\dalembertian`. If you want to have a command called `\dalembertian`, put the following in your documents preamble.

```
\NewCommandCopy\quabla\dalembertian
```

**Example of Use:**

```
\[
  \quabla f(x)
\]
```

$$\Box f(x)$$

### 5.5.2 Operators with Delimiters

`\pquabla`
`\bquabla`
`\Bquabla`
`\vquabla`
`\Vquabla`

New: 2024-07-04

```
\⟨delim⟩quabla [⟨size cmd⟩] {⟨contents⟩}
\⟨delim⟩quabla [⟨kv opts⟩] {⟨contents⟩}
\⟨delim⟩quabla [⟨size cmd⟩] _{⟨subscript⟩} {⟨contents⟩}
\⟨delim⟩quabla [⟨kv opts⟩] _{⟨subscript⟩} {⟨contents⟩}
\⟨delim⟩quabla* [⟨kv opts⟩] {⟨contents⟩}
\⟨delim⟩quabla* [⟨kv opts⟩] _{⟨subscript⟩} {⟨contents⟩}
```

The `\⟨delim⟩quabla` family of commands produce a d'Alembert operator with ⟨*contents*⟩ placed inside delimiters. Their usage is analogous to the `\⟨delim⟩grad` family of commands described in section 5.1.2.

# 6 Row- and Column Vectors

`no-crvector` The command in this section are only declared if the option `no-crvector` has not been given as a package option.

Valid keys are `delimiter`, `fill`, and `align`, their usage is described in section 3.2.2.

`\cvector`
`\rvector`

`\cvector [⟨kv opts⟩] {⟨clist⟩}`
`\rvector [⟨kv opts⟩] {⟨clist⟩}`

Updated: 2024-08-20

The commands `\cvector` and `\rvector` produce row and column vectors respectively. Both of them accept key-value options as optional argument ⟨*kv opts*⟩. Valid keys and values are described in section 3.2.2.

The mandatory argument ⟨*clist*⟩ is a comma-separated-list, whose elements are the entries of the column/row vector. If a comma has to appear inside an entry the entire entry has to be wrapped in braces.

The delimiter of the row or column vectors depends on the current value of the option `delimiter`, by default empty. The option `fill` has no effect on the commands and is simply ignored.

**TEXhackers note:** If you were to define your own `matrix*`-like environment called `mymatrix*`, which has an interface compatible to `mathtools`'s `matrix*` family of environments, you could make use of it by setting the value of `delimiter` to `my`.

### Examples:

1. Column “vector” without delimiters:

```
\[
  \cvector{a_{1},a_{2},a_{3}}
\]
```

$$\begin{matrix}a_1\\a_2\\a_3\end{matrix}$$

2. A column vectors delimited with parenthesis and different alignment:

```
\[
  \cvector[delimiter=p,align=c]{-1,2,3}
\]
```

$$\begin{pmatrix*}[c]-1\\2\\3\end{pmatrix*}$$

```
\[
  \cvector[delimiter=p,align=r]{-1,2,3}
\]
```

$$\begin{pmatrix*}[r]-1\\2\\3\end{pmatrix*}$$

```
\[
  \cvector[delimiter=p,align=l]{-1,2,3}
\]
```

$$\begin{pmatrix*}[l]-1\\2\\3\end{pmatrix*}$$

**Row- and Column Vectors with Predefined Delimiters** As vectors are commonly delimited by parenthesis, brackets, braces, etc. several shorthands producing delimited vectors are also available.

`\pcvector`
`\bcvector`
`\Bcvector`
`\vcvector`
`\Vcvector`
`\prvector`
`\brvector`
`\Brvector`
`\vrvector`
`\Vrvector`

`\⟨delim⟩cvector [⟨kv opts⟩] {⟨clist⟩}`
`\⟨delim⟩rvector [⟨kv opts⟩] {⟨clist⟩}`

Updated: 2024-08-20

The `\⟨delim⟩⟨c or r⟩vector` family of commands, accepts a list of key-value options as optional argument ⟨*kv opts*⟩. Valid keys are described in section 3.2.2.

The mandatory argument ⟨*clist*⟩ is a comma-separated-list of the entries of the vector. If a comma needs to appear inside an entry of the vector, that entry has to be wrapped in braces.

⟨*delim*⟩ may have the value `p` for parenthesis, `b` for brackets, `B` for braces, `v` for a single vertical line, or `V` for a double vertical line.

**Example**

```
\[
  \pcvector{a_{1},a_{2},a_{3}}
\]
```

$$\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix}$$

## 6.1 Small Versions for Inline Math

As the `matrix` and `matrix*` family of environments is unsuitable, for inline math, mathtools [Høg+24] provides the `smallmatrix` and `smallmatrix*` family of environments. This package provides analogous commands for row and column vectors to be typeset in inline math mode.

`\smallcvector`
`\smallrvector`
Updated: 2024-08-20

`\smallcvector [⟨kv opts⟩] {⟨clist⟩}`
`\smallrvector [⟨kv opts⟩] {⟨clist⟩}`

`\smallcvector` and `\smallrvector` produce column and row vectors suitable for inline math. Both commands accept the same optional key-value arguments as `\cvector` and `\rvector`.

**Example:**

```
An inline version of \verb|\cvector| looks like this
\(\smallcvector{1,0}\).
```

An inline version of `\cvector` looks like this $\begin{smallmatrix}1\\0\end{smallmatrix}$.

`\psmallcvector`
`\bsmallcvector`
`\Bsmallcvector`
`\vsmallcvector`
`\Vsmallcvector`
`\psmallrvector`
`\bsmallrvector`
`\Bsmallrvector`
`\vsmallrvector`
`\Vsmallrvector`
Updated: 2024-08-20

`\psmallcvector [⟨kv opts⟩] {⟨clist⟩}`
`\psmallrvector [⟨kv opts⟩] {⟨clist⟩}`
`\⟨delim⟩small⟨c or r⟩vector [⟨kv opts⟩] {⟨clist⟩}`

The `\⟨delim⟩small⟨c or r⟩vector` family of commands like `\psmallcvector` produce small inline math version of row and column vectors. Their interface is identical to the commands described above.

**Example:**

```
An inline version of \verb|\pcvector| looks like this
\(\psmallcvector{1,0}\).
```

An inline version of `\pcvector` looks like this $\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$.

# 7 Shorthands for Simple Matrices

`no-matrix` The commands in this section are only defined if the option `no-matrix` *has not been given* to the package at load-time.

## 7.1 (Anti-)diagonal Matrices

`\diagmat`
`\antidiagmat`

```
\diagmat [⟨kv opts⟩] {⟨diagonal⟩}
\antidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
```

`\diagmat` and `\antidiagmat` produce a diagonal or anti-diagonal matrix respectively. The optional argument ⟨*kv opts*⟩ accepts the key value options described in section 3.2.2.

The key `fill` determines the contents of the matrix entries which are not part of the (anti-)diagonal, its default value is `{}` i.e. empty. The key `align` determines the alignment of the entries inside the matrix, valid values are usually `l`, `c`, and `r`, the default is `c`. The key `delimiter` determines the delimiter around the matrix, its default value is `{}` (none). Valid values for the delimiters are `p` for parenthesis, `b` for brackets, `B` for braces, `v` for a single vertical line ("|"), and `V` for a double vertical line ("‖"). See the mathtools manual [Høg+24] for more information.

**TEXhackers note:** The value of `delimiter` gets inserted inside the `\begin{#1matrix*}` and `\end{#1matrix*}` commands. Therefore it would be possible to define your own `matrix*` like environment, called for example `mymatrix*` and set `delimiter=my` to make use of it.

The mandatory argument ⟨*diagonal*⟩ has to be a comma separated list of the entries of the (anti-)diagonal. If an entry of the diagonal needs to contain a comma `,` the entire entry has to be wrapped in braces.

**Examples:**

1. A diagonal and an anti-diagonal matrix without delimiters:

```
\begin{gather*}
  \diagmat{1,2,3}\\[.5ex]
  \antidiagmat{1,2,3}
\end{gather*}
```

$$\begin{gathered}\begin{matrix}1&&\\&2&\\&&3\end{matrix}\\[.5ex]\begin{matrix}&&1\\&2&\\3&&\end{matrix}\end{gathered}$$

2. A diagonal matrix delimited by parenthesis filled with zeros:

```
\[
  \diagmat[delimiter=p,fill=0]{1,2,3}
\]
```

$$\begin{pmatrix}1&0&0\\0&2&0\\0&0&3\end{pmatrix}$$

3. A diagonal matrix filled with a symbol:

```
\[
  \diagmat[fill=\square]{1,2,3}
\]
```

$$\begin{matrix}1&\square&\square\\\square&2&\square\\\square&\square&3\end{matrix}$$

4. Diagonal matrices with different alignment:

```
\[
  \diagmat[delimiter=p,align=c,fill=0]{-1,-2,-3} \qquad
  \diagmat[delimiter=p,align=l,fill=0]{-1,-2,-3} \qquad
  \diagmat[delimiter=p,align=r,fill=0]{-1,-2,-3}
\]
```

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix} \qquad \begin{pmatrix*}[l] -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix*} \qquad \begin{pmatrix*}[r] -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix*}$$

**Shorthands for (Anti-)diagonal Matrices with Delimiters** As matrices are more often than not written inside of delimiters, moremath provides several shorthands for producing those matrices without having to set the `delimiter` key explicitly.

```
\pdiagmat \pdiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\bdiagmat \bdiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Bdiagmat \Bdiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\vdiagmat \vdiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Vdiagmat \Vdiagmat [⟨kv opts⟩] {⟨diagonal⟩}
```

The `\⟨delim⟩diagmat` family of commands provides shorthands for producing a delimited (anti-)diagonal matrix without having to set the `delimiter` key explicitly every time. The pre-set `delimiter` may be overwritten by explicitly passing `delimiter` as a key-value option.

The optional argument ⟨*kv opts*⟩ accepts the same key-value arguments as `\diagmat`.

**Example:**

```
\[
  \pdiagmat{1,2,3}
\]
```

$$\begin{pmatrix} 1 & & \\ & 2 & \\ & & 3 \end{pmatrix}$$

```
\pantidiagmat \pantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\bantidiagmat \bantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Bantidiagmat \Bantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\vantidiagmat \vantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Vantidiagmat \Vantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
```

The `\⟨delim⟩antidiagmat` family of commands behaves like the `\⟨delim⟩diagmat` commands described above, except they produce anti-diagonal matrices.

**Example:**

```
\[
  \pantidiagmat{1,2,3}
\]
```

$$\begin{pmatrix} & & 1 \\ & 2 & \\ 3 & & \end{pmatrix}$$

### 7.1.1 Small Versions for Inline Math

mathtools defines special matrix environments for use in inline math mode, the `smallmatrix` and `smallmatrix*` family of environments. moremath provides for the case of typesetting an (anti-)diagonal matrix several commands which utilize these inline math versions.

```
\smalldiagmat     \smalldiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\smallantidiagmat \smallantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
```

The `\smalldiagmat` and `\smallantidiagmat` commands behave like their non-`small` counterpart described above, see their description for more information.

**Example:**

```
An inline version of a diagonal matrix looks like this
\(\smalldiagmat{1,1}\).
```

An inline version of a diagonal matrix looks like this $\begin{smallmatrix}1&\\&1\end{smallmatrix}$.

```
\psmalldiagmat \psmalldiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\bsmalldiagmat \bsmalldiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Bsmalldiagmat \Bsmalldiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\vsmalldiagmat \vsmalldiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Vsmalldiagmat \Vsmalldiagmat [⟨kv opts⟩] {⟨diagonal⟩}
```

Like the `\⟨delim⟩diagmat` commands described above there are also shorthand commands for producing an inline math version of a diagonal matrix with pre-set delimiters.

**Example:**

```
An inline math delimited diagonal matrix looks like this
\(\psmalldiagmat{1,1}\).
```

An inline math delimited diagonal matrix looks like this $\left(\begin{smallmatrix}1&\\&1\end{smallmatrix}\right)$.

```
\psmallantidiagmat \psmallantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\bsmallantidiagmat \bsmallantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Bsmallantidiagmat \Bsmallantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\vsmallantidiagmat \vsmallantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
\Vsmallantidiagmat \Vsmallantidiagmat [⟨kv opts⟩] {⟨diagonal⟩}
```

Like the `\⟨delim⟩antidiagmat` commands described above there are also shorthand commands for producing inline math versions of anti-diagonal matrices inside of delimiters.

**Example:**

```
A delimited version of an inline math anti-diagonal matrix
looks like this \(\psmallantidiagmat{1,1}\).
```

A delimited version of an inline math anti-diagonal matrix looks like this $\left(\begin{smallmatrix}&1\\1&\end{smallmatrix}\right)$.

## 7.2 Identity Matrices

As identity matrices are always quadratic, one can further simplify the typesetting of identity matrices to a command, which constructs the identity matrix from the number of dimensions. The commands in this subsection perform this.

| | |
|---|---|
| `\idmat` | `\idmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\smallidmat` | `\smallidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |

`New: 2024-07-04`

The `\idmat` command produces an identity matrix. The optional argument ⟨*kv opts*⟩ accepts any valid *matrix* key-value arguments, which are described in section 3.2.2. The mandatory argument ⟨*dimension*⟩ expects a *positive* integer, which specifies the dimensions of the identity matrix.

The `\smallidmat` command behaves like the `\idmat` command, except it produces a matrix suitable for inline math mode.

**TEXhackers note:** The maximum number of columns is determined by the TEX counter `MaxMatrixCols`, which has a default value of 10. See The LaTeX Project Team [The23] for more information.

| | |
|---|---|
| `\pidmat` | `\pidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\bidmat` | `\bidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\Bidmat` | `\Bidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\vidmat` | `\vidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\Vidmat` | `\Vidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |

`New: 2024-07-04`

Like the `\`⟨*delim*⟩`diagmat` commands, the `\`⟨*delim*⟩`idmat` commands produce an identity matrix with a pre-set delimiter around the matrix, i.e. they behave like `\idmat[delimiter = `⟨*delim*⟩`]{`⟨*dimension*⟩`}`. ⟨*delim*⟩ can be `p` for parenthesis, `b` for brackets, `B` for braces, `v` for a single vertical line ("|"), or `V` for a double vertical line ("‖").

Furthermore these commands accept the same ⟨*kv opts*⟩ as the `\idmat` command.

| | |
|---|---|
| `\psmallidmat` | `\psmallidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\bsmallidmat` | `\bsmallidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\Bsmallidmat` | `\Bsmallidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\vsmallidmat` | `\vsmallidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |
| `\Vsmallidmat` | `\Vsmallidmat [`⟨*kv opts*⟩`] {`⟨*dimension*⟩`}` |

`New: 2024-07-04`

These commands are provide inline math suitable versions of the `\`⟨*delim*⟩`idmat` commands described above. See their description for more information.

The commands accept any of the key-value options described in section 3.2.2 as optional argument ⟨*kv opts*⟩.

### Examples of Use

1. Identity matrix without delimiters:

```
\[
  \idmat{3}
\]
```

$$\begin{matrix} 1 & & \\ & 1 & \\ & & 1 \end{matrix}$$

2. Identity matrices with different delimiters:

```
\[
  \pidmat{3}\text{,}\quad \bidmat{3}\text{,}\quad
  \Bidmat{3}\text{,}\quad \vidmat{3}\text{,}\quad
  \Vidmat{3}
\]
```

$$\begin{pmatrix}1&&\\&1&\\&&1\end{pmatrix}, \quad \begin{bmatrix}1&&\\&1&\\&&1\end{bmatrix}, \quad \begin{Bmatrix}1&&\\&1&\\&&1\end{Bmatrix}, \quad \begin{vmatrix}1&&\\&1&\\&&1\end{vmatrix}, \quad \begin{Vmatrix}1&&\\&1&\\&&1\end{Vmatrix}$$

3. Inline math versions of identity matrices with delimiters:

```
Inline math mode matrices look like this:
\(\psmallidmat{2}\), \(\bsmallidmat{2}\), \(\Bsmallidmat{2}\),
\(\vsmallidmat{2}\), \(\Vsmallidmat{2}\).
```

   Inline math mode matrices look like this: $\left(\begin{smallmatrix}1&\\&1\end{smallmatrix}\right)$, $\left[\begin{smallmatrix}1&\\&1\end{smallmatrix}\right]$, $\left\{\begin{smallmatrix}1&\\&1\end{smallmatrix}\right\}$, $\left|\begin{smallmatrix}1&\\&1\end{smallmatrix}\right|$, $\left\|\begin{smallmatrix}1&\\&1\end{smallmatrix}\right\|$.

4. Identity matrix using the `fill` option for the non-diagonal elements

```
\[
  \pidmat[fill=\star]{3}
\]
```

$$\begin{pmatrix}1&\star&\star\\\star&1&\star\\\star&\star&1\end{pmatrix}$$

# 8 Shorthands for Absolute Value and Norm

`no-abs-shorthands`

The commands in this section are only declared if the option `no-abs-shorthands` has not been given to the package as a load time option.

`\abs`
`\norm`

```
\abs [⟨size cmd⟩] {⟨content⟩}
\abs* {⟨content⟩}
\norm [⟨size cmd⟩] {⟨content⟩}
\norm* {⟨content⟩}
```

The commands `\abs` and `\norm`, produce $|\cdot|$ and $\|\cdot\|$ respectively. These commands are simply paired delimiters defined using mathtools' `\DeclarePairedDelimiter` command, instruction on their usage can therefore be found in Høgholm et al. [Høg+24].

# 9 Vertically Centering Math Along the Math Axis

Sometimes it is useful to explicitly center a math symbol along the math axis. A prominent example is the case of `\mathop{\boldsymbol\nabla}\nolimits` vs. `\mathop{\nabla}\nolimits`, as displayed below.

$$\boldsymbol{\nabla} f(x) \quad \nabla f(x)$$

Here the bold nabla is slightly higher up than the non-bold version.

`\VCenterMath`

New: 2024-07-08

```
\VCenterMath {⟨contents⟩}
```

The command `\VCenterMath` centers ⟨*contents*⟩ along the current math axis, while adhering to the current math style. This command is not in a TeX-sense `\long`, i.e. it does not take `\par` tokens.

This command is somewhat dangerous as it utilizes the `\vcenter` primitive, which leaves math mode and requires to reenter it. *Use at your own responsibility.*

**TeXhackers note:** This command is a wrapper around `\vcenter`, `\hbox:n` and `\mathpalette`, it reenters math mode inside the hbox.

With `\VCenterMath` it is possible to fix the above example to:

```
\[
  \mathop{\VCenterMath{\boldsymbol\nabla}}\nolimits f(x)
  \quad \mathop{\nabla}\nolimits f(x)
\]
```

$$\boldsymbol{\nabla} f(x) \quad \nabla f(x)$$

# Part II

# Documentation for Class and Package Authors

## 1 Setting of Options

`\moremath_setup:n`

Updated: 2024-07-15

`\moremath_setup:n {`⟨*kv opts*⟩`}`

This functions sets the key-value options ⟨*kv opts*⟩ in the `moremath` namespace.

## 2 Delimited Operators

`\moremath_delim_op_noscale:NNnnn`
`\moremath_delim_op_autoscale:NNnnn`

Updated: 2024-07-15

`\moremath_delim_op_noscale:NNnnn` ⟨*math op*⟩ ⟨*delim*⟩ `{`⟨*sup tl*⟩`} {`⟨*sub tl*⟩`} {`⟨*tl*⟩`}`
`\moremath_delim_op_autoscale:NNnnn` ⟨*math op*⟩ ⟨*delim*⟩ `{`⟨*sup tl*⟩`} {`⟨*sub tl*⟩`} {`⟨*tl*⟩`}`

The two functions `\moremath_delim_op_noscale:NNnnn` and `\moremath_delim_op_autoscale:NNnnn` provide a version of a delimited operator with no scaling or automatic scaling respectively. The token list arguments ⟨*sup tl*⟩ and ⟨*sub tl*⟩ take the super- and subscript of the operator respectively. Both of those token lists may be empty.

The first argument to the function is a control sequence ⟨*math op*⟩, which should be an operator declared with `\DeclareMathOperator`. The second argument ⟨*delim*⟩ has to be a control sequence of a paired delimiter, as defined by mathtools's `\DeclarePairedDelimiter`.

The final argument ⟨*tl*⟩ is a list of arbitrary token to put inside the delimiters.

`\moremath_delim_op_manuscale:NNnnnn`
`\moremath_delim_op_manuscale:NNVnnn`

Updated: 2024-07-15

`\moremath_delim_op_manuscale:NNnnnn` ⟨*math op*⟩ ⟨*delim*⟩ `{`⟨*scale cmd*⟩`} {`⟨*sup*⟩`} {`⟨*sub*⟩`} {`⟨*tl*⟩`}`

The function `\moremath_delim_op_manuscale:NNnnnn` typesets a math operator with manual scaling of its delimiters. Its arguments are identical to `\moremath_delim_op_noscale:NNnnn` except for the argument ⟨*scale cmd*⟩ which has to be a token list containing a ⟨*size cmd*⟩ that can be understood by mathtools, usually `\big`, `\Big`, `\bigg` and `\Bigg`.

`\moremath_new_delim_op_command:NNN`
`\moremath_new_delim_op_command:cNN`

Updated: 2024-07-15

```
\moremath_new_delim_op_command:NNN ⟨new csname⟩ ⟨operator⟩ ⟨delim⟩
```

The function `\moremath_new_delim_op_command:NNN` defines a new document level command called ⟨*`new csname`*⟩, utilizing the operator ⟨*`operator`*⟩ and the delimiter ⟨*`delim`*⟩.

⟨*`operator`*⟩ has to be a control sequence referring to an operator, as declared by `\DeclareMathOperator`.

⟨*`delim`*⟩ has to be a paired delimiter as defined by mathtools' `\DeclarePairedDelimiter`.

# 3 Vector Calculus Typesetting

## 3.1 Functions Producing Standalone Operators with Subscripts

`\moremath_gradient_operator:n`
`\moremath_divergence_operator:n`
`\moremath_curl_operator:n`
`\moremath_laplace_operator:n`
`\moremath_dalembert_operator:n`

Updated: 2024-07-15

```
\moremath_gradient_operator:n {⟨subscript⟩}
\moremath_divergence_operator:n {⟨subscript⟩}
\moremath_curl_operator:n {⟨subscript⟩}
\moremath_laplace_operator:n {⟨subscript⟩}
\moremath_dalembert_operator:n {⟨subscript⟩}
```

Each of these functions return a vector calculus operator, with an (optional) subscript. The argument ⟨*`subscript`*⟩, a token list, may be empty, in which case no subscript is produced.

The actual operator produced depends on the current settings of the key-value options described in section 3 inside the current group.

## 3.2 Functions Producing Operators with Delimiters

`\moremath_delim_nabla_op_noscale:NNnn`
`\moremath_delim_nabla_op_autoscale:NNnn`

Updated: 2024-07-15

```
\moremath_delim_nabla_op_noscale:NNnn ⟨operator⟩ ⟨delim⟩
   {⟨sub⟩} {⟨contents⟩}
\moremath_delim_nabla_op_autoscale:NNnn ⟨operator⟩ ⟨delim⟩
   {⟨sub⟩} {⟨contents⟩}
```

The usage of these functions is similar to the functions `\moremath_delim_op_noscale:NNnnn` and `\moremath_delim_op_autoscale:NNnnn` except for the missing superscript.

The first argument ⟨*`operator`*⟩ has to be a function which accepts one argument (the subscript). This is usually one of the `\moremath_`⟨*`op`*⟩`_operator:n` commands.

The argument ⟨*`sub`*⟩, which may be empty, is a token list to pass as the subscript to `\moremath_`⟨*`op`*⟩`_operator:n`.

The last argument ⟨*`contents`*⟩ is a token list to put inside the delimiters.

`\moremath_delim_nabla_op_manuscale:NNnnn`
`\moremath_delim_nabla_op_manuscale:NNVnn`

Updated: 2024-07-15

```
\moremath_delim_nabla_op_manuscale:NNnnn ⟨operator⟩ ⟨delim⟩
   {⟨scale cmd⟩} {⟨sub⟩} {⟨contents⟩}
```

The usage of this function is similar to the above functions except for the additional argument ⟨*`scale cmd`*⟩ which is a token list with a ⟨*`size cmd`*⟩ that should be understood by mathtools (e.g. `\big`, `\Big`, etc.).

# 4 Formatting Matrices and Vectors

**`\moremath_column_vector:nn`**
`\moremath_column_vector:Vn`
**`\moremath_row_vector:nn`**
`\moremath_row_vector:Vn`
**`\moremath_column_smallvector:nn`**
`\moremath_column_smallvector:Vn`
**`\moremath_row_smallvector:nn`**
`\moremath_row_smallvector:Vn`

Updated: 2024-08-20

```
\moremath_column_vector:nn {⟨delim spec tl⟩} {⟨clist⟩}
\moremath_row_vector:nn {⟨delim spec tl⟩} {⟨clist⟩}
\moremath_column_smallvector:nn {⟨delim spec tl⟩} {⟨clist⟩}
\moremath_row_smallvector:nn {⟨delim spec tl⟩} {⟨clist⟩}
```

These functions produce a column or row vector. The `small`⟨*c or r*⟩`vector` versions are intended for inline math mode.

The first argument ⟨*delim spec tl*⟩ should be the specification of a delimiter, as used in the naming of the `matrix*` environments by mathtools.

The second argument ⟨*clist*⟩, is a comma-separated list of entries of the vector.

Of this commands the `c` versions produce column vectors, the `r` versions produce row vectors.

The alignment of the entries inside the vector depends on the current value of the key `align`.

**`\moremath_diagonal_matrix:nn`**
`\moremath_diagonal_matrix:(nV|Vn|VV)`
**`\moremath_antidiagonal_matrix:nn`**
`\moremath_antidiagonal_matrix:(nV|Vn|VV)`
**`\moremath_diagonal_smallmatrix:nn`**
`\moremath_diagonal_smallmatrix:(nV|Vn|VV)`
**`\moremath_antidiagonal_smallmatrix:nn`**
`\moremath_antidiagonal_smallmatrix:(nV|Vn|VV)`

Updated: 2024-08-20

```
\moremath_diagonal_matrix:nn {⟨delim tl⟩} {⟨clist⟩}
\moremath_antidiagonal_matrix:nn {⟨delim tl⟩} {⟨clist⟩}
\moremath_diagonal_smallmatrix:nn {⟨delim tl⟩} {⟨clist⟩}
\moremath_antidiagonal_smallmatrix:nn {⟨delim tl⟩}
{⟨clist⟩}
```

These functions produce (anti-)diagonal matrices. The argument ⟨*delim tl*⟩ is a token list, intended to specify the delimiter of the matrix or no delimiter if the ⟨*tl*⟩ is empty no delimiters are produced.

The argument ⟨*clist*⟩ is a list of comma separated values which specify the entries of the diagonal. Other properties of the matrix to produce like alignment (`align`) and the tokens to insert for non-diagonal entries (`fill`) depend on the current setting of the keys described in section 3.2.2.

The `smallmatrix` versions are intended for use in inline math mode.

`\moremath_id_matrix:n`
`\moremath_id_matrix:V`
`\moremath_id_smallmatrix:n`
`\moremath_id_smallmatrix:V`

New: 2024-07-04
Updated: 2024-07-15

`\moremath_id_matrix:n {⟨dimensions⟩}`
`\moremath_id_smallmatrix:n {⟨dimensions⟩}`

These functions produce a identity matrix with ⟨*dimensions*⟩ rows and columns. ⟨*dimensions*⟩ is expected to be a *positive* integer. The `smallmatrix` version utilizes mathtools' `smallmatrix*` environment, which makes it suitable for inline math mode.

The delimiter around the matrix is determined by the current value of the key `matrix / delimiter` inside the current group.

As these functions utilize `\moremath_diagonal_matrix:VV` or `\moremath_diagonal_smallmatrix:VV` and make use of temporary variables it is advised to put them into their own group, i.e. surround them by `\group_begin:` and `\group_end:`.

## 5 Vertically Centering Math Along the Math Axis

As already said in section 9 sometimes it is needed to explicitly center some math code.

`\moremath_vcenter:n`

New: 2024-07-08
Updated: 2024-08-20

`\moremath_vcenter:n {⟨contents⟩}`

The function `\moremath_vcenter:n` places ⟨*contents*⟩ inside a `hbox`, which is centered along the math axis by `\vcenter`. The contents inside the `hbox` are set in math mode. The function also applies the "outer" math style to the contents of the `hbox`.

This function is not `\long`, as it sets its contents inside a `hbox`.

**TEXhackers note:** This function is a wrapper around `\vcenter`, `\hbox:n`, and `\mathpalette`. Inside the `hbox` math mode is reentered.

# Part III

# Implementation

Start the DocStrip guards.

```
⟨*package⟩
```

Set the internal prefix for this package so that DocStrip knows what to do.

```
⟨@@=moremath⟩
```

## 1 Initial Setup

First set the required version of LaTeX, we need at least

```
\NeedsTeXFormat{LaTeX2e}[2022-11-01]
```

for key-value option handling, xparse-like document commands (especially for using the = option specification) and hooks.

Afterwards we provide rollback information, for the latexrelease rollback mechanism. (See Mitelbach [Mit18] for more information.)

For the v0.x.x versions I've decided on using the date of the first release that was uploaded to CTAN.

```
\DeclareCurrentRelease{v0}{2024-07-15}
```

Then identify this package as moremath.

```
\ProvidesExplPackage{moremath}
  {2024-08-20}{v0.5.0}{More Math Macros}
```

# 2 Key-value interfaces

To make use of key-value interfaces we need to define a few keys.

## 2.1 Package load time options

To allow for the conditional definition of macros at load time we define a few keys.

But before doing so we define a few messages to write the package options to the log file. One message to issue if the bm option has been given:

```
\msg_new:nnn { moremath } {load / bm}
{
  Option~'bm'~given.\\
  Loading~the~bm~package~\msg_line_context:.
}
```

And a more generic message to issue for the other options, which all disable certain parts of the library.

```
\msg_new:nnn {moremath} { load / disabling }
{
  Option~'#1'~given.\\
  Disabling~#2~\msg_line_context:.
}
```

\l__moremath_predef_vector_op_bool
\l__moremath_predef_operators_bool
\l__moremath_predef_crvector_bool
\l__moremath_predef_matrix_bool
\l__moremath_predef_abs_bool

To store the values of several switch type key-value options we declare several boolean variables.

```
\bool_new:N \l__moremath_predef_vector_op_bool
\bool_new:N \l__moremath_predef_operators_bool
\bool_new:N \l__moremath_predef_crvector_bool
\bool_new:N \l__moremath_predef_matrix_bool
\bool_new:N \l__moremath_predef_abs_bool
```

The variables `\l__moremath_predef_vector_op_bool`, `\l__moremath_predef_operators_bool` `\l__moremath_predef_abs_bool`, `\l__moremath_predef_matrix_bool` and `\l__moremath_predef_crvector_bool` control if the predefined macros for vector calculus, delimited operators, the matrix shorthands, the row and column vectors, and the shorthands for absolute value and norm shall be defined.

(*End of definition for* `\l__moremath_predef_vector_op_bool` *and others.*)

**bm**
**no-vector**
**no-operators**
**no-abs-shorthands**
**no-matrix**
**no-crvector**
**nopredef**

Now we define package load time keys:

```
\keys_define:nn { moremath / load }
{
```

We provide an option to conditionally load the bm-package [CMT23] to provide better looking bold symbols.

```
  bm .code:n = {
    \msg_info:nn {moremath} {load / bm}
    \RequirePackage{bm}
  },
bm .value_forbidden:n = true,
bm. usage:n = load,
```

Then we define options for en-/disabling predefined macros of this package to avoid name clashes.

```
no-vector .bool_set_inverse:N = \l__moremath_predef_vector_op_bool,
no-vector .default:n = true,
no-vector .initial:n = false,
no-vector .usage:n = load,
no-operators .bool_set_inverse:N = \l__moremath_predef_operators_bool,
no-operators .default:n = true,
no-operators .initial:n = false,
no-operators .usage:n = load,
no-abs-shorthands .bool_set_inverse:N = \l__moremath_predef_abs_bool,
no-abs-shorthands .default:n = true,
no-abs-shorthands .initial:n = false,
no-abs-shorthands .usage:n = load,
no-matrix .bool_set_inverse:N = \l__moremath_predef_matrix_bool,
no-matrix .initial:n = false,
no-matrix .default:n = true,
no-matrix .usage:n =load,
no-crvector .bool_set_inverse:N = \l__moremath_predef_crvector_bool,
no-crvector .default:n = true,
no-crvector .initial:n = false,
no-crvector .usage:n = load,
nopredef .multichoice:,
nopredef / operators .meta:nn = { moremath / load }
{
  no-operators = true
},
nopredef / vector .meta:nn = { moremath / load }
{
  no-vector = true
},
nopredef / abs .meta:nn = { moremath / load }
{
  no-abs-shorthands = true,
},
noperdef / matrix .meta:nn = { moremath / load }
{
  no-matrix = true,
},
nopredef / crvector .meta:nn = {moremath / load}
{
  no-crvector = true,
},
nopredef / all .meta:nn = { moremath / load }
{
  no-operators = true,
  no-vector = true,
  no-abs-shorthands = true
},
nopredef .usage:n = load,
```

Unknown package options get passed to mathtools.

```
  unknown .code:n = {\PassOptionsToPackage{\CurrentOption}{mathtools}},
  unknown .usage:n = load,
}
```

## 2.2 Keys controlling appearance

`\l__moremath_nabla_tl`
`\l__moremath_nabla_arrow_bool`
`\l__moremath_nabla_arrow_bold_bool`
`\l__moremath_grad_op_tl`
`\l__moremath_laplacian_symb_tl`
`\l__moremath_laplacian_delta_bool`
`\l__moremath_laplacian_arrow_bool`
`\l__moremath_laplacian_tl`
`\l__moremath_dalembert_symb_tl`

We declare several variables to store the values of the keys affecting appearance.

```
\tl_new:N \l__moremath_nabla_tl
\bool_new:N \l__moremath_nabla_arrow_bool
\bool_new:N \l__moremath_nabla_arrow_bold_bool
\tl_new:N \l__moremath_grad_op_tl
```

The symbol to use as "nabla" is stored in `\l__moremath_nabla_tl`. The variables `\l__moremath_nabla_arrow_bool` and `\l__moremath_nabla_bold_bool` determine if the nabla-symbol shall have an arrow over itself and/or be bold respectively.

The variable `\l__moremath_grad_op_tl` contains a user provided token list to overwrite the built-in gradient operator of the package.

```
\tl_new:N \l__moremath_laplacian_symb_tl
\bool_new:N \l__moremath_laplacian_delta_bool
\bool_new:N \l__moremath_laplacian_arrow_bool
\tl_new:N \l__moremath_laplacian_tl
```

The token list variable `\l__moremath_laplacian_symb_tl` stores the tokens to be used for the Laplace operator. The boolean variable `\l__moremath_laplacian_delta_bool` determines if a delta should be used instead of $\nabla^2$ for the laplacian symbol. If the boolean variable `\l__moremath_laplacian_arrow_bool` a small arrow will be placed over the Laplace operator symbol. If the user wants to overwrite the symbol used for the Laplacian, the user provided list of tokens is stored in the variable `\l__moremath_laplacian_tl`.

Finally we define a variable to hold the symbol to use for the d'Alembert operator.

```
\tl_new:N \l__moremath_dalembert_symb_tl
```

(*End of definition for* `\l__moremath_nabla_tl` *and others.*)

`\l__moremath_vcenter_bool`

Additionally we declare a variable to decide if certain math symbols shall be centered explicitly along the math axis.

```
\bool_new:N \l__moremath_vcenter_bool
```

If `\l__moremath_vcenter_bool` is true, the symbols of certain math operators, should be centered explicitly.

(*End of definition for* `\l__moremath_vcenter_bool`.)

`nabla`
`arrownabla`
`boldnabla`
`grad-op`
`laplacian-symb`
`delta-laplace`
`arrowlaplace`
`laplacian`
`dalembert-symb`

First define keys for the vector calculus macros.

```
\keys_define:nn { moremath }
{
```

First we define keys for use with the gradient and gradient based operators.

```
  % Symbol to use for the nabla
   nabla .tl_set:N = \l__moremath_nabla_tl,
   nabla .initial:n = {\nabla},
   nabla .value_required:n = true,
   % shall the nabla have an arrow over it
   arrownabla .bool_set:N = \l__moremath_nabla_arrow_bool,
   arrownabla .default:n = {true},
   arrownabla .initial:n = {false},
   % shall the nabla be bold
   boldnabla .bool_set:N = \l__moremath_nabla_bold_bool,
   boldnabla .default:n = {true},
   boldnabla .initial:n = {false},
```

We also provide an override for the gradient operator

```
   % Symbol to use for the gradient operator
   grad-op .tl_set:N = \l__moremath_grad_op_tl,
   grad-op .value_required:n = true,
```

Then we define keys for the laplacian.

```
  % Symbol to use for the laplacian
  laplacian-symb .tl_set:N = \l__moremath_laplacian_symb_tl,
  laplacian-symb .initial:n = {\l__moremath_nabla_tl},
  % shall the Laplace operator be a delta
  delta-laplace .bool_set:N = \l__moremath_laplacian_delta_bool,
  delta-laplace .initial:n = {false},
  % shall the laplace operator have an arrow over itself
  arrowlaplace .bool_set:N = \l__moremath_laplacian_arrow_bool,
  arrowlaplace .default:n = {true},
  arrowlaplace .initial:n = {false},
  % overwrite the laplace operator
  laplacian .tl_set:N = \l__moremath_laplacian_tl,
  laplacian .value_required:n = true,
```

And keys for the d'Alembert operator.

```
  dalembert-symb .tl_set:N = \l__moremath_dalembert_symb_tl,
  dalembert-symb .initial:n = {\square},
```

`vcenter`

The `vcenter` option will control the manual centering of certain math operators.

```
  vcenter .bool_set:N = \l__moremath_vcenter_bool,
  vcenter .initial:n = true,
  vcenter .value_required:n = true,
}% \keys_define:nn
```

`delimiter`
`fill`
`align`

Then we define some keys for the matrix based environments:

```
\keys_define:nn { moremath / matrix }
{
```

`\l__moremath_matrix_delim_tl`
`\l__moremath_matrix_fill_tl`
`\l__moremath_matrix_align_tl`

Every one of the following keys stores its value inside a token list variable.

```
  delimiter .tl_set:N = \l__moremath_matrix_delim_tl,
  delimiter .initial:n = {},
  fill .tl_set:N = \l__moremath_matrix_fill_tl,
  fill .initial:n = {},
  align .tl_set:N = \l__moremath_matrix_align_tl,
  align .initial:n = {c},
  align .value_required:n = true,
}
```

The keys `delimiter` and `fill` set the variables `\l__moremath_matrix_delim_tl` and `\l__moremath_matrix_fill_tl` respectively. `\l__moremath_matrix_delim_tl` determines the delimiter in use to surround matrices and `\l__moremath_matrix_fill_tl` determines the fill values of the `\diag`, `\smalldiag`, `\Xdiag` and `\Xsmalldiag` commands, which are used for non-diagonal matrix entries. The variable `\l__moremath_matrix_align_tl` contains the alignment specifier for use with the `matrix*` environment.

(*End of definition for* `\l__moremath_matrix_delim_tl`*,* `\l__moremath_matrix_fill_tl`*, and* `\l__moremath_matrix_align_tl`*.*)

## 2.3 Functions for Setting Options

Now we define a function for setting the options within the document

`\moremath_setup:n`

```
\cs_new_protected:Nn \moremath_setup:n
{
  \keys_set:nn {moremath} {#1}
}
```

(*End of definition for* `\moremath_setup:n`*. This function is documented on page 23.*)

Additionally we provide the user with a version of this command.

`\moremathsetup`

```
\NewDocumentCommand \moremathsetup {m}
{
  \moremath_setup:n {#1}
}
```

(*End of definition for* `\moremathsetup`*. This function is documented on page 5.*)

We also need a function for setting the package load time options. This function should set all given values for all key families and pass unknown options to mathtools.

`\__moremath_load_time_setup:`

```
\cs_new_protected:Nn \__moremath_load_time_setup:
{
  \ProcessKeyOptions[ moremath / load ]
}
```

(*End of definition for* `\__moremath_load_time_setup:`*.*)

# 3 Package Initialization

We now “initialize” the package by processing the package options, all unknown options are passed to mathtools [Høg+24], which is loaded afterwards. Because certain mathtools-features are needed by this package, we need to require a version of at least 2004/06/05. As explained in section 2.1 this may also load bm if the bm package option has been given.

```
\__moremath_load_time_setup:
```

If the `no-vector` option has not been given during load time, we also need the amssymb package [The] for the `\square` command. We first define a message to inform the user about this.

```
\msg_new:nnn { moremath } { load / loading-amssymb }
{
  Vector~calculus~commands~enabled.\\
  Loading~amssymb~package~\msg_line_context:.
}
```

Then we conditionally load the package.

```
\bool_if:NT \l__moremath_predef_vector_op_bool
{
  \msg_info:nn { moremath } { load / loading-amssymb }
  \RequirePackage{amssymb}
}
```

Finally we load our most important dependency mathtools.

```
\RequirePackage{mathtools}[2004/06/05]
```

# 4 Abstractions for TeX-primitives and LaTeX 2$_\varepsilon$-commands

As this package makes use of certain TeX-primitives and LaTeX 2$_\varepsilon$-commands, which are either part of the LaTeX-kernel or of some package. It is sensible to provide a (package internal) abstraction for those commands, so that they can be switched out once a LaTeX3-command has been defined for them.

## 4.1 TeX-primitives

`\__moremath_tex_vcenter:n`

`\__moremath_tex_vcenter:n` is an abstraction for `\tex_vcenter:D` i.e. `\vcenter`. Its single argument is passed to `\tex_vcenter:D`.

```
\cs_new_protected:Nn \__moremath_tex_vcenter:n
{
  \tex_vcenter:D {#1}
}
```

(*End of definition for* `\__moremath_tex_vcenter:n`*.*)

`\__moremath_tex_mathpalette:Nn`

The function `\__moremath_tex_mathpalette:Nn` is an abstraction for the `\mathpalette`, primitive. The function takes two arguments

#1 : The ⟨*csname*⟩ of the macro to pass to `\mathpalette`
This macro **must** accept **two** arguments: The first is a single token containing the current math-style, the second is the contents to "typeset". See https://tex.stackexchange.com/questions/34393/the-mysteries-of-mathpalette/34412#34412 for further information.

#2 : The ⟨*tokens*⟩ to forward to the macro passed as first argument.

```
\cs_new_protected:Nn \__moremath_tex_mathpalette:Nn
{
  \mathpalette #1 {#2}
}
```

(*End of definition for* `\__moremath_tex_mathpalette:Nn`*.*)

`\__moremath_tex_mathsurround:n`

The function `\__moremath_tex_mathsurround:n` provides (hopefully) a correct abstraction for the `\mathsurround` primitive. It takes one argument

`#1` : A ⟨*length expr*⟩ for the whitespace surrounding a math switch.

```
\cs_new_protected_nopar:Nn \__moremath_tex_mathsurround:n
{
  \tex_mathsurround:D = #1
}
```

(*End of definition for* `\__moremath_tex_mathsurround:n`*.*)

`\__moremath_tex_mathop:n`

The function `\__moremath_tex_mathop:n` provides an abstraction for the `\mathop` TeX-primitive. It accepts a single argument:

`#1` : ⟨*Tokens*⟩ to treat as a math operator.

```
\cs_new_protected_nopar:Nn \__moremath_tex_mathop:n
{
  \tex_mathop:D {#1}
}
```

(*End of definition for* `\__moremath_tex_mathop:n`*.*)

## 4.2 LaTeX 2$_\varepsilon$-commands

`\__moremath_matrix_star_begin:nn`
`\__moremath_matrix_star_begin:nV`
`\__moremath_matrix_star_begin:Vn`
`\__moremath_matrix_star_begin:VV`
`\__moremath_matrix_star_end:n`
`\__moremath_matrix_star_end:V`

The functions `\__moremath_matrix_star_begin:nn` and `\__moremath_matrix_star_end:n` provide an abstraction to the `matrix*` family of environments provided by mathtools.

`\__moremath_matrix_star_begin:nn` takes two arguments:

`#1` : A ⟨*tl*⟩ determining the type of the matrix.
This is used to further specify which `matrix*` environment to use.
The ⟨*tl*⟩ may be empty.

`#2` : A ⟨*tl*⟩ specifying the column type.

```
\cs_new_nopar:Nn \__moremath_matrix_star_begin:nn
{
  \begin{ #1 matrix* } [ #2 ]
}
```

We also create `nV`, `Vn`, and `VV` variants of this function:

```
\cs_generate_variant:Nn \__moremath_matrix_star_begin:nn { nV, Vn, VV }
```

The function `\__moremath_matrix_star_end:n` takes only one argument

`#1` : The ⟨*tl*⟩ determining the type of the `matrix*` environment.
The ⟨*tl*⟩ **must** have the same value as used for `\__moremath_matrix_star_begin:nn`.

```
\cs_new_nopar:Nn \__moremath_matrix_star_end:n
{
  \end{ #1 matrix* }
}
```

We also provide a `V` type variant of this function.

```
\cs_generate_variant:Nn \__moremath_matrix_star_end:n {V}
```

(*End of definition for* `\__moremath_matrix_star_begin:nn` *and* `\__moremath_matrix_star_end:n`*.*)

## 5 Centering Math Symbols Along the Math-Axis

Certain math constructs such cause TEX to not center the operator along the math axis, like case of `\mathop{\nabla}\nolimits` vs. `\mathop{\boldsymbol\nabla}\nolimits`, as can be seen below.

$$\mathop{\nabla}\nolimits f(x) \quad \text{vs.} \quad \mathop{\boldsymbol{\nabla}}\nolimits f(x)$$

As can be seen the bold nabla symbol is slightly higher up than the non-bold version. Because of this it is sometimes useful to manually center some math symbols, along the math axis.[1]

`\moremath_vcenter:n` The function `\moremath_vcenter:n` is a wrapper around the `\vcenter` TEX primitive. It takes a single argument.

`#1` : A ⟨*tl*⟩ containing math mode material to center along the math axis.
This argument is typeset in math mode.

The function uses the `\mathpalette` primitive to switch to the right math style. The function is not in a TEX-sense long, i.e. it does not take `\par` tokens.

As this function might be useful not only for internal use by moremath, we declare it as a public function here.

```
\cs_new_protected_nopar:Nn \moremath_vcenter:n
{
  \__moremath_tex_mathpalette:Nn \__moremath_vcenter:Nn {#1}
}
```

(*End of definition for* `\moremath_vcenter:n`. *This function is documented on page 26.*)

`\__moremath_vcenter:Nn` As `\mathpalette` needs as its first argument a macro which takes two arguments (the first is a math style switch and the second the contents).[2] We define an internal helper function for `\moremath_vcenter:n` for `\mathpalette` to call.

The function `\__moremath_vcenter:Nn` takes two arguments:

`#1` : The math style macro, which is passed to this function by `\mathpalette`.

`#2` : The ⟨*tl*⟩ to typeset inside the `\vcenter`.

Because of the properties of `\vcenter`, it switches to vertical mode, we need to put the contents to typeset inside a horizontal box. Because of this we also need to reenter math mode, and because of this we need to remove the spacing inserted by entering math mode by setting `\mathsurround` to zero.

```
\cs_new_protected_nopar:Nn \__moremath_vcenter:Nn
{
  \__moremath_tex_vcenter:n
  {
    \hbox:n
    {
      $
      \__moremath_tex_mathsurround:n {0pt}
      #1 {#2}
      $
    }
  }
}
```

(*End of definition for* `\__moremath_vcenter:Nn`.)

---

[1] The code in this section is inspired by http://www.tug.org/TUGboat/Articles/tb22-4/tb72perlS.pdf

[2] See TEXSE answer https://tex.stackexchange.com/a/34412

### Document Level Command

Although unlikely there might arise the need for a document author to center some math along the math axis. For this purpose we are going to define a new document level command.

But first we are going to declare some messages to use by the command. The first message informs the user that the command is not available, because its ⟨*csname*⟩ is already taken.

```
\msg_new:nnn { moremath } { csname-already-defined }
{
  Control~sequence~' #1 '~is~already~ defined.\\
  Skipping~definition~\msg_line_context:
}
```

The second message should be issued if a command that only works in math mode was given outside of it.

```
\msg_new:nnnn { moremath } { vcenter / only-in-math-mode }
{
  Command~#1~used~outside~math~mode~\msg_line_context:.
}
{
  The~command~#1~may~only~be~used~inside~math~mode.
}
```

`\VCenterMath`

Now to the document level command for centering math along the math axis.

```
\cs_if_free:NTF \VCenterMath
{
  \NewDocumentCommand \VCenterMath { m }
  {
    \mode_if_math:TF
    {
      \moremath_vcenter:n {#1}
    }{% \mode_if_math:TF FALSE BRANCH
      \msg_error:nnn { moremath } { vcenter / only-in-math-mode } {\VCenterMath}
    }
  }
}{% \cs_if_free:NTF \VCenterMath FALSE BRANCH
  \msg_warning:nnn { moremath } { csname-already-defined } {\VCenterMath}
}
```

(*End of definition for* `\VCenterMath`. *This function is documented on page* 22.)

## 6 Declaring Paired Delimiters for Internal Use

`\__moremath_inparent:w`
`\__moremath_inbrack:w`
`\__moremath_inbrace:w`
`\__moremath_in_vert:w`
`\__moremath_in_Vert:w`

As we are going to use mathtools' *paired delimiters* at several places throughout this package, we define *paired delimiters* for internal use, as functions with weird syntax.

```
\DeclarePairedDelimiter{\__moremath_inparent:w}{\lparen}{\rparen}
\DeclarePairedDelimiter{\__moremath_inbrack:w}{\lbrack}{\rbrack}
\DeclarePairedDelimiter{\__moremath_inbrace:w}{\lbrace}{\rbrace}
\DeclarePairedDelimiter{\__moremath_in_vert:w}{\lvert}{\rvert}
\DeclarePairedDelimiter{\__moremath_in_Vert:w}{\lVert}{\rVert}
```

(*End of definition for* `\__moremath_inparent:w` *and others.*)

# 7 Delimited Operators

We need three different functions for providing the delimited operators. But as we share a lot of code between those, we define an additional helper function beforehand.

`\__moremath_operator:Nnn`

The function `\__moremath_operator:Nnn` takes care of the operator part of the new delimiter. It allows the operator to have super- and subscripts. It takes three arugments.

`#1` : The ⟨*csname*⟩ of the operator to use.
`#2` : A ⟨*token list*⟩, which is used as the superscript operator.
This argument may be empty
`#3` : A ⟨*tl*⟩, which is used as the subscript operator.
The ⟨*tl*⟩ may be empty.

```
\cs_new_protected:Nn \__moremath_operator:Nnn
{
  #1
  % add superscript if present
  \tl_if_empty:nF {#2} {^{#2}}
  % add subscript if present
  \tl_if_empty:nF {#3} { \c_math_subscript_token {#3} }
}
```

(*End of definition for* `\__moremath_operator:Nnn`*.*)

We now define three version of the delimited operators.

`\moremath_delim_op_noscale:NNnnn`
`\moremath_delim_op_autoscale:NNnnn`

`\moremath_delim_op_noscale:NNnnn` is provides a delimited operator without any scaling of the delimiters and `\moremath_delim_op_autoscale:NNnnn` provides a version with automatic scaling of the delimiters. Both of them take five arguments:

`#1` : The ⟨*csname*⟩ of the operator to use.
Any operator declared with amsmath's `\operatorname` and/or `\DeclareMathOperator` is valid for this.
`#2` : The ⟨*csname*⟩ of a paired delimiter declared by mathtools' [Høg+24] `\DeclarePairedDelimiter`.
`#3` : A ⟨*tl*⟩ to use as the superscript of the operator.
`#4` : A ⟨*tl*⟩ to use as the subscript of the operator.
`#5` : A ⟨*tl*⟩ to insert inside the delimiters.

```
\cs_new_protected_nopar:Nn \moremath_delim_op_noscale:NNnnn
{
  \__moremath_operator:Nnn #1 {#3} {#4}
  % #2 is the paired delimiter
  #2 {#5}
}

\cs_new_protected_nopar:Nn \moremath_delim_op_autoscale:NNnnn
{
  \__moremath_operator:Nnn #1 {#3} {#4}
  % #2 is the paired delimiter
  #2 * {#5}
}
```

(*End of definition for* `\moremath_delim_op_noscale:NNnnn` *and* `\moremath_delim_op_autoscale:NNnnn`*. These functions are documented on page 23.*)

`\moremath_delim_op_manuscale:NNnnnn`
`\moremath_delim_op_manuscale:NNVnnn`

`\moremath_delim_op_manuscale:NNnnnn` provides a delimited operator with manual scaling. This version takes six arguments:

#1 : The ⟨*csname*⟩ of the operator to use.
#2 : The ⟨*csname*⟩ of the paired delimiter declared by mathtools’ [Høg+24] `\DeclarePairedDelimiter`.
#3 : A ⟨*tl*⟩ containing the scaling macro i.e. `\big`, `\Big`, `\Bigg`,…
#4 : A ⟨*tl*⟩ containing the superscript of the operator.
The ⟨*tl*⟩ may be empty
#5 : A ⟨*tl*⟩ containing the subscript of the operator
The ⟨*tl*⟩ may be empty.
#6 : A ⟨*tl*⟩ to insert inside the delimiters

```
\cs_new_protected_nopar:Nn \moremath_delim_op_manuscale:NNnnnn
{
  \__moremath_operator:Nnn #1 {#4} {#5}
  % #2 is the paired delimiter
  #2 [ #3 ] {#6}
}
```

We also provide a variant for the scaling macro.

```
\cs_generate_variant:Nn \moremath_delim_op_manuscale:NNnnnn {NNVnnn}
```

(*End of definition for* `\moremath_delim_op_manuscale:NNnnnn`*. This function is documented on page 23.*)

For the creation of document level commands we also create a function, so that the user is also able to declare new delimited operators. But before we do so we create a message to inform the user if a ⟨*csname*⟩ was already taken.

```
\msg_new:nnn { moremath } { delimop / already-defined-skip }
{
  Control~sequence~’#1’~is~already~defined.\\
  Skipping~definition~of~delimited~operator~’#1’~\msg_line_context:.
}
```

We also create a message to inform the user about conflicting options given to the command.

```
\msg_new:nnn { moremath } { delimop / auto-manu-scale-conflict }
{
  Both~star~and~scale-cmd~given~to~’#1’.\\
  Automatic~scaling~will~be~preferred,~the~size~command~will~be~
  ignored~\msg_line_context:.
}
```

`\moremath_new_delim_op_command:NNN`
`\moremath_new_delim_op_command:cNN`

`\moremath_new_delim_op_command:NNN` takes three arguments:
#1 : The ⟨*csname*⟩ to be defined.
#2 : The ⟨*csname*⟩ of the operator to use, which should be an operator declared with `\DeclareMathOperator`.
#3 : The ⟨*csname*⟩ of the delimiter to use, which should have been declared with `\DeclarePairedDelimiter`.

```
\cs_new_protected:Nn \moremath_new_delim_op_command:NNN
{
  \cs_if_free:NTF #1
  {
    \exp_args:NNe \NewDocumentCommand #1
    { s o E{ ^ \char_generate:nn {‘_} {8} }{{}{}} m }
    {
      \tl_if_novalue:nTF {##2}
      {
        % second argument empty
        \bool_if:nTF {##1}
        {
          % star given
          \moremath_delim_op_autoscale:NNnnn #2 #3 {##3} {##4} {##5}
        }{
          % star not given
          \moremath_delim_op_noscale:NNnnn #2 #3 {##3} {##4} {##5}
        }
      }{
        % second argument present
        % star given?
        \bool_if:nTF {##1}
        {
          % Warn if both star and scaling factor are present
          \msg_warning:nnn { moremath } { delimop / auto-manu-scale-conflict }
            {#1}
          \moremath_delim_op_autoscale:NNnnn #2 #3 {##3} {##4} {##5}
        }{ % FALSE BRANCH
        \moremath_delim_op_manuscale:NNnnnn #2 #3 {##2} {##3} {##4} {##5}
        }
      }
    }
  }{ % \cs_if_free:nTF #1 FALSE BRANCH
    \msg_warning:nnn { moremath } { delimop / already-defined-skip }
      {#1}
  }
}

\cs_generate_variant:Nn \moremath_new_delim_op_command:NNN {cNN}
```

(*End of definition for* `\moremath_new_delim_op_command:NNN`*. This function is documented on page 24.*)

## 7.1 Document Level Commands

`\DeclareDelimitedOperator`

Finally provide the user with a command to declare an additional delimited operator.

```
\NewDocumentCommand\DeclareDelimitedOperator { m m m }
{
  \msg_redirect_name:nnn { moremath } { delimop / already-defined-skip } { error }
  \moremath_new_delim_op_command:NNN #1 #2 #3
  \msg_redirect_name:nnn { moremath } { delimop / already-defined-skip } {}
}
```

(*End of definition for* `\DeclareDelimitedOperator`*. This function is documented on page 8.*)

As amsmath [The23] pre-defines the following operators it is only sensible to also define delimited versions of them:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| `\arccos` | arccos | `\deg` | deg | `\lg` | lg | `\projlim` | proj lim |
| `\arcsin` | arcsin | `\det` | det | `\lim` | lim | `\sec` | sec |
| `\arctan` | arctan | `\dim` | dim | `\liminf` | lim inf | `\sin` | sin |
| `\arg` | arg | `\exp` | exp | `\limsup` | lim sup | `\sinh` | sinh |
| `\cos` | cos | `\gcd` | gcd | `\ln` | ln | `\sup` | sup |
| `\cosh` | cosh | `\hom` | hom | `\log` | log | `\tan` | tan |
| `\cot` | cot | `\inf` | inf | `\max` | max | `\tanh` | tanh |
| `\coth` | coth | `\injlim` | inj lim | `\min` | min | | |
| `\csc` | csc | `\ker` | ker | `\Pr` | Pr | | |

| | | | |
|---|---|---|---|
| `\varinjlim` | $\varinjlim$ | `\varliminf` | $\varliminf$ |
| `\varprojlim` | $\varprojlim$ | `\varlimsup` | $\varlimsup$ |

`\__moremath_new_delim_op_cmds:nN`

`\__moremath_new_delim_op_cmds:nN` creates document level macros of the form `\`⟨*prefix*⟩⟨*op name*⟩. It declares fife versions `\p`⟨*op name*⟩, `\b`⟨*op name*⟩, `\B`⟨*op name*⟩, `\v`⟨*op name*⟩ and `\V`⟨*op name*⟩.

The function takes two arguments:

`#1` : A token list which contains ⟨*op name*⟩

`#2` : The ⟨*csname*⟩ of the operator to use.

```
\cs_new_protected:Nn \__moremath_new_delim_op_cmds:nN
{
  \moremath_new_delim_op_command:cNN {p #1} #2 \__moremath_inparent:w
  \moremath_new_delim_op_command:cNN {b #1} #2 \__moremath_inbrack:w
  \moremath_new_delim_op_command:cNN {B #1} #2 \__moremath_inbrace:w
  \moremath_new_delim_op_command:cNN {v #1} #2 \__moremath_in_vert:w
  \moremath_new_delim_op_command:cNN {V #1} #2 \__moremath_in_Vert:w
}
```

(*End of definition for* `\__moremath_new_delim_op_cmds:nN`.)

The decision if the following macros are defined depends on a package load time option.

```
\bool_if:NTF \l__moremath_predef_operators_bool
{
```

We define the commands for the operators already declared by amsmath.

`\parccos`
`\barccos`
`\Barccos`
`\varccos`
`\Varccos`

For `\arccos`,

```
\__moremath_new_delim_op_cmds:nN {arccos} \arccos
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

`\parcsin`
`\barcsin`
`\Barcsin`
`\varcsin`
`\Varcsin`

`\arcsin`,

```
\__moremath_new_delim_op_cmds:nN {arcsin} \arcsin
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

`\parctan`
`\barctan`
`\Barctan`
`\varctan`
`\Varctan`

`\arctan`,

```
\__moremath_new_delim_op_cmds:nN {arctan} \arctan
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

`\parg`
`\barg`
`\Barg`
`\varg`
`\Varg`

`\arg`

```
\__moremath_new_delim_op_cmds:nN {arg} \arg
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pcos`
`\bcos`
`\Bcos`
`\vcos`
`\Vcos`

`\cos`

```
\__moremath_new_delim_op_cmds:nN {cos} \cos
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pcosh`
`\bcosh`
`\Bcosh`
`\vcosh`
`\Vcosh`

`\cosh`,

```
\__moremath_new_delim_op_cmds:nN {cosh} \cosh
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pcot`
`\bcot`
`\Bcot`
`\vcot`
`\Vcot`

`\cot`,

```
\__moremath_new_delim_op_cmds:nN {cot} \cot
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pcoth`
`\bcoth`
`\Bcoth`
`\vcoth`
`\Vcoth`

`\coth`,

```
\__moremath_new_delim_op_cmds:nN {coth} \coth
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pcsc`
`\bcsc`
`\Bcsc`
`\vcsc`
`\Vcsc`

`\csc`,

```
\__moremath_new_delim_op_cmds:nN {csc} \csc
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pdeg`
`\bdeg`
`\Bdeg`
`\vdeg`
`\Vdeg`

`\deg`,

```
\__moremath_new_delim_op_cmds:nN {deg} \deg
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pdet`
`\bdet`
`\Bdet`
`\vdet`
`\Vdet`

`\det`,

```
\__moremath_new_delim_op_cmds:nN {det} \det
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pdim`
`\bdim`
`\Bdim`
`\vdim`
`\Vdim`

`\dim`,

```
\__moremath_new_delim_op_cmds:nN {dim} \dim
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pexp`
`\bexp`
`\Bexp`
`\vexp`
`\Vexp`

`\exp`,

```
\__moremath_new_delim_op_cmds:nN {exp} \exp
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\pgcd`
`\bgcd`
`\Bgcd`
`\vgcd`
`\Vgcd`

`\gcd`,

```
\__moremath_new_delim_op_cmds:nN {gcd} \gcd
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

`\phom`
`\bhom`
`\Bhom`
`\vhom`
`\Vhom`

`\hom`,

```
\__moremath_new_delim_op_cmds:nN {hom} \hom
```

(*End of definition for* `\parccos`. *This function is documented on page 7.*)

\pinf
\binf
\Binf
\vinf
\Vinf

`\inf`,

```
\__moremath_new_delim_op_cmds:nN {inf} \inf
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pinjlim
\binjlim
\Binjlim
\vinjlim
\Vinjlim

`\injlim`,

```
\__moremath_new_delim_op_cmds:nN {injlim} \injlim
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pker
\bker
\Bker
\vker
\Vker

`\ker`,

```
\__moremath_new_delim_op_cmds:nN {ker} \ker
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\plg
\blg
\Blg
\vlg
\Vlg

`\lg`,

```
\__moremath_new_delim_op_cmds:nN {lg} \lg
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\plim
\blim
\Blim
\vlim
\Vlim

`\lim`,

```
\__moremath_new_delim_op_cmds:nN {lim} \lim
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pliminf
\bliminf
\Bliminf
\vliminf
\Vliminf

`\liminf`,

```
\__moremath_new_delim_op_cmds:nN {liminf} \liminf
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\plimsup
\blimsup
\Blimsup
\vlimsup
\Vlimsup

`\limsup`,

```
\__moremath_new_delim_op_cmds:nN {limsup} \limsup
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pln
\bln
\Bln
\vln
\Vln

`\ln`

```
\__moremath_new_delim_op_cmds:nN {ln} \ln
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\plog
\blog
\Blog
\vlog
\Vlog

`\log`,

```
\__moremath_new_delim_op_cmds:nN {log} \log
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pmax
\bmax
\Bmax
\vmax
\Vmax

`\max`,

```
\__moremath_new_delim_op_cmds:nN {max} \max
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pmin
\bmin
\Bmin
\vmin
\Vmin

`\min`,

```
\__moremath_new_delim_op_cmds:nN {min} \min
```

(*End of definition for* `\parccos`*. This function is documented on page* 7.)

\pPr
\bPr
\BPr
\vPr
\VPr

\Pr,

```
\__moremath_new_delim_op_cmds:nN {Pr} \Pr
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\pprojlim
\bprojlim
\Bprojlim
\vprojlim
\Vprojlim

\projlim,

```
\__moremath_new_delim_op_cmds:nN {projlim} \projlim
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\psec
\bsec
\Bsec
\vsec
\Vsec

\sec,

```
\__moremath_new_delim_op_cmds:nN {sec} \sec
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\psin
\bsin
\Bsin
\vsin
\Vsin

\sin,

```
\__moremath_new_delim_op_cmds:nN {sin} \sin
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\psinh
\bsinh
\Bsinh
\vsinh
\Vsinh

\sinh,

```
\__moremath_new_delim_op_cmds:nN {sinh} \sinh
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\psup
\bsup
\Bsup
\vsup
\Vsup

\sup,

```
\__moremath_new_delim_op_cmds:nN {sup} \sup
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\ptan
\btan
\Btan
\vtan
\Vtan

\tan,

```
\__moremath_new_delim_op_cmds:nN {tan} \tan
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\ptanh
\btanh
\Btanh
\vtanh
\Vtanh

and \tanh.

```
\__moremath_new_delim_op_cmds:nN {tanh} \tanh
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

\pvarinjlim
\bvarinjlim
\Bvarinjlim
\vvarinjlim
\Vvarinjlim
\pvarprojlim
\bvarprojlim
\Bvarprojlim
\vvarprojlim
\Vvarprojlim
\pvarliminf
\bvarliminf
\Bvarliminf
\vvarliminf
\Vvarliminf
\pvarlimsup
\bvarlimsup
\Bvarlimsup
\vvarlimsup
\Vvarlimsup

We also provide delimited versions of \varinjlim, \varprojlim, \varliminf, and \varlimsup.

```
\__moremath_new_delim_op_cmds:nN {varinjlim} \varinjlim
\__moremath_new_delim_op_cmds:nN {varprojlim} \varprojlim
\__moremath_new_delim_op_cmds:nN {varliminf} \varliminf
\__moremath_new_delim_op_cmds:nN {varlimsup} \varlimsup
```

(*End of definition for* \parccos*. This function is documented on page* 7*.*)

```
}{
  \msg_info:nnnn {moremath} {load /disabling} {no-operators}
  {
    predefined~delimited~operator~macros
  }
} % End of the conditional
```

# 8 Vector Calculus Macros

For providing macros which help with vector differentials, we first need some setup functions.

## 8.1 Macros Providing Symbols of Operators

`\__moremath_maybe_vcenter:n`

Sometimes a symbol should be centered explicitly, as this will depend on the current setting of `vcenter`, i.e. the current value of `\l__moremath_vcenter_bool`, we provide a helper function which puts its argument

`#1` : A list of ⟨*`tokens`*⟩

inside a `\vcenter` by means of `\moremath_vcenter:n` or not depending on the current value of `\l__moremath_vcenter_bool`.

```
\cs_new_protected_nopar:Nn \__moremath_maybe_vcenter:n
{
  \bool_if:NTF \l__moremath_vcenter_bool
  {
    \moremath_vcenter:n {#1}
  }{
    #1
  }
}
```

(*End of definition for* `\__moremath_maybe_vcenter:n`.)

`\__moremath_gradient_operator_get:`

This function returns the gradient operator depending on the current setting of the keys.

```
\cs_new_protected:Nn \__moremath_gradient_operator_get:
{
  \tl_if_empty:NTF \l__moremath_grad_op_tl
  {
    \__moremath_tex_mathop:n
    {
```

Otherwise we first need to check if the operator shall have an arrow over it. Afterwards if the nabla shall be bold.

```
      \bool_if:NTF \l__moremath_nabla_arrow_bool
      {
        \vec
        {
```

In case `\l__moremath_nabla_arrow_bool` is true we should *maybe* center the symbol.

```
          \__moremath_maybe_vcenter:n
          {
            \bool_if:NT \l__moremath_nabla_bold_bool
            {
              \boldsymbol
            }
            \l__moremath_nabla_tl
          }
        }
      }{
```

Like in the case above we should *maybe* center the symbol if `\l__moremath_nabla_bold_bool` is true.

```
        \bool_if:NTF \l__moremath_nabla_bold_bool
        {
          \__moremath_maybe_vcenter:n
          {
            \boldsymbol
            \l__moremath_nabla_tl
          }
        }{
          \l__moremath_nabla_tl
        }
      }
    }% \__moremath_tex_mathop:n
    \nolimits
  }{
```

If the user provided its own implementation of the operator, we simply return it.

```
    \l__moremath_grad_op_tl
  }
}
```

(*End of definition for* `\__moremath_gradient_operator_get:`*.*)

`\__moremath_laplace_operator_get:`

Then we define a function for returning the laplace operator symbol, depending on the currently set keys. The function wraps the symbol for the operator inside `\mathop` to provide the right spacing.

```
\cs_new_protected:Nn \__moremath_laplace_operator_get:
{
  \tl_if_empty:NTF \l__moremath_laplacian_tl
  {
    \__moremath_tex_mathop:n
    {
      \bool_if:NTF \l__moremath_laplacian_delta_bool
      {
        \Delta
      }{
        \bool_if:NTF \l__moremath_laplacian_arrow_bool
        {
          \vec{
            \__moremath_maybe_vcenter:n
            {
              \bool_if:NT \l__moremath_nabla_bold_bool {\boldsymbol}
              \l__moremath_laplacian_symb_tl
            }
          }
        }{
          \bool_if:NTF \l__moremath_nabla_bold_bool
          {
            \__moremath_maybe_vcenter:n
            {
              \boldsymbol
              \l__moremath_laplacian_symb_tl
            }
          }{
            \l__moremath_laplacian_symb_tl
          }
        }
      }
    }\nolimits
    \bool_if:NF \l__moremath_laplacian_delta_bool
    {
      \c_math_superscript_token
      {
        2
      }
    }
  }{
    \l__moremath_laplacian_tl
  }
}
```

(*End of definition for* `\__moremath_laplace_operator_get:`*.*)

`\__moremath_dalembert_operator_get:`

This function returns the d'Alembert operator depending on the currently set keys. The symbol for the d'Alembert operator is wrapped inside `\mathop` to provide proper spacing.

```
\cs_new_protected:Nn \__moremath_dalembert_operator_get:
{
  \__moremath_tex_mathop:n
  {
    \l__moremath_dalembert_symb_tl
  }% \__moremath_tex_mathop:n
  \nolimits
}
```

(*End of definition for* `\__moremath_dalembert_operator_get:`*.*)

## 8.2 Macros Producing an Operator

After we have defined the symbols it is now time to provide a function which produces the entire gradient operator

`\moremath_gradient_operator:n`
**`\moremath_laplace_operator:n`**

This function takes one arguments, the subscript to use with the operator.

```
\cs_new_protected_nopar:Nn \moremath_gradient_operator:n
{
  \__moremath_gradient_operator_get:
  \tl_if_empty:nF {#1} {\c_math_subscript_token {#1}}
}
```

Like for the gradient operator we do the same for the laplacian

```
\cs_new_protected_nopar:Nn \moremath_laplace_operator:n
{
  \__moremath_laplace_operator_get:
  \tl_if_empty:nF {#1} {\c_math_subscript_token {#1}}
}
```

(*End of definition for* `\moremath_gradient_operator:n` *and* `\moremath_laplace_operator:n`*. These functions are documented on page 24.*)

`\moremath_divergence_operator:n`
`\moremath_curl_operator:n`

Using the already defined gradient operator it is possible to define a function which acts as an operator suitable for representing the divergence operator and the curl operator. Like the gradient operator this functions take one argument

`#1` : the subscript of the gradient operator.

```
\cs_new_protected_nopar:Nn \moremath_divergence_operator:n
{
```

The braces around `\moremath_gradient_operator:n` are necessary to avoid issues with the spacing between `\cdot` and the following `\mathopen` from any braces.[3] Example:

$$\text{With braces:}\quad {\nabla} \cdot (f(x)),\qquad \text{without braces:}\quad \nabla \cdot\!(f(x))$$

```
  {
    \moremath_gradient_operator:n {#1}
  }
  \cdot
}

\cs_new_protected_nopar:Nn \moremath_curl_operator:n
{
  {
    \moremath_gradient_operator:n {#1}
  }
  \times
}
```

(*End of definition for* `\moremath_divergence_operator:n` *and* `\moremath_curl_operator:n`*. These functions are documented on page 24.*)

`\moremath_dalembert_operator:n`

This function produces the d'Alembert operator, including an optional subscript. The function takes one argument:

`#1` : A ⟨*tl*⟩ with the contents of the subscript of the operator.
This variable may be empty, in this case no subscript (not even an empty one) is produced.

```
\cs_new_protected:Nn \moremath_dalembert_operator:n
{
  \__moremath_dalembert_operator_get:
  \tl_if_empty:nF {#1}
  {
    \c_math_subscript_token {#1}
  }
}
```

(*End of definition for* `\moremath_dalembert_operator:n`*. This function is documented on page 24.*)

## 8.3 Producing Delimited Vector Calculus Operators

`\moremath_delim_nabla_op_noscale:NNnn`
`\moremath_delim_nabla_op_autoscale:NNnn`

These functions produce a vector calculus operator with ⟨*contents*⟩ inside delimiters. The `autoscale` variant scales the delimiters with the size of ⟨*contents*⟩, the `noscale` variant does no scaling at all.

---

[3] See: https://tex.stackexchange.com/a/223914

#1 : The ⟨*csname*⟩ of the operator to use.
This should have the same form as `\moremath_gradient_operator:n`, i.e. the function passed as ⟨*csname*⟩ should accept one argument to typeset as subscript.
#2 : The ⟨*csname*⟩ of the *paired delimiter* to use.
The paired delimiter has to be one declared using mathtools' `\DeclarePairedDelimiter` command.
#3 : A ⟨*tl*⟩ with the subscript of the operator.
#4 : A ⟨*tl*⟩ with the contents to typeset inside the delimiters

We begin with the `noscale` version.

```
\cs_new_protected_nopar:Nn \moremath_delim_nabla_op_noscale:NNnn
{
  #1 {#3} #2{#4}
}
```

Then we create the version with automatic scaling.

```
\cs_new_protected_nopar:Nn \moremath_delim_nabla_op_autoscale:NNnn
{
  #1 {#3} #2 * {#4}
}
```

(*End of definition for* `\moremath_delim_nabla_op_noscale:NNnn` *and* `\moremath_delim_nabla_op_autoscale:NNnn`*. These functions are documented on page 24.*)

`\moremath_delim_nabla_op_manuscale:NNnnn`

This function produces a vector calculus operator like `\moremath_delim_nabla_op_noscale:NNnn` and `\moremath_delim_nabla_op_autoscale:NNnn`, but with manual scaling. It takes five arguments:
#1 : The ⟨*csname*⟩ of the operator to use.
#2 : The ⟨*csname*⟩ of the paired delimiter.
#3 : A ⟨*token list*⟩ containing the ⟨*scale cmd*⟩ like `\big`, `\Big`, `\bigg`, etc.
#4 : A ⟨*tl*⟩ with the subscript of the operator.
#5 : A ⟨*tl*⟩ with the contents to typeset inside the delimiters.

```
\cs_new_protected_nopar:Nn \moremath_delim_nabla_op_manuscale:NNnnn
{
  #1 {#4} #2 [#3] {#5}
}
```

`\moremath_delim_nabla_op_manuscale:NNVnn`

We also declare a variant for passing a variable with the ⟨*scale cmd*⟩ instead of a ⟨*token list*⟩.

```
\cs_generate_variant:Nn \moremath_delim_nabla_op_manuscale:NNnnn {NNVnn}
```

(*End of definition for* `\moremath_delim_nabla_op_manuscale:NNnnn` *and* `\moremath_delim_nabla_op_manuscale:NNVnn`*. These functions are documented on page 24.*)

To ease the definition of those macros we define a function for defining the delimited versions.

Of course we also need a way to declare a user interface for this functions. For this purpose we first need some helpers for setting necessary parameters.

`\__moremath_parse_kv_args:nN`

This helper macro takes two arguments,
#1 : A token list of the key-value arguments.
#2 : The ⟨*csname*⟩ of a token list to put the scale value in.

The function sets the given keys but before it does so it searches for a key named `scale` and puts it into the second argument. For this to work it is necessary that all values have `=` in them.

```
\cs_new_protected:Nn \__moremath_parse_kv_args:nN
{
```

We first put the key-value arguments inside a property list. Afterwards we check if the key `scale` has been given. If yes we pop it and assign it to the second argument. Otherwise we do nothing.

```
  \prop_set_from_keyval:Nn \l_tmpa_prop {#1}

  \prop_pop:NnNT \l_tmpa_prop {scale} #2 {}

  \keys_set:ne {moremath} {\prop_to_keyval:N \l_tmpa_prop}
}
```

(*End of definition for* `\__moremath_parse_kv_args:nN`.)

We also need a warning message for conflicting arguments, to inform the user that one of his options is going to be ignored.

```
\msg_new:nnn { moremath } { vector-calc / scale-star-conflict }
{
  Both~star~and~scaling~factor~given~to~'#1'.\\
  Automatic~scaling~will~be~preferred,~the~size~command~'#2'~will~be~
  ignored~\msg_line_context:.
}
```

`\__moremath_new_delim_nabla_doc_cmd:NNN`
`\__moremath_new_delim_nabla_doc_cmd:cNN`

This function takes three arguments:

`#1` : The ⟨*csname*⟩ of the to be defined command

`#2` : The ⟨*csname*⟩ of the operator to use.
This should be a function like `\moremath_gradient_operator:n`.

`#3` : The ⟨*csname*⟩ of the delimiter function to use.
This should be a macro/command created with mathtools `\DeclarePairedDelimiter` command.

Its purpose is to create a new document level command, for the delimited vector calculus operators.

```
\cs_new_protected:Nn \__moremath_new_delim_nabla_doc_cmd:NNN
{
  \cs_if_free:NTF #1
  {
    \exp_args:NNe \NewDocumentCommand #1
    {
      s ={scale} o E{ \char_generate:nn {`_}{8} }{ {} } m
    }
    { % command code
      \group_begin:
      % optional arguments given?
      \tl_if_novalue:nF {##2}
      {
        \__moremath_parse_kv_args:nN {##2} \l_tmpa_tl
      }
      % star given?
      \bool_if:nTF {##1}
      {
        % scale factor given?
        \tl_if_empty:NF \l_tmpa_tl
        {
          \msg_warning:nnnV { moremath } { vector-calc / scale-star-conflict }
            {#1} \l_tmpa_tl
        }
        \moremath_delim_nabla_op_autoscale:NNnn #2 #3 {##3} {##4}
      }{ % \bool_if:nTF {##1} FALSE BRANCH
        % scale factor given?
        \tl_if_empty:NTF \l_tmpa_tl
        {
          \moremath_delim_nabla_op_noscale:NNnn #2 #3 {##3} {##4}
        }{ % FALSE BRANCH
          \moremath_delim_nabla_op_manuscale:NNVnn #2 #3 \l_tmpa_tl {##3} {##4}
        }
      } % \bool_if:nTF {##1}
      \group_end:
    }
  }{ % \cs_if_free:NTF #1 FALSE BRANCH
    \msg_warning:nnn { moremath } { vector-calc / already-defined-skip } {#1}
  } % \cs_if_free:NTF #1
}
%
\cs_generate_variant:Nn \__moremath_new_delim_nabla_doc_cmd:NNN {cNN}
```

(*End of definition for* `\__moremath_new_delim_nabla_doc_cmd:NNN`.)

`\__moremath_new_nabla_doc_cmds:nN`

This internal function creates five different document level commands at once, using `\__moremath_new_delim_nabla_doc_cmd:cNN`. It takes two arguments:

`#1` : The a ⟨*suffix tl*⟩ for constructing the command names.
The resulting commands will have the form `\p`⟨*suffix*⟩, `\b`⟨*suffix*⟩, `\B`⟨*suffix*⟩, `\v`⟨*suffix*⟩ and `\V`⟨*suffix*⟩.

`#2` : The ⟨*csname*⟩ of the operator to use

```
\cs_new_protected:Nn \__moremath_new_nabla_doc_cmds:nN
{
  \__moremath_new_delim_nabla_doc_cmd:cNN { p #1 } #2 \__moremath_inparent:w
  \__moremath_new_delim_nabla_doc_cmd:cNN { b #1 } #2 \__moremath_inbrack:w
  \__moremath_new_delim_nabla_doc_cmd:cNN { B #1 } #2 \__moremath_inbrace:w
  \__moremath_new_delim_nabla_doc_cmd:cNN { v #1 } #2 \__moremath_in_vert:w
  \__moremath_new_delim_nabla_doc_cmd:cNN { V #1 } #2 \__moremath_in_Vert:w
}
```

(*End of definition for* `\__moremath_new_nabla_doc_cmds:nN`.)

## 8.4 Document Level Commands

The predefined macros for vector calculus are also guarded by a package option to be conditionally disabled by the user.

```
\bool_if:NTF \l__moremath_predef_vector_op_bool
{
```

### 8.4.1 Standalone Operators

The user might want to use also a non-delimited version of the vector calculus operators, we provide them with a standalone version of those.

As the definition of a new document command can fail if the ⟨*csname*⟩ clashes with some already defined macro, we define an error message to use when defining document level commands.

```
\msg_new:nnn { moremath } { vector-calc / already-defined-skip }
{
  Control~sequence~’#1’~is~already~defined.\\
  Skipping~definition~\msg_line_context:.
}
```

`\grad`
`\divergence`
`\curl`
`\laplacian`

Now we provide the user with document level commands for `\moremath_`⟨*op*⟩`_operator:n`.

```
\cs_if_free:NTF \grad
{
  \exp_args:NNe \NewDocumentCommand \grad { !o E{ \char_generate:nn {‘_}{8} }{{}} }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn {moremath} {#1}
    }
    \moremath_gradient_operator:n {#2}
    \group_end:
  }
}{
  \msg_warning:nnn {moremath} { vector-calc / already-defined-skip } {\grad}
}

\cs_if_free:NTF \divergence
{
  \exp_args:NNe \NewDocumentCommand \divergence
  { !o E{ \char_generate:nn {‘_} {8} }{{}} }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn {moremath} {#1}
    }
    \moremath_divergence_operator:n {#2}
    \group_end:
  }
}{
  \msg_warning:nnn {moremath} { vector-calc / already-defined-skip } {\divergence}
}

\cs_if_free:NTF \curl
{
  \exp_args:NNe \NewDocumentCommand \curl
    { !o E{ \char_generate:nn {‘_}{8} }{{}} }
  {
    \group_begin:
```

```
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn {moremath} {#1}
    }
    \moremath_curl_operator:n {#2}
    \group_end:
  }
}{
  \msg_warning:nnn {moremath} {vector-calc / already-defined-skip} {\curl}
}

\cs_if_free:NTF \laplacian
{
  \exp_args:NNe \NewDocumentCommand \laplacian
  { !o E{ \char_generate:nn {`_}{8} }{{}} }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn {moremath} {#1}
    }
    \moremath_laplace_operator:n {#2}
    \group_end:
  }
}{
  \msg_warning:nnn {moremath} { vector-calc / already-defined-skip }
  {\laplacian}
}
```

We now also define a command to use as a standalone d'Alembert operator. As the name `\dalembertian` is a bit cumbersome to type out, I've decided to use one of its alternate names `\quabla`

`\quabla`

```
\cs_if_free:NTF \quabla
{
  \exp_args:NNe \NewDocumentCommand \quabla
    { !o E{ \char_generate:nn {`_} { 8 } }{{}} }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath } {#1}
    }
    \moremath_dalembert_operator:n {#2}
    \group_end:
  }
}{ % \cs_if_free:NTF \quabla FALSE BRANCH
  \msg_warning:nnn { moremath } { vector-calc / already-defined-skip }
    {\quabla}
}
```

(*End of definition for* `\grad` *and others. These functions are documented on page 9.*)

### 8.4.2 Operators with Delimiters

`\pgrad` `\bgrad` `\Bgrad` `\vgrad` `\Vgrad`

Now lets declare the delimited gradient operators. We provide five versions using parenthesis, brackets, braces, single `\vert`, and double `\Vert`.

```
\__moremath_new_nabla_doc_cmds:nN {grad} \moremath_gradient_operator:n
```

(*End of definition for* `\pgrad` *and others. These functions are documented on page 10.*)

`\pdiv` `\bdiv` `\Bdiv` `\vdiv` `\Vdiv`

Now we do the same for the divergence operator.

```
\__moremath_new_nabla_doc_cmds:nN {div} \moremath_divergence_operator:n
```

(*End of definition for* `\pdiv` *and others. These functions are documented on page 12.*)

`\pcurl` `\bcurl` `\Bcurl` `\vcurl` `\Vcurl`

Now to the curl macros.

```
\__moremath_new_nabla_doc_cmds:nN {curl} \moremath_curl_operator:n
```

(*End of definition for* `\pcurl` *and others. These functions are documented on page 13.*)

`\plaplacian` `\blaplacian` `\Blaplacian` `\vlaplacian` `\Vlaplacian`

Next we take care of the laplacian

```
\__moremath_new_nabla_doc_cmds:nN {laplacian} \moremath_laplace_operator:n
```

(*End of definition for* `\plaplacian` *and others. These functions are documented on page 14.*)

`\pquabla` `\bquabla` `\Bquabla` `\vquabla` `\Vquabla`

Finally we define delimited commands for the d'Alembert operator.

```
\__moremath_new_nabla_doc_cmds:nN {quabla} \moremath_dalembert_operator:n
```

(*End of definition for* `\pquabla` *and others. These functions are documented on page 15.*)

If the user issued `no-vector` as a package loading option, write this to the log file.

```
}{ % IF NOT \l__moremath_predef_vector_op_bool
  \msg_info:nnnn {moremath} { load /disabling } {no-vector}
  {
    predefined~vector~calculus~macros
  }
} % END \l__moremath_predef_vector_op_bool
```

# 9 Macros Producing Matrices and Vectors

## 9.1 Producing Row and Column Vectors

The functions in this section are used to generate row and column vectors utilizing mathtools [Høg+24] `matrix*` and `smallmatrix*` environments.

`\l__moremath_vector_entries_seq`

For generating row or column vectors it is necessary to store the entries inside a variable. `\l__moremath_vector_entries_seq` is used for this purpose.

```
\seq_new:N \l__moremath_vector_entries_seq
```

(*End of definition for* `\l__moremath_vector_entries_seq`*.*)

Then we need a function for formatting the actual entries of the vector. We need two version one for the row vector and one for the column vector version.

`\__moremath_seq_to_column_vector:N` `\__moremath_seq_to_row_vector:N`

These functions take one argument

`#1` : A sequence which should be converted to the contents of the single column/row matrix.

They format the input suitable to be put inside a `matrix*` environment. We begin with the column vector version.

```
\cs_new_protected_nopar:Nn \__moremath_seq_to_column_vector:N
{
  \seq_use:Nn #1 {\\}
}
```

Then we get to the row vector.

```
\cs_new_protected_nopar:Nn \__moremath_seq_to_row_vector:N
{
  \seq_use:Nn #1 {&}
}
```

(*End of definition for* `\__moremath_seq_to_column_vector:N` *and* `\__moremath_seq_to_row_vector:N`*.*)

In the next step we construct the single row/column matrix from the user provided input.

`\moremath_column_vector:nn`
`\moremath_column_vector:Vn`
`\moremath_row_vector:nn`
`\moremath_row_vector:Vn`

The two commands `\moremath_column_vector:nn` and `\moremath_row_vector:nn` construct the matrix, they both take two arguments.

`#1` : The delimiter specifier.
This should be one of the prefixes of the ⟨*prefix*⟩`matrix*`, environments.
Another possibility is to issue `small` as this parameter to get an inline matrix.

`#2` : The comma separated contents of the matrix.

```
\cs_new_protected_nopar:Nn \moremath_column_vector:nn
{
  \seq_clear:N \l__moremath_vector_entries_seq
  \seq_set_from_clist:Nn \l__moremath_vector_entries_seq {#2}

  \__moremath_matrix_star_begin:nV {#1} \l__moremath_matrix_align_tl
    \__moremath_seq_to_column_vector:N \l__moremath_vector_entries_seq
  \__moremath_matrix_star_end:n {#1}
}
%
\cs_new_protected_nopar:Nn \moremath_row_vector:nn
{
  \seq_clear:N \l__moremath_vector_entries_seq
  \seq_set_from_clist:Nn \l__moremath_vector_entries_seq {#2}

  \__moremath_matrix_star_begin:nV {#1} \l__moremath_matrix_align_tl
    \__moremath_seq_to_row_vector:N \l__moremath_vector_entries_seq
  \__moremath_matrix_star_end:n {#1}
}
```

We also provide a `Vn` variant of those functions.

```
\cs_generate_variant:Nn \moremath_column_vector:nn { Vn }
\cs_generate_variant:Nn \moremath_row_vector:nn { Vn }
```

(*End of definition for* `\moremath_column_vector:nn` *and* `\moremath_row_vector:nn`*. These functions are documented on page 25.*)

`\moremath_column_smallvector:nn`
`\moremath_column_smallvector:Vn`
`\moremath_row_smallvector:nn`
`\moremath_row_smallvector:Vn`

To make the code more readable, we define functions specifically for creating row and column vectors using the `smallmatrix*` family of environments. These functions take the same arguments as the non-small versions above.

```
\cs_new_protected_nopar:Nn \moremath_column_smallvector:nn
```

```
{
  \moremath_column_vector:nn {#1 small} {#2}
}

\cs_new_protected_nopar:Nn \moremath_row_smallvector:nn
{
  \moremath_row_vector:nn {#1 small} {#2}
}
% Variants
\cs_generate_variant:Nn \moremath_column_smallvector:nn { Vn }
\cs_generate_variant:Nn \moremath_row_smallvector:nn { Vn }
```

(*End of definition for* `\moremath_column_smallvector:nn` *and* `\moremath_row_smallvector:nn`*. These functions are documented on page 25.*)

## 9.2 Shorthands for Simple Matrices

The construction of several simple matrices like diagonal matrices, can be simplified as there is no need to use the `matrix` environment.[4]

`\l__moremath_mat_diag_entries_seq`
`\l__moremath_mat_row_entries_seq`

We construct the matrices row by row, therefore we need to store the currently constructed row inside a variable. The same is true for the diagonal entries which also need to be stored somewhere. We therefore declare two sequence variables `\l__moremath_mat_diag_entries_seq` and `\l__moremath_mat_row_seq` for this purpose

```
\seq_clear_new:N \l__moremath_mat_diag_entries_seq
\seq_clear_new:N \l__moremath_mat_row_entries_seq
```

(*End of definition for* `\l__moremath_mat_diag_entries_seq` *and* `\l__moremath_mat_row_entries_seq`*.*)

### 9.2.1 (Anti-)diagonal matrices

We split the construction of the matrix into multiple parts, utilizing internal helper functions.

`\__moremath_constr_diagmat_row:n`
`\__moremath_constr_antidiagmat_row:n`

`\__moremath_constr_diagmat_row:n` and `\__moremath_constr_antidiagmat_row:n` take one integer argument:
`#1` : The number of the current row.
They both construct a matrix row and place it inside the input stream. They take the content of the (anti-)diagonal from the variable `\l__moremath_mat_diag_entries_seq` and use the variable `\l__moremath_mat_row_entries_seq` to store the current row.

```
\cs_new_protected:Nn \__moremath_constr_diagmat_row:n
{
  \seq_clear:N \l__moremath_mat_row_entries_seq
```

As all diagonal matrices $M \in A^{m\times n}$, where $A$ is a field, are quadratic i.e. $A^{m\times n} \equiv A^{n\times n}$ the length of the diagonal sequence equals the number of rows and columns of the matrix. We exploit this fact here.

```
  \int_step_inline:nn {\seq_count:N \l__moremath_mat_diag_entries_seq}
  {
    \int_compare:nTF { #1 == ##1 }
    {
```

---

[4] The code in this section is heavily inspired by the following answer on TEXSE: https://tex.stackexchange.com/a/539741

```
      \seq_put_right:Nx \l__moremath_mat_row_entries_seq
      {
        \seq_item:Nn \l__moremath_mat_diag_entries_seq { #1 }
      }
    }{ % false branch
      \seq_put_right:NV \l__moremath_mat_row_entries_seq \l__moremath_matrix_fill_tl
    } % \int_compare:nTF { #1 == ##1 }
  }
  \seq_use:Nn \l__moremath_mat_row_entries_seq { & } \\
}

% Anti-diagonal version
\cs_new_protected:Nn \__moremath_constr_antidiagmat_row:n
{
  \seq_clear:N \l__moremath_mat_row_entries_seq
  \int_step_inline:nn {\seq_count:N \l__moremath_mat_diag_entries_seq}
  {
    \int_compare:nTF { #1 == ##1 }
    {
      % as this is a anti diagonal matrix we put in the elements from the
      % left so that the first entry is the right most entry
      \seq_put_left:Nx \l__moremath_mat_row_entries_seq
      {
        \seq_item:Nn  \l__moremath_mat_diag_entries_seq { #1 }
      }
    }{ % false branch
      \seq_put_left:NV \l__moremath_mat_row_entries_seq \l__moremath_matrix_fill_tl
    } % \int_compare:nTF { #1 == ##1 }
  }
  \seq_use:Nn \l__moremath_mat_row_entries_seq { & } \\
}
```

(*End of definition for* `\__moremath_constr_diagmat_row:n` *and* `\__moremath_constr_antidiagmat_row:n`*.*)

`\moremath_diagonal_matrix:nn`
`\moremath_antidiagonal_matrix:nn`
`\moremath_diagonal_smallmatrix:nn`
`\moremath_antidiagonal_smallmatrix:nn`

The `\moremath_`⟨*a or d*⟩`_matrix:nn` and `\moremath_`⟨*a or d*⟩`_smallmatrix:nn` family of functions produce a matrix based on mathtools [Høg+24] `matrix*` environment. The functions take two arguments.

`#1` : The delimiter specifier.
This should be one of the prefixes of the ⟨*prefix*⟩`matrix*` environments.
`#2` : The comma separated contents of the (anti-)diagonal.

These functions also use the values of the variables `\l__moremath_matrix_fill_tl` and `\l__moremath_matrix_align_tl`. And modifies the variable `\l__moremath_mat_diag_entries_seq`.

```
\cs_new_protected:Nn \moremath_diagonal_matrix:nn
{
  \seq_set_from_clist:Nn \l__moremath_mat_diag_entries_seq { #2 }
  \__moremath_matrix_star_begin:nV { #1 } \l__moremath_matrix_align_tl
    \int_step_function:nN { \seq_count:N \l__moremath_mat_diag_entries_seq }
      \__moremath_constr_diagmat_row:n
  \__moremath_matrix_star_end:n { #1 }
}
%
% Anti-diagonal matrix
```

```
\cs_new_protected:Nn \moremath_antidiagonal_matrix:nn
{
  \seq_set_from_clist:Nn \l__moremath_mat_diag_entries_seq { #2 }
  \__moremath_matrix_star_begin:nV { #1 } \l__moremath_matrix_align_tl
    \int_step_function:nN { \seq_count:N \l__moremath_mat_diag_entries_seq }
      \__moremath_constr_antidiagmat_row:n
  \__moremath_matrix_star_end:n { #1 }
}

% Small versions
\cs_new_protected:Nn \moremath_diagonal_smallmatrix:nn
{
  \moremath_diagonal_matrix:nn {#1 small} {#2}
}

\cs_new_protected:Nn \moremath_antidiagonal_smallmatrix:nn
{
  \moremath_antidiagonal_matrix:nn {#1 small} {#2}
}
```

For convenience we also define some variants of the above functions.

```
\cs_generate_variant:Nn \moremath_diagonal_matrix:nn { nV, Vn, VV }
\cs_generate_variant:Nn \moremath_antidiagonal_matrix:nn { nV, Vn, VV }
\cs_generate_variant:Nn \moremath_diagonal_smallmatrix:nn { nV, Vn, VV}
\cs_generate_variant:Nn \moremath_antidiagonal_smallmatrix:nn { nV, Vn, VV }
```

**\moremath_diagonal_matrix:nV**
**\moremath_diagonal_matrix:Vn**
**\moremath_diagonal_matrix:VV**
\moremath_antidiagonal_matrix:nV
\moremath_antidiagonal_matrix:Vn
\moremath_antidiagonal_matrix:VV
\moremath_diagonal_smallmatrix:nV
\moremath_diagonal_smallmatrix:Vn
\moremath_diagonal_smallmatrix:VV
\moremath_antidiagonal_smallmatrix:nV
\moremath_antidiagonal_smallmatrix:Vn
\moremath_antidiagonal_smallmatrix:VV

(*End of definition for* `\moremath_diagonal_matrix:nn` *and others. These functions are documented on page 25.*)

### 9.2.2 Identity Matrices

As we already have functions available for producing diagonal matrices, it makes only sense to also provide a shorthand for producing an identity matrix, i.e. a diagonal matrix with "1" along the diagonal.

__moremath_generate_one_filled_clist:Nn

The function `\__moremath_generate_one_filled_clist:Nn` produces a ⟨*clist*⟩, consisting only of "1" as entries. This function takes two arguments:
#1 : The *csname* of a ⟨*clist var*⟩ to store the ⟨*clist*⟩ into.
#2 : An ⟨*int*⟩ to represent the number of entries to produce.

```
\cs_new_protected_nopar:Nn \__moremath_generate_one_filled_clist:Nn
{
  \seq_clear:N \l_tmpa_seq
  \int_step_inline:nn {#2}
  {
    \seq_put_right:NV \l_tmpa_seq \c_one_int
  }
  \clist_set_from_seq:NN #1 \l_tmpa_seq
}
```

We also define a variant accepting an integer variable.

__moremath_generate_one_filled_clist:NV

```
\cs_generate_variant:Nn \__moremath_generate_one_filled_clist:Nn { N V }
```

(*End of definition for* `\__moremath_generate_one_filled_clist:Nn` *and* `\__moremath_generate_one_filled_clist:NV`.)

`\l__moremath_id_entries_clist`

As we want to utilize the `\moremath_diagonal_matrix:VV` and `\moremath_diagonal_smallmatrix:VV` functions for creating the identity matrix we declare an internal ⟨*clist var*⟩ called `\l__moremath_id_entries_clist` for passing the ⟨*clist*⟩ around.

```
\clist_new:N \l__moremath_id_entries_clist
```

(*End of definition for* `\l__moremath_id_entries_clist`.)

`\moremath_id_matrix:n`
`\moremath_id_smallmatrix:n`

These functions are intended to produce an identity matrix from an integer expression. They take one argument.
`#1` : The number of diagonal entries.

```
\cs_new_protected_nopar:Nn \moremath_id_matrix:n
{
  \clist_clear:N \l__moremath_id_entries_clist
  \__moremath_generate_one_filled_clist:Nn \l__moremath_id_entries_clist {#1}
  \moremath_diagonal_matrix:VV \l__moremath_matrix_delim_tl \l__moremath_id_entries_clist
}
```

```
\cs_new_protected_nopar:Nn \moremath_id_smallmatrix:n
{
  \clist_clear:N \l__moremath_id_entries_clist
  \__moremath_generate_one_filled_clist:Nn \l__moremath_id_entries_clist {#1}
  \moremath_diagonal_smallmatrix:VV \l__moremath_matrix_delim_tl \l__moremath_id_entries_cli
}
```

We also provide variants, which accepts a `V`-type argument:

`\moremath_id_matrix:V`
`\moremath_id_smallmatrix:V`

```
\cs_generate_variant:Nn \moremath_id_matrix:n { V }
\cs_generate_variant:Nn \moremath_id_smallmatrix:n { V }
```

(*End of definition for* `\moremath_id_matrix:n` *and others. These functions are documented on page 26.*)

## 9.3 Document Level Commands

Now we define document level commands for the previously defined functions.

But before we do so we define a message to be issued in case the targeted ⟨*csname*⟩ is already defined elsewhere.

```
\msg_new:nnnn { moremath } { matrix / already-defined-doc-cmd-skip }
{
  Control~sequence~'#1'~is~already~defined.\\
  Skipping~definition~\msg_line_context:.
}
{
  The~control~sequence~'#1'~has~already\\
  been~defined~by~some~other~package.\\
  And~I~am~refusing~to~overwrite~the~existing~definition,\\
  therefore~I~am~skipping~the~definition~of~this~command.
}
```

### 9.3.1 Row and Column Vectors

We begin with the row and column vector functions. As with the other document level commands, we guard the definitions with a key value option, so that the user can disable them.

```
\bool_if:nTF \l__moremath_predef_crvector_bool
{
```

\cvector
\rvector
\smallcvector
\smallrvector

First we define the document level command for the bare column vector

```
\cs_if_free:NTF \cvector
{
  \NewDocumentCommand \cvector { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_column_vector:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  % issue a warning message if the csname is already taken.
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
  {
    \cvector
  }
} % \cs_if_free:NTF \cvector
```

and the row vector.

```
\cs_if_free:NTF \rvector
{
  \NewDocumentCommand \rvector { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_row_vector:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  % warn if csname is already taken
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip}
  {
    \rvector
  }
} % \cs_if_free:NTF \rvector
```

Then we define the smaller inline versions of those commands.

```
\cs_if_free:NTF \smallcvector
{
  \NewDocumentCommand \smallcvector { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn {moremath / matrix} {#1}
    }
    \moremath_column_smallvector:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  % Issue a warning message if the csname is already taken
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
  {
    \smallcvector
  }
} % \cs_if_free:NTF \smallcvector

\cs_if_free:NTF \smallrvector
{
  \NewDocumentCommand \smallrvector { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_row_smallvector:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  % warn if csname is taken
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
  {
    \smallrvector
  }
}
```

(*End of definition for* `\cvector` *and others. These functions are documented on page 16.*)

**Commands with Pre-defined Delimiters** Next we define several shorthands to for the commonly used delimiters, to avoid code duplication, we first create some helper functions which define those functions.

`\__moremath_new_vector_shorth_doc_cmd:NNn`

The function `\__moremath_new_vector_shorth_doc_cmd` creates a new vector shorthand, command. It takes three arguments:

`#1` : The ⟨*csname*⟩ to be defined.

`#2` : The ⟨*function*⟩ to use for this shorthand.
This should be one of the `\moremath_`⟨*type*⟩`_`⟨*size*⟩`vector:Vn` like commands.

`#3` : The ⟨*delimiter*⟩ to use.
Usually one of `p`, `b`, `B`, `v`, `V`.

```
\cs_new_protected:Nn \__moremath_new_vector_shorth_doc_cmd:NNn
{
  \cs_if_free:NTF #1
  {
    \NewDocumentCommand #1 { o m }
    {
      \group_begin:
      % set the delimiter key pre-set for this function
      \keys_set:nn {moremath / matrix } {delimiter = #3}
      \tl_if_novalue:nF {##1}
      {
        \keys_set:nn {moremath / matrix } {##1}
      }
      #2 \l__moremath_matrix_delim_tl {##2}
      \group_end:
    }
  }{
    % warn if csname is taken
    \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
    {
      #1
    }
  }
}
```

(*End of definition for* `\__moremath_new_vector_shorth_doc_cmd:NNn`.)

`\pcvector`
`\bcvector`
`\Bcvector`
`\vcvector`
`\Vcvector`
`\prvector`
`\brvector`
`\Brvector`
`\vrvector`
`\Vrvector`

Now we define shorthands for all of the matrix types so that the user does not have to specify `delimiter=`⟨*delim*⟩ every time. We begin with the column vector.

```
% parenthesis
\__moremath_new_vector_shorth_doc_cmd:NNn \pcvector \moremath_column_vector:Vn {p}
% brackets
\__moremath_new_vector_shorth_doc_cmd:NNn \bcvector \moremath_column_vector:Vn {b}
% braces
\__moremath_new_vector_shorth_doc_cmd:NNn \Bcvector \moremath_column_vector:Vn {B}
% single vert
\__moremath_new_vector_shorth_doc_cmd:NNn \vcvector \moremath_column_vector:Vn {v}
% double vert
\__moremath_new_vector_shorth_doc_cmd:NNn \Vcvector \moremath_column_vector:Vn {V}
```

Now to the row vectors.

```
% parenthesis
\__moremath_new_vector_shorth_doc_cmd:NNn \prvector \moremath_row_vector:Vn {p}
% brackets
\__moremath_new_vector_shorth_doc_cmd:NNn \brvector \moremath_row_vector:Vn {b}
% braces
\__moremath_new_vector_shorth_doc_cmd:NNn \Brvector \moremath_row_vector:Vn {B}
% single vert
\__moremath_new_vector_shorth_doc_cmd:NNn \vrvector \moremath_row_vector:Vn {v}
% double vert
\__moremath_new_vector_shorth_doc_cmd:NNn \Vrvector \moremath_row_vector:Vn {V}
```

(*End of definition for* `\pcvector` *and others. These functions are documented on page 16.*)

`\psmallcvector`
`\bsmallcvector`
`\Bsmallcvector`
`\vsmallcvector`
`\Vsmallcvector`
`\psmallrvector`
`\bsmallrvector`
`\Bsmallrvector`
`\vsmallrvector`
`\Vsmallrvector`

We also define shorthands for the `\shortcvector` and `\shortrvector` versions.

```
% column vectors
% parenthesis
\__moremath_new_vector_shorth_doc_cmd:NNn \psmallcvector \moremath_column_smallvector:Vn
```

```
  {p}
% brackets
\__moremath_new_vector_shorth_doc_cmd:NNn \bsmallcvector \moremath_column_smallvector:Vn
  {b}
% braces
\__moremath_new_vector_shorth_doc_cmd:NNn \Bsmallcvector \moremath_column_smallvector:Vn
  {B}
% single vert
\__moremath_new_vector_shorth_doc_cmd:NNn \vsmallcvector \moremath_column_smallvector:Vn
  {v}
% double vert
\__moremath_new_vector_shorth_doc_cmd:NNn \Vsmallcvector \moremath_column_smallvector:Vn
  {V}
%
% row vectors
% parenthesis
\__moremath_new_vector_shorth_doc_cmd:NNn \psmallrvector \moremath_row_smallvector:Vn {p}
% brackets
\__moremath_new_vector_shorth_doc_cmd:NNn \bsmallrvector \moremath_row_smallvector:Vn {b}
% braces
\__moremath_new_vector_shorth_doc_cmd:NNn \Bsmallrvector \moremath_row_smallvector:Vn {B}
% single vert
\__moremath_new_vector_shorth_doc_cmd:NNn \vsmallrvector \moremath_row_smallvector:Vn {v}
% double vert
\__moremath_new_vector_shorth_doc_cmd:NNn \Vsmallrvector \moremath_row_smallvector:Vn {V}


}{ % \bool_if:nTF \l__moremath_predef_crvector_bool FALSE PATH
  \msg_info:nnnn {moremath} {load / disabling} {no-crvector}
  {
    commands~producing~row~and~column~vectors
  }
} % \bool_if:nTF \l__moremath_predef_crvector_bool
```

(*End of definition for* `\psmallcvector` *and others. These functions are documented on page 17.*)

### 9.3.2 (Anti-)diagonal Matrices

Now to the (anti-)diagonal matrix shorthands, these are also guarded by a key value option.

```
\bool_if:nTF \l__moremath_predef_matrix_bool
{
```

`\diagmat`
`\antidiagmat`
`\smalldiagmat`
`\smallantidiagmat`

```
\cs_if_free:NTF \diagmat
{
  \NewDocumentCommand \diagmat { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_diagonal_matrix:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  \msg_warning:nnn {moremath} {matrix / already-defined-doc-cmd-skip}
  {
    \diagmat
  }
} % \cs_if_free:nTF \diagmat

\cs_if_free:NTF \antidiagmat
{
  \NewDocumentCommand \antidiagmat { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_antidiagonal_matrix:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
  {
    \antidiagmat
  }
} % \cs_if_free:nTF \antidiagmat

\cs_if_free:NTF \smalldiagmat
{
  \NewDocumentCommand \smalldiagmat { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_diagonal_smallmatrix:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
  {
    \smalldiagmat
  }
}

\cs_if_free:NTF \smallantidiagmat
{
  \NewDocumentCommand \smallantidiagmat { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_antidiagonal_smallmatrix:Vn \l__moremath_matrix_delim_tl {#2}
    \group_end:
  }
}{
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
  {
    \smallantidiagmat
  }
}
```

(*End of definition for* `\diagmat` *and others. These functions are documented on page 18.*)

**(Anti-)diagonal Matrices with Pre-defined Delimiters** As it is sort of cumbersome to always specify the delimiter key, we also provide commands with pre-defined delimiters.

`\__moremath_new_matrix_shorth_doc_cmd:NNn`

To provide several shorthands for delimited matrices, we use a helper function to avoid code duplication. `\__moremath_new_matrix_shorth_doc_cmd:NNn` takes three arguments:

`#1` : The ⟨*csname*⟩ to define
`#2` : The ⟨*csname*⟩ of the matrix function to use, which should have the signature `Vn`.
`#3` : The "predefined" delimiter of this version

```
\cs_new_protected:Nn \__moremath_new_matrix_shorth_doc_cmd:NNn
{
  \cs_if_free:NTF #1
  {
    \NewDocumentCommand #1 { o m }
    {
      \group_begin:
      \tl_if_empty:nF {#3}
      {
        \keys_set:nn { moremath / matrix }
        {
          delimiter = #3
        }
      } % \tl_if_empty:nF {#3}
      \tl_if_novalue:nF {##1}
      {
        \keys_set:nn { moremath / matrix } {##1}
      }
      #2 \l__moremath_matrix_delim_tl {##2}
      \group_end:
    }
  }{
    \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
    {
      #1
    }
  }
}
```

(*End of definition for* `\__moremath_new_matrix_shorth_doc_cmd:NNn`*.*)

We now define the shorthand commands with predefined delimiters.

`\pdiagmat`
`\bdiagmat`
`\Bdiagmat`
`\vdiagmat`
`\Vdiagmat`

We begin with the regular diagonal matrix

```
\__moremath_new_matrix_shorth_doc_cmd:NNn \pdiagmat \moremath_diagonal_matrix:Vn {p}
\__moremath_new_matrix_shorth_doc_cmd:NNn \bdiagmat \moremath_diagonal_matrix:Vn {b}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Bdiagmat \moremath_diagonal_matrix:Vn {B}
\__moremath_new_matrix_shorth_doc_cmd:NNn \vdiagmat \moremath_diagonal_matrix:Vn {v}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Vdiagmat \moremath_diagonal_matrix:Vn {V}
```

(*End of definition for* `\pdiagmat` *and others. These functions are documented on page 19.*)

`\pantidiagmat`
`\bantidiagmat`
`\Bantidiagmat`
`\vantidiagmat`
`\Vantidiagmat`

Now for the anti-diagonal matrix commands.

```
\__moremath_new_matrix_shorth_doc_cmd:NNn \pantidiagmat
  \moremath_antidiagonal_matrix:Vn {p}
\__moremath_new_matrix_shorth_doc_cmd:NNn \bantidiagmat
  \moremath_antidiagonal_matrix:Vn {b}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Bantidiagmat
  \moremath_antidiagonal_matrix:Vn {B}
\__moremath_new_matrix_shorth_doc_cmd:NNn \vantidiagmat
  \moremath_antidiagonal_matrix:Vn {v}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Vantidiagmat
  \moremath_antidiagonal_matrix:Vn {V}
```

(*End of definition for* `\pantidiagmat` *and others. These functions are documented on page 19.*)

`\psmalldiagmat`
`\bsmalldiagmat`
`\Bsmalldiagmat`
`\vsmalldiagmat`
`\Vsmalldiagmat`

We continue with the inline math versions based on the `smallmatrix*` environment.

```
\__moremath_new_matrix_shorth_doc_cmd:NNn \psmalldiagmat
  \moremath_diagonal_smallmatrix:Vn {p}
\__moremath_new_matrix_shorth_doc_cmd:NNn \bsmalldiagmat
  \moremath_diagonal_smallmatrix:Vn {b}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Bsmalldiagmat
  \moremath_diagonal_smallmatrix:Vn {B}
\__moremath_new_matrix_shorth_doc_cmd:NNn \vsmalldiagmat
  \moremath_diagonal_smallmatrix:Vn {v}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Vsmalldiagmat
  \moremath_diagonal_smallmatrix:Vn {V}
```

(*End of definition for* `\psmalldiagmat` *and others. These functions are documented on page 20.*)

`\psmallantidiagmat`
`\bsmallantidiagmat`
`\Bsmallantidiagmat`
`\vsmallantidiagmat`
`\Vsmallantidiagmat`

We provide also anti-diagonal versions of the small matrices.

```
\__moremath_new_matrix_shorth_doc_cmd:NNn \psmallantidiagmat
  \moremath_antidiagonal_smallmatrix:Vn {p}
\__moremath_new_matrix_shorth_doc_cmd:NNn \bsmallantidiagmat
  \moremath_antidiagonal_smallmatrix:Vn {b}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Bsmallantidiagmat
  \moremath_antidiagonal_smallmatrix:Vn {B}
\__moremath_new_matrix_shorth_doc_cmd:NNn \vsmallantidiagmat
  \moremath_antidiagonal_smallmatrix:Vn {v}
\__moremath_new_matrix_shorth_doc_cmd:NNn \Vsmallantidiagmat
  \moremath_antidiagonal_smallmatrix:Vn {V}
```

(*End of definition for* `\psmallantidiagmat` *and others. These functions are documented on page 20.*)

### 9.3.3 Identity Matrices

We also provide document level commands for producing an identity matrix. These commands are also guarded by the same variable as the other matrix commands (`\l__moremath_predef_matrix_bool`).

`\idmat`
`\smallidmat`

We provide two document level commands for producing the identiy matrix, one for inline math mode and one for display math mode.

We start with the display math mode version.

```
\cs_if_free:NTF \idmat
{
  \NewDocumentCommand \idmat { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_id_matrix:n {#2}
    \group_end:
  }
}{ % \cs_if_free:NTF \idmat FALSE BRANCH
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
    {\idmat}
}
```

Afterwards we continue with the inline math mode version.

```
\cs_if_free:NTF \smallidmat
{
  \NewDocumentCommand \smallidmat { o m }
  {
    \group_begin:
    \tl_if_novalue:nF {#1}
    {
      \keys_set:nn { moremath / matrix } {#1}
    }
    \moremath_id_smallmatrix:n {#2}
    \group_end:
  }
}{ % \cs_if_free:NTF \smallidmat FALSE BRANCH
  \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
    {\smallidmat}
}
```

(*End of definition for* `\idmat` *and* `\smallidmat`*. These functions are documented on page 21.*)

**Identity Matrices with Pre-defined Delimiters** We also define shorthands for the commonly used delimiters around matrices, to avoid code duplication, we first declare a helper function for this.

`\__moremath_new_id_matrix_doc_cmd:NNn`

This function creates a new document level command for an identity matrix like command. It allows pre-setting ⟨*kv opts*⟩. The function takes three arguments.

`#1` : The ⟨*csname*⟩ of the document level command to define

`#2` : The ⟨*csname*⟩ of the function to use
This is indented to be one of `\moremath_id_matrix:n` or `\moremath_id_smallmatrix:n`

`#3` : ⟨*kv opts*⟩ to preset in the `moremath / matrix` namespace for this command
This is meant to be used for pre-setting the key `delimiter`.

```
\cs_new_protected:Nn \__moremath_new_id_matrix_doc_cmd:NNn
{
  \cs_if_free:NTF #1
  {
    \NewDocumentCommand #1 { o m }
    {
      \group_begin:
      \tl_if_empty:nF {#3}
      {
        \keys_set:nn { moremath / matrix } {#3}
      }
      \tl_if_novalue:nF {##1}
      {
        \keys_set:nn { moremath / matrix } {##1}
      }
      #2 {##2}
      \group_end:
    }
  }{ % \cs_if_free:NTF #1 FALSE BRANCH
    \msg_warning:nnn { moremath } { matrix / already-defined-doc-cmd-skip }
      {#1}
  }
}
```

(*End of definition for* `\__moremath_new_id_matrix_doc_cmd:NNn`.)

`\pidmat`
`\bidmat`
`\Bidmat`
`\vidmat`
`\Vidmat`

We begin with the display math versions, starting with the version delimited by parenthesis,

```
\__moremath_new_id_matrix_doc_cmd:NNn \pidmat \moremath_id_matrix:n { delimiter = p }
```

continue with the bracketed version,

```
\__moremath_new_id_matrix_doc_cmd:NNn \bidmat \moremath_id_matrix:n { delimiter = b }
```

the version using braces,

```
\__moremath_new_id_matrix_doc_cmd:NNn \Bidmat \moremath_id_matrix:n { delimiter = B }
```

single vertical lines,

```
\__moremath_new_id_matrix_doc_cmd:NNn \vidmat \moremath_id_matrix:n { delimiter = v }
```

and finally double vertical lines.

```
\__moremath_new_id_matrix_doc_cmd:NNn \Vidmat \moremath_id_matrix:n { delimiter = V }
```

(*End of definition for* `\pidmat` *and others. These functions are documented on page 21.*)

`\psmallidmat`
`\bsmallidmat`
`\Bsmallidmat`
`\vsmallidmat`
`\Vsmallidmat`

Now we also define shorthands for inline math mode. We start again defining the version using parenthesis,

```
\__moremath_new_id_matrix_doc_cmd:NNn \psmallidmat \moremath_id_smallmatrix:n
  { delimiter = p }
```

then brackets,

```
\__moremath_new_id_matrix_doc_cmd:NNn \bsmallidmat \moremath_id_smallmatrix:n
  { delimiter = b }
```

then braces,

```
\__moremath_new_id_matrix_doc_cmd:NNn \Bsmallidmat \moremath_id_smallmatrix:n
  { delimiter = B }
```

followed by single vertical lines,

```
\__moremath_new_id_matrix_doc_cmd:NNn \vsmallidmat \moremath_id_smallmatrix:n
  { delimiter = v }
```

and finally double vertical lines.

```
\__moremath_new_id_matrix_doc_cmd:NNn \Vsmallidmat \moremath_id_smallmatrix:n
  { delimiter = V }
```

(*End of definition for* `\psmallidmat` *and others. These functions are documented on page 21.*)

```
}{ % \bool_if:nTF \l__moremath_predef_matrix_bool FALSE BRANCH
  \msg_info:nnnn {moremath} { load / disabling } { no-matrix }
  {
    (anti-)diagonal~matrix~commands
  }
} % \bool_if:nTF \l__moremath_predef_matrix_bool
```

## 10 Shorthand Macros for Absolute Value and Norm

We first declare another warning message to inform the user of the case that, the ⟨*csnames*⟩ are already taken.

```
\msg_new:nnnn { moremath } { abs-shorth / csname-already-defined-skip }
{
  Control~sequence~’#1’~is~already~defined.\\
  Skipping~declaration~of~paired~delimiter~\msg_line_context:.\\
  Use~package~option~’no-abs-shorthands’~to~disable~the~paired\\
  delimiter~shorthands.
}{
  The~control~sequence~’#1’~has~already~been\\
  defined~by~something~else.\\
  I~am~refusing~to~overwrite~its~existing~definition~and~instead~avoid\\
  declaring~a~paired~delimiter.\\
}
```

As with the other parts these macros may be conditionally disabled.

```
\bool_if:NTF \l__moremath_predef_abs_bool
{
```

`\abs`
`\norm`

These macros provide shorthands for $|\langle \mathit{content} \rangle|$ and $\|\langle \mathit{content} \rangle\|$.

```
\cs_if_free:NTF \abs
{
  \DeclarePairedDelimiter \abs {\lvert} {\rvert}
}{
  % warn if the csname is taken
  \msg_warning:nnn { moremath } { abs-shorth / csname-already-defined-skip }
    {\abs}
} % \cs_if_free:NTF \abs
```

```

\cs_if_free:NTF \norm
{
  \DeclarePairedDelimiter \norm {\lVert} {\rVert}
}{
  % warn if csname is already taken
  \msg_warning:nnn { moremath } { abs-shorth / csname-already-defined-skip }
    {\norm}
} % \cs_if_free:NTF
```

(*End of definition for* `\abs` *and* `\norm`*. These functions are documented on page 22.*)

```
}{
  \msg_info:nnnn {moremath} {load / disabling} {no-abs-shorthands}
  {
    '\abs'~and~'\norm'~macros
  }
} % End of the conditional
⟨/package⟩
```

# Copyright and License

The following copyright notice applies to the moremath package:

>

The file `MANIFEST.md` has to be distributed together with the package.

# References

[CMT23] David Carlisle, Frank Mittelbach, and The LaTeX Project Team. *The bm package. Access bold symbols in maths mode.* Version 1.2f. Dec. 19, 2023. URL: https://ctan.org/pkg/bm (visited on 07/04/2024).

[Høg+24] Morten Høgholm et al. *The mathtools package. Mathematical tools to use with amsmath.* Version 1.30. Mar. 11, 2024. URL: https://ctan.org/pkg/mathtools (visited on 07/04/2024).

[Mit18] Frank Mitelbach. “A rollback concept for packages and classes.” In: *TUGboat* 2 (2018). URL: https://www.latex-project.org/publications/2018-FMi-TUB-tb122mitt-version-rollback.pdf (visited on 08/06/2024).

[The] The American Mathematical Society. *The amsfonts package. TeX fonts from the American Mathematical Society.* Version 3.04. URL: https://ctan.org/pkg/amsfonts (visited on 07/04/2024).

[The23] The LaTeX Project Team. *The amsmath package. AMS mathematical facilities for LaTeX.* Version 2.17o. May 13, 2023. URL: https://ctan.org/pkg/amsmath (visited on 07/04/2024).

# Change History

v0.1.0 – 2024-06-28

- General: Initial development release . . 1
- `align`: Added options for the matrix-based environments of mathtools. . . . . 31
- `dalembert-symb`: Added options controlling appearance. . . . . 30
- `\DeclareDelimitedOperator`: Added document level command. . . . . 38
- `\laplacian`: Added `\grad`, `\divergence`, `\curl` and `\laplacian` commands. . . . . 51
- `\moremath_antidiagonal_smallmatrix:VV`: Added variants. . . . . 56
- `\moremath_antidiagonal_smallmatrix:nn`: Added functions producing (anti-)diagonal matrices. Also added versions for inline math. . . 56
- `\moremath_curl_operator:n`: Added functions. . . . . 46
- `\moremath_delim_nabla_op_autoscale:NNnn`: Added `noscale` and `autoscale` functions. . . . . 47
- `\moremath_delim_nabla_op_manuscale:NNVnn`: Add variant. . . . . 47
- `\moremath_delim_nabla_op_manuscale:NNnnn`: Add function. . . . . 47
- `\moremath_delim_op_autoscale:NNnnn`: Added `noscale` and `autoscale` versions. . . . . 36
- `\moremath_delim_op_manuscale:NNVnnn`: Function added. . . . . 37
- `\moremath_laplace_operator:n`: Added functions. . . . . 45
- `\l__moremath_matrix_align_tl`: Added internal variables. . . . . 31
- `\moremath_new_delim_op_command:cNN`: Added function and variant. . . . . 38
- `\moremath_row_smallvector:Vn`: Added functions producing inline math row- and column vectors. . . 54
- `\moremath_row_vector:Vn`: Added functions producing row- and column vectors. . . . . 53
- `\moremath_setup:n`: Added function for setting options. . . . . 31
- `\moremathsetup`: Added document command for setting package options. . . . . 31
- `nopredef`: Added package load-time options. . . . . 29
- `\norm`: Added shorthands for absolute value and norm. . . . . 68
- `\smallantidiagmat`: Added commands producing (anti-)diagonal matrices, including inline math versions. . . . 63
- `\smallrvector`: Added commands producing row and column vectors, including inline math versions. . . . 59
- `\Vantidiagmat`: Added commands for delimited antidiagonal matrices. . . 64
- `\Varccos`: Added delimited document commands for all amsmath-defined operators. . . . . 39
- `\Vcurl`: Added commands for delimited curl operators. . . . . 52
- `\Vdiagmat`: Added commands for delimited diagonal matrices. . . . . 64
- `\Vdiv`: Added commands for delimited divergence operators. . . . . 52
- `\Vgrad`: Added commands for delimited gradient operators. . . . . 52
- `\Vlaplacian`: Added commands for delimited Laplace operators. . . . . 52
- `\Vrvector`: Added commands for row and column vectors with predefined delimiters. . . . . 60
- `\Vsmallantidiagmat`: Added commands for delimited

anti-diagonal matrices suitable for inline math. 64
- `\Vsmalldiagmat`: Added commands for delimited diagonal matrices suitable for inline math. 64
- `\Vsmallrvector`: Added commands for inline math row and column vectors with predefined delimiters. 61

v0.2.0 – 2024-07-04
- General: Fixed example to match description. 13
  - Fixed syntax description of standalone operators not matching reality. 15
  - New: Load `amssymb` if the `no-vector` option has *not* been given. 32
- `dalembert-symb`: Added: New option `dalembert-symb`. 30
- `\moremath_antidiagonal_smallmatrix:VV`: Added `nV` and `VV` variants. 56
- `\moremath_dalembert_operator:n`: Added new function `\moremath_dalembert_operator:n`. 46
- `\__moremath_dalembert_operator_get::`: Added function: `\__moremath_dalembert_operator_get:` 45
- `\l__moremath_dalembert_symb_tl`: Added: New variable `\l__moremath_dalembert_symb_tl` 29
  - Explicitly declared variables before using them to set keys. 29
- `\__moremath_generate_one_filled_clist:Nn`: Added function. 56
- `\l__moremath_id_entries_clist`: Added variable. 57
- `\moremath_id_smallmatrix:n`: Added functions `\moremath_id_matrix:n` and `\moremath_id_smallmatrix:n` and variants. 57
- `\l__moremath_predef_abs_bool`: Explicitly declared boolean variables used to store key-value options. 27
- `\quabla`: Added `\quabla` command. 51
- `\smallidmat`: Added `\idmat` and `\smallidmat` document level commands. 65
- `\Vidmat`: Added document level commands `\pidmat`, `\bidmat`, `\Bidmat`, `\vidmat`, and `\Vidmat`. 66
- `\Vquabla`: Added commands for a delimited d'Alembert operator. 52
- `\Vsmallidmat`: Added document level commands `\psmallidmat`, `\bsmallidmat`, `\Bsmallidmat`, `\vsmallidmat` and `\Vsmallidmat`. 67

v0.3.0 – 2024-07-08
- General: Split the documentation into several parts. 1
- `\laplacian`: Changed: Spaces between the ⟨*csname*⟩ and the optional argument are now disallowed for `\grad`, `\divergence`, `\curl` and `\laplacian`. 51
- `\moremath_vcenter:n`: Added new function. 34
- `\quabla`: Changed: Spaces between the ⟨*csname*⟩ and the optional argument are now disallowed. 51
- `vcenter`: Added new option. 30
- `\VCenterMath`: Added new document level command. 35

v0.4.0 – 2024-07-15
- General: **Breaking Change:** Rename the package and the ⟨*prefix*⟩ used for function names from `conmath` to `moremath`. 1

v0.5.0 – 2024-08-20
- General: Add information for LaTeX's package rollback mechanism. 27
- `\__moremath_dalembert_operator_get::`: Now uses `\__moremath_tex_mathop:n` instead of TeX-primitives. 45
- `\__moremath_gradient_operator_get::`: Now uses `\__moremath_tex_mathop:n` instead of TeX-primitives. 44
- `\__moremath_laplace_operator_get::`: Now uses `\__moremath_tex_mathop:n` instead of TeX-primitives. 45
- `\__moremath_matrix_star_end:V`: Added new functions and variants. 33
- `\__moremath_new_vector_shorth_doc_cmd:NNn`: Require function with `Vn` signature to pass variable correctly. 60
- `\moremath_row_smallvector:Vn`: Added new `Vn` variants. 54
- `\moremath_row_vector:Vn`: Added new `Vn` variants. 53
- `\__moremath_tex_mathop:n`: Added new function. 33
- `\__moremath_tex_mathpalette:Nn`: Added new function. 32

__moremath_tex_mathsurround:n: Added new function. . . . . . . . . . . 33
\moremath_vcenter:n: Changed: Make use of __moremath_tex_mathpalette:Nn. . . . . . . . . . . . . 34
__moremath_vcenter:Nn: Changed: Now uses __moremath_tex_vcenter:n and __moremath_tex_mathsurround:n instead of TEX-primitives. . . . . . 34
\smallrvector: Use Vn versions of functions when passing variables around. . . . . . . . . . . . . . . . . . . 59
\Vrvector: Use functions with Vn signature instead of nn . . . . . . . . 60
\Vsmallrvector: Use functions with Vn signature instead of nn . . . . . . 61

# 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.