基于 l3fparray 的轻量矩阵

Lightweight Matrix Based on l3fparray

Ms_yam*

2026-11-16

1 用户文档 [English]

1.1 宏包简介

matrix 提供一种轻量矩阵类型，主要用于 TeX 文档中的矩阵演示、数值示例和几何变换。使用者可以创建矩阵、设置矩阵元素、执行矩阵运算、遍历元素，并将矩阵排版到数学公式中。

本宏包中的矩阵具有确定的行数和列数。通常，矩阵创建后其维度不应修改，且本宏包不提供直接修改矩阵维度的接口。但当矩阵作为运算结果时，其维度会根据运算结果自动更新。矩阵元素则可以通过赋值命令修改。矩阵的创建和修改均为全局操作：在 TeX 分组内执行修改命令，分组结束后矩阵内容不会自动恢复。

空矩阵也具有明确的行数和列数，但不包含任何元素。行数和列数不能为负数。

1.2 快速开始

下面的例子创建一个 2×3 矩阵，然后设置元素，并在数学模式中排版：

```
\ExplSyntaxOn
\matrix_new:Nnn \l_demo_matrix { 2 } { 3 }
\matrix_gset:Nn \l_demo_matrix { 1, 2, 3, 4, 5, 6 }
\[
  \matrix_use:N \l_demo_matrix
\]
\ExplSyntaxOff
```

*E-mail: Ms_yam@163.com

其效果为：

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

1.3 命令与参数约定

本宏包主要作为 expl3 的基础扩展宏包，其主要接口遵循 expl3 的命名规范，使用时应处于 `\ExplSyntaxOn` 状态。此外，为方便不熟悉 expl3 的 LaTeX 用户使用，本宏包也为常用操作提供了相应的 LaTeX2e 风格命令。

expl3 风格命令名称中，冒号后的字母表示参数的形式：`N` 接收单个控制序列，`n` 接收按原样传入的一般参数（通常写在一对花括号中），`c` 则根据给出的字符名称构造控制序列。因此，`N` 形式直接接收矩阵变量，例如 `\matrix_new:Nnn`；`c` 形式接收矩阵变量的名称（不带反斜杠），例如 `\matrix_new:cnn {l_demo_matrix} {2} {2}`。同一组命令的不同变体执行相同操作，区别仅在于相应参数的传递形式。

在本文档中，⟨*matrix*⟩ 表示矩阵变量；⟨*destination*⟩ 表示接收结果的目标矩阵；⟨*source*⟩ 表示作为输入的源矩阵。除创建命令外，作为参数传入的矩阵都应当已经创建。

1.4 创建矩阵

`\matrix_new:Nnn` `\matrix_new:cnn`

`\matrix_new:Nnn` ⟨*matrix*⟩ {⟨*rows*⟩} {⟨*columns*⟩}

创建一个普通矩阵，所有元素均初始化为零。

⟨*matrix*⟩ 是新矩阵的变量；⟨*rows*⟩ 和 ⟨*columns*⟩ 分别指定矩阵的行数和列数。两个参数均按整数表达式求值，且必须为非负数。维度允许为零。

如果矩阵变量已经定义，则产生错误。

`\matrix_const:Nnnn` `\matrix_const:cnnn`

`\matrix_const:Nnnn` ⟨*matrix*⟩ {⟨*rows*⟩} {⟨*columns*⟩} {⟨*comma list*⟩}

创建一个常量矩阵，并使用逗号分隔列表初始化其元素。元素按照行优先顺序填入；列表中未指定的元素初始化为零。常量矩阵不能通过本宏包提供的矩阵修改命令进行修改。

⟨*matrix*⟩、⟨*rows*⟩ 和 ⟨*columns*⟩ 的含义与 `\matrix_new:Nnn` 相同。⟨*comma list*⟩ 是矩阵元素列表，每一项均按浮点数表达式求值。列表项数不得超过矩阵的元素总数；如果超过，则产生错误，且矩阵不会被创建。

1.5 设置和修改矩阵

以下命令用于修改已创建矩阵的内容。目标矩阵必须是普通矩阵；如果目标矩阵是常量矩阵，则会产生错误。所有修改操作均为全局操作。

\matrix_gzero:N
\matrix_gzero:c

\matrix_gzero:N ⟨*matrix*⟩

将 ⟨*matrix*⟩ 的所有元素设为零，同时保持矩阵的行数和列数不变。如果矩阵为空，则其维度信息保持不变。

\matrix_gset:Nn
\matrix_gset:cn

\matrix_gset:Nn ⟨*matrix*⟩ {⟨*comma list*⟩}

使用逗号分隔列表重新设置 ⟨*matrix*⟩ 的所有元素，并按照行优先顺序写入。列表中的每一项均按浮点数表达式求值。列表项数不得超过矩阵的元素总数；如果列表项数不足，则剩余元素设为零。如果列表项数过多，则产生错误，且本次设置不会执行。矩阵的行数和列数保持不变。

\matrix_gset_eq:NN
\matrix_gset_eq:(cN|Nc)

\matrix_gset_eq:NN ⟨*destination*⟩ ⟨*source*⟩

将 ⟨*source*⟩ 的维度和元素复制到 ⟨*destination*⟩。目标矩阵的维度会相应更新。复制完成后，两个矩阵的数据相互独立；后续修改其中一个矩阵不会影响另一个矩阵。

⟨*destination*⟩ 必须是已创建的普通矩阵；⟨*source*⟩ 必须是已创建的矩阵，可以是普通矩阵或常量矩阵。源矩阵不会被修改。

1.6 逐元素运算

本节介绍对矩阵中的每个元素分别执行相同运算的命令。根据结果的存储方式，命令分为两种形式：一种直接将结果写回原矩阵，另一种将结果写入另一个目标矩阵。无论采用哪种形式，用于存储运算结果的矩阵都必须是已创建的普通矩阵。

直接写回原矩阵的形式保持矩阵的维度不变；写入目标矩阵的形式则会将目标矩阵的维度更新为与源矩阵相同。当目标矩阵与源矩阵是同一个矩阵时，运算结果仍会写回该矩阵，但相比直接写回原矩阵的形式，需要额外的临时矩阵空间分配和回收操作。因此，这种情况下建议使用直接写回原矩阵的形式。

\matrix_gnegate:N
\matrix_gabs:N
\matrix_gsign:N

\matrix_gnegate:N ⟨*matrix*⟩
\matrix_gabs:N ⟨*matrix*⟩
\matrix_gsign:N ⟨*matrix*⟩

分别将 ⟨*matrix*⟩ 的每个元素替换为其相反数、绝对值和符号值，并将运算结果写回其自身。

\matrix_gnegate:NN
\matrix_gabs:NN
\matrix_gsign:NN

\matrix_gnegate:NN ⟨*destination*⟩ ⟨*source*⟩
\matrix_gabs:NN ⟨*destination*⟩ ⟨*source*⟩
\matrix_gsign:NN ⟨*destination*⟩ ⟨*source*⟩

对 ⟨*source*⟩ 的每个元素执行相应运算，并将结果写入 ⟨*destination*⟩。

\matrix_ground:Nn `\matrix_ground:Nn ⟨matrix⟩ {⟨digits⟩}`

对 ⟨*matrix*⟩ 的每个元素执行舍入，并将结果写回其自身。⟨*digits*⟩ 是传递给 l3fp 的 round 运算的精度参数。

\matrix_ground:NNn `\matrix_ground:NNn ⟨destination⟩ ⟨source⟩ {⟨digits⟩}`

对 ⟨*source*⟩ 的每个元素执行舍入，并将结果写入 ⟨*destination*⟩。⟨*digits*⟩ 的含义与 \matrix_ground:Nn 中相同。

\matrix_gadd:Nn `\matrix_gadd:Nn ⟨matrix⟩ {⟨scalar⟩}`
\matrix_gsub:Nn `\matrix_gsub:Nn ⟨matrix⟩ {⟨scalar⟩}`
\matrix_gmul:Nn `\matrix_gmul:Nn ⟨matrix⟩ {⟨scalar⟩}`
\matrix_gdiv:Nn `\matrix_gdiv:Nn ⟨matrix⟩ {⟨scalar⟩}`
\matrix_gpower:Nn `\matrix_gpower:Nn ⟨matrix⟩ {⟨scalar⟩}`

将矩阵 ⟨*matrix*⟩ 的每个元素分别与 ⟨*scalar*⟩ 执行加、减、乘、除或乘方运算，并将结果写回其自身。对于矩阵 A 和标量 c，对应的逐元素运算依次为 $A+c$、$A-c$、$A\times c$、A/c 和 A^c。⟨*scalar*⟩ 按 l3fp 浮点表达式求值。

\matrix_gadd:NNn `\matrix_gadd:NNn ⟨destination⟩ ⟨source⟩ {⟨scalar⟩}`
\matrix_gsub:NNn `\matrix_gsub:NNn ⟨destination⟩ ⟨source⟩ {⟨scalar⟩}`
\matrix_gmul:NNn `\matrix_gmul:NNn ⟨destination⟩ ⟨source⟩ {⟨scalar⟩}`
\matrix_gdiv:NNn `\matrix_gdiv:NNn ⟨destination⟩ ⟨source⟩ {⟨scalar⟩}`
\matrix_gpower:NNn `\matrix_gpower:NNn ⟨destination⟩ ⟨source⟩ {⟨scalar⟩}`

将 ⟨*source*⟩ 的每个元素分别与 ⟨*scalar*⟩ 执行加、减、乘、除或乘方运算，并将结果写入 ⟨*destination*⟩。对于源矩阵 A 和标量 c，对应的逐元素运算依次为 $A+c$、$A-c$、$A\times c$、A/c 和 A^c。⟨*scalar*⟩ 按 l3fp 浮点表达式求值。

1.7 矩阵间的运算

本节介绍两个矩阵之间的加法、减法和乘法运算。根据结果的存储方式，命令分为两种形式：一种将结果写入目标矩阵，另一种直接写回操作数。用于存储运算结果的矩阵必须是已创建的普通矩阵；作为输入的操作数必须是已创建的矩阵，可以是普通矩阵或常量矩阵。

\matrix_gadd:NNN `\matrix_gadd:NNN ⟨destination⟩ ⟨matrix A⟩ ⟨matrix B⟩`
\matrix_gsub:NNN `\matrix_gsub:NNN ⟨destination⟩ ⟨matrix A⟩ ⟨matrix B⟩`

分别计算两个矩阵对应元素的和或差，并将结果写入 ⟨*destination*⟩。两个操作数的行数和列数必须分别相等，否则产生维度不匹配错误。目标矩阵的维度更新为与操作数相同。

\matrix_gadd:NN \matrix_gadd:NN ⟨*matrix A*⟩ ⟨*matrix B*⟩
\matrix_gsub:NN \matrix_gsub:NN ⟨*matrix A*⟩ ⟨*matrix B*⟩

分别执行原地运算 $A \leftarrow A + B$ 和 $A \leftarrow A - B$。两个矩阵的行数和列数必须分别相等，否则产生维度不匹配错误。运算结果写回第一个矩阵，其维度保持不变；第二个矩阵仅作为输入，不会被修改。

\matrix_gmul:NNN \matrix_multiply:NNN ⟨*destination*⟩ ⟨*matrix A*⟩ ⟨*matrix B*⟩

计算两个矩阵的乘积 AB。⟨*matrix A*⟩ 的列数必须等于 ⟨*matrix B*⟩ 的行数，否则产生维度不匹配错误。将结果写入 ⟨*destination*⟩，其维度更新为 ⟨*matrix A*⟩ 的行数乘以 ⟨*matrix B*⟩ 的列数。

\matrix_gmul:NN \matrix_multiply:NN ⟨*matrix A*⟩ ⟨*matrix B*⟩

计算两个矩阵的乘积 AB。⟨*matrix A*⟩ 的列数必须等于 ⟨*matrix B*⟩ 的行数，否则产生维度不匹配错误。将结果写回第一个矩阵，即执行 $A \leftarrow AB$；第一个矩阵的维度更新为乘积结果的维度。

1.8 基于矩阵的映射

映射按行优先顺序访问矩阵中的元素，并在每次访问时将当前元素值传递给回调。映射机制只负责读取和传递元素值，不会自动将回调结果写回矩阵。如果矩阵为空，则不会调用回调。

\matrix_map_function:NN \matrix_map_function:NN ⟨*matrix*⟩ ⟨*function*⟩

对 ⟨*matrix*⟩ 的每个元素调用一次 ⟨*function*⟩。⟨*function*⟩ 必须接收一个参数，该参数为当前元素值。

\matrix_map_inline:Nn \matrix_map_inline:Nn ⟨*matrix*⟩ {⟨*inline code*⟩}

对 ⟨*matrix*⟩ 的每个元素执行一次 ⟨*inline code*⟩。在内联代码中，`##1` 表示当前元素值。由于代码作为映射命令的参数传入，参数标记需要写成两个 `#`。

例如，下面的代码依次将矩阵元素写入终端：

```
\matrix_map_inline:Nn \l_demo_matrix
  { \iow_term:n { ##1 } }
```

\matrix_map_break: \matrix_map_break:
\matrix_map_break:n \matrix_map_break:n {⟨*code*⟩}

在映射回调中调用 `\matrix_map_break:` 可以提前结束当前映射。`\matrix_map_break:n` 除了结束映射，还会在退出映射时执行 ⟨*code*⟩。

1.9 显示和检查矩阵

\matrix_use:N \matrix_use:N ⟨*matrix*⟩
\matrix_use:c 在数学模式中排版 ⟨*matrix*⟩。非空矩阵以圆括号矩阵的形式显示；空矩阵则显示其维度信息，例如 2×0。本命令不会自动进入数学模式，因此必须在数学模式中调用。

\matrix_log:N \matrix_log:N ⟨*matrix*⟩
\matrix_log:c 将 ⟨*matrix*⟩ 的维度、存储信息和元素内容写入编译日志，用于检查矩阵状态和调试。本命令不会在页面正文中排版矩阵。

1.10 预定义矩阵和临时变量

\c_empty_matrix 0×0 空矩阵。

\c_identity_two_matrix
\c_identity_three_matrix
\c_identity_four_matrix 分别为 2×2、3×3 和 4×4 单位矩阵。

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}、\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}、\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

\g_tmpa_matrix
\g_tmpb_matrix 用于全局赋值的临时矩阵变量。本宏包的内部代码不会使用这些变量，因此可以安全地与本宏包提供的命令一起使用。但是，它们可能会被其他代码覆盖，因此只能用于短期存储。

1.11 LaTeX2e 命令

本宏包计划提供部分 LaTeX2e 风格的命令接口。由于目前尚未完成相关命令的设计与实现，本节暂为空。

1.12 错误条件

当前实现会在以下情况下报告错误：

- 创建矩阵时，行数或列数为负数；
- 创建矩阵时，矩阵变量已经定义；

- 将常量矩阵用作修改操作的目标矩阵；
- 创建或设置矩阵时，元素列表中的项数超过矩阵的元素总数；
- 执行矩阵间的逐元素二元运算时，两个操作数的维度不一致；
- 执行矩阵乘法时，左操作数的列数与右操作数的行数不一致。

2 User Guide [中文]

2.1 Package Overview

The matrix package provides a lightweight matrix type, primarily intended for matrix demonstrations, numerical examples, and geometric transformations in TeX documents. It allows users to create matrices, set matrix elements, perform matrix operations, iterate over elements, and typeset matrices in mathematical expressions.

Matrices provided by this package have a defined number of rows and columns. In general, their dimensions are not intended to be changed after creation, and the package provides no direct interface for modifying them. However, when a matrix is used to store an operation result, its dimensions are updated automatically to match the result. Matrix elements can be modified using assignment commands. Matrix creation and modification operations are global: changes made within a TeX group are not automatically reverted when the group ends.

Empty matrices also have well-defined numbers of rows and columns, but contain no elements. The number of rows and columns cannot be negative.

2.2 Quick Start

The following example creates a 2×3 matrix, assigns values to its elements, and typesets the matrix in math mode:

```
\ExplSyntaxOn
\matrix_new:Nnn \l_demo_matrix { 2 } { 3 }
\matrix_gset:Nn \l_demo_matrix { 1, 2, 3, 4, 5, 6 }
\[
  \matrix_use:N \l_demo_matrix
\]
\ExplSyntaxOff
```

The output is:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

2.3 Command and Argument Conventions

This package is primarily intended as a basic extension to expl3, and its main interfaces follow the expl3 naming conventions. These commands should be used with

\ExplSyntaxOn enabled. In addition, the package provides corresponding LaTeX2e-style commands for commonly used operations, making them accessible to LaTeXusers who are not familiar with expl3.

In expl3-style command names, the letters following the colon indicate the forms of the arguments: N accepts a single control sequence, n accepts a general argument (usually enclosed in a pair of braces), and c converts a character-based name into a control sequence for use. Thus, the N form takes a matrix variable directly, as in \matrix_new:Nnn, whereas the c form takes the name of a matrix variable (without the leading backslash), as in \matrix_new:cnn {l_demo_matrix} {2} {2}. Different variants of the same command perform the same operation; they differ only in how the corresponding arguments are passed.

In this document, ⟨*matrix*⟩ denotes a matrix variable; ⟨*destination*⟩ denotes the target matrix that receives a result; and ⟨*source*⟩ denotes an input matrix. Except for creation commands, any matrix passed as an argument must have been created beforehand.

2.4 Creating Matrices

\matrix_new:Nnn
\matrix_new:cnn

\matrix_new:Nnn ⟨*matrix*⟩ {⟨*rows*⟩} {⟨*columns*⟩}

Creates a regular matrix with all elements initially set to zero.

⟨*matrix*⟩ is the variable for the new matrix. ⟨*rows*⟩ and ⟨*columns*⟩ specify the number of rows and columns, respectively. Both arguments are evaluated as integer expressions and must be non-negative. Zero dimensions are allowed.

An error is raised if the matrix variable has already been defined.

\matrix_const:Nnnn
\matrix_const:cnnn

\matrix_const:Nnnn ⟨*matrix*⟩ {⟨*rows*⟩} {⟨*columns*⟩} {⟨*comma list*⟩}

Creates a constant matrix and initializes its elements from a comma-separated list. The elements are filled in row-major order. Any elements not specified in the list are initialized to zero. A constant matrix cannot be modified using the matrix modification commands provided by this package.

⟨*matrix*⟩, ⟨*rows*⟩, and ⟨*columns*⟩ have the same meanings as for \matrix_new:Nnn. ⟨*comma list*⟩ is the list of matrix elements, with each item evaluated as a floating-point expression. The number of items must not exceed the total number of elements in the matrix. If it does, an error is raised and the matrix is not created.

2.5 Setting and Modifying Matrices

The following functions modify the contents of an existing matrix. The destination matrix must be a regular matrix; an error is raised if it is a constant matrix. All modifications are global.

`\matrix_gzero:N` `\matrix_gzero:c`

`\matrix_gzero:N` ⟨*matrix*⟩

Sets all elements of ⟨*matrix*⟩ to zero, leaving its number of rows and columns unchanged. If the matrix is empty, its dimensions remain unchanged.

`\matrix_gset:Nn` `\matrix_gset:cn`

`\matrix_gset:Nn` ⟨*matrix*⟩ {⟨*comma list*⟩}

Sets all elements of ⟨*matrix*⟩ using a comma-separated list, with the elements filled in row-major order. Each item in the list is evaluated as a floating-point expression. The number of items must not exceed the total number of elements in the matrix. If fewer items are provided, the remaining elements are set to zero. If too many items are provided, an error is raised and the assignment is not performed. The dimensions of the matrix remain unchanged.

`\matrix_gset_eq:NN` `\matrix_gset_eq:(cN|Nc)`

`\matrix_gset_eq:NN` ⟨*destination*⟩ ⟨*source*⟩

Copies the dimensions and elements of ⟨*source*⟩ to ⟨*destination*⟩. The dimensions of the destination matrix are updated accordingly. After the copy, the two matrices have independent data; subsequent modifications to either matrix do not affect the other.

⟨*destination*⟩ must be an existing regular matrix. ⟨*source*⟩ must be an existing matrix and may be either a regular matrix or a constant matrix. The source matrix is not modified.

2.6 Element-wise Operations

This section describes commands that perform the same operation on each element of a matrix. Depending on how the results are stored, the commands are provided in two forms: one writes the results back to the original matrix, while the other writes them to a separate destination matrix. In either form, the matrix used to store the results must be an existing regular (non-constant) matrix.

The form that writes the results back to the original matrix preserves its dimensions, whereas the form that writes the results to a destination matrix updates the destination dimensions to match those of the source matrix. If the destination and

source are the same matrix, the results are still written back to that matrix. However, compared with the form that writes directly to the original matrix, this requires additional allocation and deallocation of temporary matrix storage. Therefore, the former form is recommended in this case.

`\matrix_gnegate:N` `\matrix_gnegate:N` ⟨*matrix*⟩
`\matrix_gabs:N` `\matrix_gabs:N` ⟨*matrix*⟩
`\matrix_gsign:N` `\matrix_gsign:N` ⟨*matrix*⟩

Replaces each element of ⟨*matrix*⟩ with its negation, absolute value, or sign, respectively, and writes the results back to the same matrix.

`\matrix_gnegate:NN` `\matrix_gnegate:NN` ⟨*destination*⟩ ⟨*source*⟩
`\matrix_gabs:NN` `\matrix_gabs:NN` ⟨*destination*⟩ ⟨*source*⟩
`\matrix_gsign:NN` `\matrix_gsign:NN` ⟨*destination*⟩ ⟨*source*⟩

Applies the corresponding operation to each element of ⟨*source*⟩ and writes the results to ⟨*destination*⟩.

`\matrix_ground:Nn` `\matrix_ground:Nn` ⟨*matrix*⟩ {⟨*digits*⟩}

Rounds each element of ⟨*matrix*⟩ and writes the results back to the same matrix. ⟨*digits*⟩ is the precision argument passed to the `round` operation of l3fp.

`\matrix_ground:NNn` `\matrix_ground:NNn` ⟨*destination*⟩ ⟨*source*⟩ {⟨*digits*⟩}

Rounds each element of ⟨*source*⟩ and writes the results to ⟨*destination*⟩. The meaning of ⟨*digits*⟩ is the same as for `\matrix_ground:Nn`.

`\matrix_gadd:Nn` `\matrix_gadd:Nn` ⟨*matrix*⟩ {⟨*scalar*⟩}
`\matrix_gsub:Nn` `\matrix_gsub:Nn` ⟨*matrix*⟩ {⟨*scalar*⟩}
`\matrix_gmul:Nn` `\matrix_gmul:Nn` ⟨*matrix*⟩ {⟨*scalar*⟩}
`\matrix_gdiv:Nn` `\matrix_gdiv:Nn` ⟨*matrix*⟩ {⟨*scalar*⟩}
`\matrix_gpower:Nn` `\matrix_gpower:Nn` ⟨*matrix*⟩ {⟨*scalar*⟩}

Performs addition, subtraction, multiplication, division, or exponentiation, respectively, between each element of ⟨*matrix*⟩ and ⟨*scalar*⟩, and writes the results back to the same matrix. For a matrix A and a scalar c, the corresponding element-wise operations are $A + c$, $A - c$, $A \times c$, A/c, and A^c, respectively. ⟨*scalar*⟩ is evaluated as a floating-point expression by l3fp.

\matrix_gadd:NNn \matrix_gadd:NNn ⟨*destination*⟩ ⟨*source*⟩ {⟨*scalar*⟩}
\matrix_gsub:NNn \matrix_gsub:NNn ⟨*destination*⟩ ⟨*source*⟩ {⟨*scalar*⟩}
\matrix_gmul:NNn \matrix_gmul:NNn ⟨*destination*⟩ ⟨*source*⟩ {⟨*scalar*⟩}
\matrix_gdiv:NNn \matrix_gdiv:NNn ⟨*destination*⟩ ⟨*source*⟩ {⟨*scalar*⟩}
\matrix_gpower:NNn \matrix_gpower:NNn ⟨*destination*⟩ ⟨*source*⟩ {⟨*scalar*⟩}

Performs addition, subtraction, multiplication, division, or exponentiation, respectively, between each element of ⟨*source*⟩ and ⟨*scalar*⟩, and writes the results to ⟨*destination*⟩. For a source matrix A and a scalar c, the corresponding element-wise operations are $A + c$, $A - c$, $A \times c$, A/c, and A^c, respectively. ⟨*scalar*⟩ is evaluated as a floating-point expression by l3fp.

2.7 Operations Between Matrices

This section describes addition, subtraction, and multiplication between two matrices. Depending on how the results are stored, the commands are provided in two forms: one writes the results to a destination matrix, while the other writes them back to an operand. The matrix used to store the results must be an existing regular matrix; input operands must be existing matrices and may be either regular or constant matrices.

\matrix_gadd:NNN \matrix_gadd:NNN ⟨*destination*⟩ ⟨*matrix A*⟩ ⟨*matrix B*⟩
\matrix_gsub:NNN \matrix_gsub:NNN ⟨*destination*⟩ ⟨*matrix A*⟩ ⟨*matrix B*⟩

Computes the element-wise sum or difference of two matrices, respectively, and writes the result to ⟨*destination*⟩. The two operands must have the same number of rows and the same number of columns; otherwise, a dimension mismatch error is raised. The dimensions of the destination matrix are updated to match those of the operands.

\matrix_gadd:NN \matrix_gadd:NN ⟨*matrix A*⟩ ⟨*matrix B*⟩
\matrix_gsub:NN \matrix_gsub:NN ⟨*matrix A*⟩ ⟨*matrix B*⟩

Performs the in-place operation $A \leftarrow A + B$ or $A \leftarrow A - B$, respectively. The two matrices must have the same number of rows and the same number of columns; otherwise, a dimension mismatch error is raised. The result is written back to the first matrix, whose dimensions remain unchanged. The second matrix is used only as input and is not modified.

\matrix_gmul:NNN \matrix_multiply:NNN ⟨*destination*⟩ ⟨*matrix A*⟩ ⟨*matrix B*⟩

Computes the matrix product AB. The number of columns in ⟨*matrix A*⟩ must equal the number of rows in ⟨*matrix B*⟩; otherwise, a dimension mismatch error is raised. The result is written to ⟨*destination*⟩, whose dimensions are updated to the number of rows in ⟨*matrix A*⟩ by the number of columns in ⟨*matrix B*⟩.

\matrix_gmul:NN \matrix_multiply:NN ⟨*matrix A*⟩ ⟨*matrix B*⟩

Computes the matrix product AB. The number of columns in ⟨*matrix A*⟩ must equal the number of rows in ⟨*matrix B*⟩; otherwise, a dimension mismatch error is raised. The result is written back to the first matrix, performing the operation $A \leftarrow AB$. The dimensions of the first matrix are updated to match those of the product.

2.8 Mapping over Matrices

Mapping visits the elements of a matrix in row-major order and passes the value of the current element to the callback on each iteration. The mapping mechanism only reads and passes element values; it does not automatically write callback results back to the matrix. If the matrix is empty, the callback is not invoked.

\matrix_map_function:NN \matrix_map_function:NN ⟨*matrix*⟩ ⟨*function*⟩

Calls ⟨*function*⟩ once for each element of ⟨*matrix*⟩. ⟨*function*⟩ must accept one argument, which is the value of the current element.

\matrix_map_inline:Nn \matrix_map_inline:Nn ⟨*matrix*⟩ {⟨*inline code*⟩}

Executes ⟨*inline code*⟩ once for each element of ⟨*matrix*⟩. Within the inline code, `##1` represents the value of the current element. Since the code is passed as an argument to the mapping command, the parameter marker must be doubled.

For example, the following code writes the matrix elements to the terminal in sequence:

```
\matrix_map_inline:Nn \l_demo_matrix
  { \iow_term:n { ##1 } }
```

\matrix_map_break: \matrix_map_break:
\matrix_map_break:n \matrix_map_break:n {⟨*code*⟩}

Calling \matrix_map_break: within a mapping callback terminates the current mapping early. In addition to terminating the mapping, \matrix_map_break:n executes ⟨*code*⟩ when the mapping exits.

2.9 Displaying and Inspecting Matrices

\matrix_use:N \matrix_use:N ⟨*matrix*⟩
\matrix_use:c

Typesets ⟨*matrix*⟩ in mathematical mode. Non-empty matrices are displayed as parenthesized matrices, while empty matrices display their dimensions, such as 2×0. This command does not enter mathematical mode automatically and must therefore be used in mathematical mode.

\matrix_log:N \matrix_log:N ⟨*matrix*⟩
\matrix_log:c

Writes the dimensions, storage information, and element contents of ⟨*matrix*⟩ to the compilation log for inspection and debugging. This command does not typeset the matrix in the document body.

2.10 Predefined Matrices and Scratch Variables

\c_empty_matrix

The empty 0×0 matrix.

\c_identity_two_matrix
\c_identity_three_matrix
\c_identity_four_matrix

The 2×2, 3×3, and 4×4 identity matrices, respectively.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}、\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}、\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

\g_tmpa_matrix
\g_tmpb_matrix

Scratch matrix variables for global assignments. These variables are not used by the package's internal code and can therefore be safely used with commands provided by this package. However, they may be overwritten by other code and should only be used for short-term storage.

2.11 LaTeX2e Commands

The package is intended to provide some LaTeX2e-style command interfaces. As the design and implementation of these commands have not yet been completed, this section is currently empty.

2.12 Error Conditions

The current implementation reports an error under the following conditions:

- The number of rows or columns is negative when creating a matrix;
- The matrix variable has already been defined when creating a matrix;
- A constant matrix is used as the target of a modification operation;
- The number of items in the element list exceeds the total number of elements in the matrix when creating or setting a matrix;
- The dimensions of the two operands do not match when performing an element-wise binary operation between matrices;
- The number of columns in the left operand does not match the number of rows in the right operand when performing matrix multiplication.

3 实现设计

一个矩阵对象由矩阵描述符和矩阵数据两部分组成。矩阵变量直接存储矩阵描述符，而矩阵描述符中的一个字段记录矩阵数据的存储位置。

矩阵数据实际存储在 fparray 数组中。由于后者的数据修改采用全局操作，为保证矩阵描述符与矩阵数据的更新保持同步，两者的更新也均采用全局操作。因此，矩阵的创建和修改不会因普通 TEX 分组结束而自动恢复。

3.1 矩阵描述符

矩阵描述符是一个特殊的凭据表，由以下四个整数字段组成：`<length>`、`<rows>`、`<columns>` 和 `<space-index>`。各字段之间使用专用扫描标记分隔，并以结束标记结尾。各字段的含义如下：

`<length>` 矩阵的元素总数，即行数与列数的乘积；

`<rows>` / `<columns>` 矩阵的行数和列数；

`<space-index>` 矩阵在所属存储类别中的逻辑索引。

3.2 存储管理

矩阵数据存储在 fparray 数组的一段连续空间中。存储管理器负责这段空间的分配与定位，并根据矩阵的元素总数选择相应的存储类别。

3.2.1 存储类别与数组组织

通常，存储管理器按照固定的空间容量，将 fparray 数组划分为若干个独立空间，每个空间用于存储一个矩阵。目前，存储管理器提供 A、B、C 三种存储类别，分别用于存储小型、中型和大型矩阵。对于超大型矩阵，则采用 D 类存储方式，为每个矩阵单独创建数组。

各类别的当前参数如下：

类别	单个空间容量	每个数组可容纳的空间数
A	20	200
B	100	40
C	1000	4
D	按矩阵长度分配	每个数组对应一个矩阵

3.2.2 逻辑索引与空间定位

矩阵的存储类别由其元素总数决定，具体划分如下：元素总数不超过 20 时使用 A 类；超过 20 且不超过 100 时使用 B 类；超过 100 且不超过 1000 时使用 C 类；超过 1000 时使用 D 类。

`<space-index>` 表示矩阵在所属存储类别中的逻辑索引。矩阵的元素总数决定其存储类别，而存储类别与逻辑索引共同确定矩阵数据的实际存储位置。

对于 A、B、C 类，存储管理器根据逻辑索引以及该类别每个数组所能容纳的空间数，计算矩阵所属的数组编号和空间基址。数组名称由存储类别字母和数组编号组成，空间基址从 1 开始。

对于 D 类，每个矩阵对应一个单独的数组，逻辑索引对应其数组。数组基址始终为 1，数组长度由矩阵的元素总数决定。

3.2.3 分配与回收

空间分配优先使用回收序列中的索引；没有可回收空间时，才使用该类别的下一个索引。元素总数为零时不分配实际空间。普通空矩阵使用索引 0，常量空矩阵使用索引 -1；这两个值是空矩阵的逻辑标记，不对应任何数组或数组区间。

回收时，A、B、C 类只将空间索引放回对应的回收序列，底层数组继续保留，后续分配可以复用该空间。D 类回收时，除登记索引外，还会删除对应的独立数组。空间回收函数本身不判断矩阵是否为常量；常量空间是否允许回收由矩阵操作层保证。

3.3 常量矩阵

常量矩阵不使用额外的常量标记字段。宏包使用一个全局序列保存常量矩阵的完整描述符内容，并通过描述符内容比较判断矩阵是否为常量。修改矩阵之前，矩阵操作层检查目标描述符是否存在于该序列中；匹配时报告常量矩阵错误，并中止当前操作。

这种识别方式以描述符内容而不是控制序列名称为依据。因此，复制或等价引用常量矩阵描述符后，只要描述符内容不变，仍会被识别为常量矩阵。常量矩阵的非空数据仍按正常类别规则分配空间。

常量保护由矩阵操作接口实现，存储管理器不承担常量判断。该保护针对宏包提供的矩阵修改接口，不等同于对底层 fparray 变量的访问控制。

3.4 零矩阵与空矩阵

零矩阵是所有元素均为零的矩阵，不是一种单独的存储类别。零矩阵仍按其元素总数选择 A、B、C 或 D 类空间；归零操作直接将其现有空间置零。

空矩阵是元素总数为零的合法矩阵，包括 0×0、$m \times 0$ 和 $0 \times n$。空矩阵不分配实际存储空间，但行数和列数仍保留在描述符中。因此，不同维度的空矩阵仍是不同形状的矩阵，参与运算时仍须执行维度兼容性检查。

对于结果元素总数为零的运算，结果描述符仍会按目标维度更新，但不会分配空间。

4 宏包准备

```
⟨*package⟩
⟨@@=matrix⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{lt3matrix}{2026-11-16}{0.1}
  {a lightweight global matrix base-on fparray}
```

定义一些 expl3 函数的常用变体。

```
\cs_generate_variant:Nn \iow_log:n { V, e }
```

\l__matrix_tmpa_int
\l__matrix_tmpb_int
\l__matrix_tmpa_fp
\l__matrix_tmpb_fp
\l__matrix_tmpa_tl
\l__matrix_tmpb_tl

定义一些通用的临时变量。

```
\int_new:N  \l__matrix_tmpa_int
\int_new:N  \l__matrix_tmpb_int
\fp_new:N   \l__matrix_tmpa_fp
\fp_new:N   \l__matrix_tmpb_fp
\tl_new:N   \l__matrix_tmpa_tl
\tl_new:N   \l__matrix_tmpb_tl
```

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

5 核心命令

5.1 消息

negative-dimension
constant-matrix
too-many-elements
dimension-mismatch

错误消息。

```
\msg_new:nnn { matrix } { negative-dimension }
  {
    Matrix~dimensions~'#1'~must~be~non-negative.
  }
\msg_new:nnn { matrix } { constant-matrix }
  {
    The~matrix~'#1'~is~constant~and~cannot~be~modified.
  }
\msg_new:nnn { matrix } { too-many-elements }
  {
```

```
    The~input~list~'#1'~contains~too~many~elements.
  }
\msg_new:nnn { matrix } { dimension-mismatch }
  {
    The~dimensions~of~the~operands~are~incompatible~for~this~operation.
  }
```

(*End of definition for* `negative-dimension` *and others. These variables are documented on page* **??**.)

5.2 元信息

矩阵的元信息包含总长度、行数、列数及实际数据索引，索引指向实际数据。

5.2.1 基础变量

\s__matrix 存储数据用的扫描标记。

\s__matrix_mark

\s__matrix_stop

```
\scan_new:N \s__matrix
\scan_new:N \s__matrix_mark
\scan_new:N \s__matrix_stop
```

(*End of definition for* `\s__matrix`, `\s__matrix_mark`, *and* `\s__matrix_stop`.)

\g__matrix_const_seq 常量矩阵登记表。

```
\seq_new:N \g__matrix_const_seq
```

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

\l__matrix_length_int 保存矩阵信息的临时变量。

\l__matrix_row_int

\l__matrix_col_int

\l__matrix_index_int

```
\int_new:N  \l__matrix_length_int
\int_new:N  \l__matrix_row_int
\int_new:N  \l__matrix_col_int
\int_new:N  \l__matrix_index_int
```

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

5.2.2 设置信息

__matrix_meta_gset:N 将 `#1` 的值设置为矩阵基本信息。

__matrix_meta_gset:Nnnnn

```
\cs_new:Nn \__matrix_meta_gset:Nnnnn
  {
    \tl_gset:Nn #1
      {
        \s__matrix      #2 % Length
        \s__matrix_mark #3 % Row Count
```

```
        \s__matrix_mark #4 % Column Count
        \s__matrix_mark #5 % Index
        \s__matrix_stop
      }
  }
\cs_generate_variant:Nn \__matrix_meta_gset:Nnnnn { NVVVV }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_meta_gset:N
  {
    \__matrix_meta_gset:NVVVV #1
      \l__matrix_length_int
      \l__matrix_row_int
      \l__matrix_col_int
      \l__matrix_index_int
  }
```

(*End of definition for* `\__matrix_meta_gset:N` *and* `\__matrix_meta_gset:Nnnnn`.)

5.2.3 解析信息

`\__matrix_parse_aux:w` 解析矩阵信息，并结果设置到 #5~#8。

```
\cs_new:Npn \__matrix_parse_aux:w
  \s__matrix      #1
  \s__matrix_mark #2
  \s__matrix_mark #3
  \s__matrix_mark #4
  \s__matrix_stop #5#6#7#8
  {
    \int_set:Nn #5 {#1}
    \int_set:Nn #6 {#2}
    \int_set:Nn #7 {#3}
    \int_set:Nn #8 {#4}
  }
```

(*End of definition for* `\__matrix_parse_aux:w`.)

`\__matrix_meta_parse:N` 解析矩阵信息，并结果设置到 #2~#5。

`\__matrix_meta_parse:NNNNN`

```
\cs_new:Nn \__matrix_meta_parse:NNNNN
  {
    \exp_last_unbraced:NV
      \__matrix_parse_aux:w #1 #2#3#4#5
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_meta_parse:N
  {
    \exp_last_unbraced:NV
      \__matrix_parse_aux:w #1
      \l__matrix_length_int
      \l__matrix_row_int
      \l__matrix_col_int
      \l__matrix_index_int
  }
```

(*End of definition for* `\__matrix_meta_parse:N` *and* `\__matrix_meta_parse:NNNNN`.)

5.3 数据池

在实际实现中，每个 `fparray` 按固定大小划分为若干空间，每个空间只存储一份矩阵数据。根据空间大小，预定义小（A 类）、中（B 类）和大（C 类）三种存储池，以适应不同大小的矩阵。

In practice, each `fparray` is divided into a number of fixed-size spaces, each of which stores one matrix. Three storage pool classes are predefined according to space size: small (A), medium (B), and large (C), to accommodate matrices of different sizes.

对于元素总数超过 C 类空间容量的矩阵（D 类），则为每个矩阵单独分配一个数组。

Matrices whose number of elements exceeds the capacity of a C-class space (D class) are allocated a separate array.

5.3.1 常量

`\c__matrix_size_a_int` `\c__matrix_size_b_int` `\c__matrix_size_c_int`

三种划分 fparray 对应的空间大小。

```
\int_const:Nn \c__matrix_size_a_int { 20   }
\int_const:Nn \c__matrix_size_b_int { 100  }
\int_const:Nn \c__matrix_size_c_int { 1000 }
```

(*End of definition for* `\c__matrix_size_a_int`, `\c__matrix_size_b_int`, *and* `\c__matrix_size_c_int`.)

`\c__matrix_array_length_int`

每个 fparray 变量的长度。

```
\int_const:Nn \c__matrix_array_length_int { 4000 }
```

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

\c__matrix_count_a_int
\c__matrix_count_b_int
\c__matrix_count_c_int

单个 fparray 实际划分出的空间数量。

```
\int_const:Nn \c__matrix_count_a_int
  { \c__matrix_array_length_int / \c__matrix_size_a_int }
\int_const:Nn \c__matrix_count_b_int
  { \c__matrix_array_length_int / \c__matrix_size_b_int }
\int_const:Nn \c__matrix_count_c_int
  { \c__matrix_array_length_int / \c__matrix_size_c_int }
```

(*End of definition for* `\c__matrix_count_a_int`, `\c__matrix_count_b_int`, *and* `\c__matrix_count_c_int`.)

5.3.2 变量

\g__matrix_a_i_fparray
\g__matrix_b_i_fparray
\g__matrix_c_i_fparray
\g__matrix_d_i_fparray

四种数据池演示，当前不实际申请（使用时自动申请）。
数据池的变量名形式为：`\g__matrix_<type>_<array index>_faparray`。

```
⟨demo⟩\fparray_new:Nn \g__matrix_a_i_fparray { \c__matrix_array_length_int }
⟨demo⟩\fparray_new:Nn \g__matrix_b_i_fparray { \c__matrix_array_length_int }
⟨demo⟩\fparray_new:Nn \g__matrix_c_i_fparray { \c__matrix_array_length_int }
⟨demo⟩\fparray_new:Nn \g__matrix_d_i_fparray { \l__matrix_length_int }
```

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

\g__matrix_next_a_int
\g__matrix_next_b_int
\g__matrix_next_c_int
\g__matrix_next_d_int

数据池中的下一个空间索引。

```
\int_new:N    \g__matrix_next_a_int
\int_new:N    \g__matrix_next_b_int
\int_new:N    \g__matrix_next_c_int
\int_new:N    \g__matrix_next_d_int

\int_gset:Nn  \g__matrix_next_a_int { 1 }
\int_gset:Nn  \g__matrix_next_b_int { 1 }
\int_gset:Nn  \g__matrix_next_c_int { 1 }
\int_gset:Nn  \g__matrix_next_d_int { 1 }
```

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

\g__matrix_free_a_seq
\g__matrix_free_b_seq
\g__matrix_free_c_seq
\g__matrix_free_d_seq

数据池回收的空间索引。

```
\seq_new:N    \g__matrix_free_a_seq
\seq_new:N    \g__matrix_free_b_seq
\seq_new:N    \g__matrix_free_c_seq
\seq_new:N    \g__matrix_free_d_seq
```

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

\l__matrix_space_class_tl
\l__matrix_space_size_int

用于存储空间属性的临时变量。

```
\tl_new:N   \l__matrix_space_class_tl
\int_new:N  \l__matrix_space_size_int
```

(*End of definition for* `\l__matrix_space_class_tl` *and* `\l__matrix_space_size_int`.)

`\l__matrix_fparray` 用于存储实际分配的空间的信息的临时变量。

`\l__matrix_base_int`

```
\cs_new:Npn \l__matrix_fparray {}
\int_new:N  \l__matrix_base_int
```

(*End of definition for* `\l__matrix_fparray` *and* `\l__matrix_base_int`.)

5.3.3 空间选择

`\__matrix_space_select:` 查询某个长度（`#1`）的矩阵对应的空间的相关属性（类型与大小）。结果保存在 `\l__matrix_space_type_tl` 与 `\l__matrix_space_size_int` 中。

`\__matrix_space_select:n`

```
\cs_new:Nn \__matrix_space_select:n
  {
    \int_compare:nNnTF {#1} > { \c__matrix_size_a_int }
      {
        \int_compare:nNnTF {#1} > { \c__matrix_size_b_int }
          {
            \int_compare:nNnTF {#1} > { \c__matrix_size_c_int }
              { % Class D
                \tl_set:Nn  \l__matrix_space_class_tl {d}
                \int_set:Nn \l__matrix_space_size_int {#1}
              }
              {  % Class C
                \tl_set:Nn  \l__matrix_space_class_tl {c}
                \int_set:Nn \l__matrix_space_size_int { \c__matrix_size_c_int }
              }
          }
          { % Class B
              \tl_set:Nn  \l__matrix_space_class_tl {b}
              \int_set:Nn \l__matrix_space_size_int { \c__matrix_size_b_int }
          }
      }
      {
        \int_compare:nNnTF {#1} = { 0 }
          { % Empty
            \tl_set:Nn  \l__matrix_space_class_tl {}
            \int_set:Nn \l__matrix_space_size_int { 0 }
          }
          { % Class A
            \tl_set:Nn  \l__matrix_space_class_tl {a}
            \int_set:Nn \l__matrix_space_size_int { \c__matrix_size_a_int }
          }
```

```
    }
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_space_select:
  {
    \__matrix_space_select:n { \l__matrix_length_int }
  }
```

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

5.3.4 空间分配

`\__matrix_space_allocate_aux:n` 获取下一个可用空间索引。`#1` 空间类型，结果保存在 `\l__matrix_index_int` 中。

```
\cs_new:Nn \__matrix_space_allocate_aux:n
  {
    \seq_if_empty:cTF { g__matrix_free_#1_seq }
      { % 无回收空间，使用下一个空间
        \int_set_eq:Nc \l__matrix_index_int { g__matrix_next_#1_int  }
        \int_set_eq:Nc \l__matrix_tmpa_int  { c__matrix_count_#1_int }
        \int_compare:nNnT
          { \int_mod:nn { \l__matrix_index_int } { \l__matrix_tmpa_int } } = { 1 }
          { % 需要新数组
            \int_set:Nn \l__matrix_tmpb_int
              { \int_div_truncate:nn { \l__matrix_index_int - 1} { \l__matrix_tmpa_int } }
            \int_incr:N \l__matrix_tmpb_int
            \exp_args:Nc \fparray_new:Nn
              { g__matrix_#1_ \int_to_roman:n { \l__matrix_tmpb_int } _fparray }
              { \c__matrix_array_length_int }
          }
          \int_gincr:c { g__matrix_next_#1_int }
      }
      { % 有回收空间，直接使用
        \seq_gpop_left:cN { g__matrix_free_#1_seq } \l__matrix_tmpa_tl
        \int_set:Nn \l__matrix_index_int { \l__matrix_tmpa_tl }
      }
  }
\cs_generate_variant:Nn \__matrix_space_allocate_aux:n { V }
```

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

`\__matrix_space_allocate_aux_d:n` 获取下一个可用空间索引（`D` 类型）。`#1` 空间大小，结果保存在 `\l__matrix_index_int` 中。

```
\cs_new:Nn \__matrix_space_allocate_aux_d:n
```

```
  {
    \seq_if_empty:NTF \g__matrix_free_d_seq
      { % 无回收空间，使用下一个空间
        \int_set_eq:NN \l__matrix_index_int \g__matrix_next_d_int
        \int_gincr:N \g__matrix_next_d_int
      }
      { % 有回收空间，直接使用
        \seq_gpop_left:NN \g__matrix_free_d_seq \l__matrix_tmpa_tl
        \int_set:Nn \l__matrix_index_int { \l__matrix_tmpa_tl }
      }
      \exp_args:Nc \fparray_new:Nn
        { g__matrix_d_ \int_to_roman:n { \l__matrix_index_int } _fparray }
        { #1 }
  }
```

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

`\__matrix_space_allocate:` 申请空间。`#1` 空间大小，结果保存在 `\l__matrix_index_int` 中。

`\__matrix_space_allocate:n`

```
\cs_new:Nn \__matrix_space_allocate:n
  {
    \int_compare:nNnTF {#1} = { 0 }
      { \int_set:Nn \l__matrix_index_int { 0 } } % Emtpy Matrices
      {
        \__matrix_space_select:n {#1}
        \int_compare:nNnTF {#1} > { \c__matrix_size_c_int }
          { \__matrix_space_allocate_aux_d:n {#1} } % Class D
          { \__matrix_space_allocate_aux:V \l__matrix_space_class_tl } % Class A ~ C
      }
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_space_allocate:
  {
    \__matrix_space_allocate:n { \l__matrix_length_int }
  }
```

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

5.3.5 空间回收

`\__matrix_space_free_aux:nn` 回收空间辅助函数。`#1` 空间大小，`#2` 空间索引。

```
\cs_new:Nn \__matrix_space_free_aux:nn
  {
    \__matrix_space_select:n {#1}
```

```
    \seq_gput_right:cn { g__matrix_free_ \l__matrix_space_class_tl _seq } {#2}
    \int_compare:nNnT {#1} > { \c__matrix_size_c_int }
      { \cs_undefine:c { g__matrix_d_ \int_to_roman:n { #2 } _fparray } }
  }
\cs_generate_variant:Nn \__matrix_space_free_aux:nn { VV }
```

(*End of definition for* `\__matrix_space_free_aux:nn`.)

`\__matrix_space_free:` 回收空间函数。#1 空间大小，#2 空间索引。不负责常量空间校对。

`\__matrix_space_free:nn`

```
\cs_new:Nn \__matrix_space_free:nn
  {
    \int_compare:nNnT {#1} > {0}
      {
        \int_set:Nn \l__matrix_tmpa_int {#1}
        \int_set:Nn \l__matrix_tmpb_int {#2}
        \__matrix_space_free_aux:VV \l__matrix_tmpa_int \l__matrix_tmpb_int
      }
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_space_free:
  {
    \__matrix_space_free:nn
      { \l__matrix_length_int }
      { \l__matrix_index_int  }
  }
```

(*End of definition for* `\__matrix_space_free:` *and* `\__matrix_space_free:nn`.)

5.3.6 查询定位

`\__matrix_space_query_aux:nn` 查询指定类型的空间的信息。#1 空间类型，#2 空间索引。结果保存在 `\l__matrix_fparray` 与 `\l__matrix_base_int` 中。

```
\cs_new:Nn \__matrix_space_query_aux:nn
  {
    \int_set_eq:Nc \l__matrix_tmpa_int { c__matrix_count_#1_int }
    \int_set:Nn    \l__matrix_tmpb_int
      { \int_div_truncate:nn { #2 - 1 } { \l__matrix_tmpa_int } }
    \int_incr:N    \l__matrix_tmpb_int
    \cs_set_eq:Nc  \l__matrix_fparray
      { g__matrix_#1_ \int_to_roman:n { \l__matrix_tmpb_int } _fparray }
    \int_set:Nn    \l__matrix_base_int
      {
        \int_mod:nn { #2 - 1 } { \l__matrix_tmpa_int } *
```

```
        \int_use:c  { c__matrix_size_#1_int } + 1
      }
  }
\cs_generate_variant:Nn \__matrix_space_query_aux:nn { V }
```

(*End of definition for* `\__matrix_space_query_aux:nn`.)

`\__matrix_space_query:`
`\__matrix_space_query:nn`

查询空间信息。#1 空间大小，#2 空间索引。结果保存在 `\l__matrix_fparray` 与 `\l__matrix_base_int` 中。

```
\cs_new:Nn \__matrix_space_query:nn
  {
    \__matrix_space_select:n {#1}
    \int_compare:nNnTF {#1} = { 0 }
      {
        \int_zero:N \l__matrix_base_int
        \cs_set:Npn \l__matrix_fparray {}
      }
      {
        \int_compare:nNnTF {#1} > { \c__matrix_size_c_int }
          { % Class D
            \int_set:Nn   \l__matrix_base_int { 1 }
            \cs_set_eq:Nc \l__matrix_fparray
              { g__matrix_d_ \int_to_roman:n {#2} _fparray }
          }
          { \__matrix_space_query_aux:Vn \l__matrix_space_class_tl {#2} } % Class A ~ C
      }
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_space_query:
  {
    \__matrix_space_query:nn
      { \l__matrix_length_int }
      { \l__matrix_index_int  }
  }
```

(*End of definition for* `\__matrix_space_query:` *and* `\__matrix_space_query:nn`.)

5.3.7 数据设置

`\__matrix_data_gzero:`
`\__matrix_data_gzero:Nnn`

将指定 fparray 的指定区间的数据置零。#1 数组，#2 起点，#3 长度。

```
\cs_new:Nn \__matrix_data_gzero:Nnn
  {
    \int_step_inline:nnn {#2} { #2 + #3 - 1 }
```

```
      { \fparray_gset:Nnn #1 {##1} { 0 } }
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_data_gzero:
  {
    \__matrix_data_gzero:Nnn \l__matrix_fparray
      { \l__matrix_base_int }
      { \l__matrix_space_size_int }
  }
```

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

`\__matrix_data_gset:n`
`\__matrix_data_gset:Nnnn`

使用指定数据将填充 fparray 的指定区间，数据不足时补零。#1 数组，#2 起点，#3 长度，#4 指定数据。

```
\cs_new:Nn \__matrix_data_gset:Nnnn
  {
    \int_set:Nn \l__matrix_tmpa_int {#2}
    \clist_map_inline:nn {#4} % Fill Data
      {
        \fparray_gset:Nnn #1 { \l__matrix_tmpa_int } {##1}
        \int_incr:N \l__matrix_tmpa_int
      }
    \int_step_inline:nnn      % Fill 0
      { \l__matrix_tmpa_int } { #2 + #3 - 1 }
      { \fparray_gset:Nnn #1 {##1} { 0 } }
  }
```

这是一个使用临时变量的简化版本。

```
\cs_new:Nn \__matrix_data_gset:n
  {
    \__matrix_data_gset:Nnnn \l__matrix_fparray
      { \l__matrix_base_int }
      { \l__matrix_space_size_int }
      {#1}
  }
```

(*End of definition for* `\__matrix_data_gset:n` *and* `\__matrix_data_gset:Nnnn`.)

`\__matrix_data_gcopy:NnNnn`

使两块区域的内容完成相等。#1 目标数组，#2 目标起点，#3 源数组，#4 源起点，#4 长度。

```
\cs_new:Nn \__matrix_data_gcopy:NnNnn
  {
    \int_step_inline:nnn { 0 } { #5 - 1 }
```

```
      {
        \fparray_gset:Nnn #1 { #2 + ##1 }
          { \fparray_item:Nn #3 { #4 + ##1 } }
      }
  }
```

(*End of definition for* `\__matrix_data_gcopy:NnNnn`.)

5.3.8 数据输出

`\__matrix_data_typeset:Nnnn` 在公式环境中排版 fparray 内容，命令本身不会自动切换到公式环境。#1 数组，#2 起点，#3 长度，#4 每行列数。

```
\cs_new:Nn \__matrix_data_typeset:Nnnn
  {
    \tl_set:Nn \l__matrix_tmpa_tl
      { \left( \begin{array}{*{#4}{c}} }
    \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
      {
        \tl_put_right:Ne \l__matrix_tmpa_tl
          { \fparray_item:Nn #1 {##1} }
        \int_compare:nNnTF
          { \int_mod:nn { ##1 - #2 + 1} { #4 } } = { 0 }
          { \tl_put_right:Nn \l__matrix_tmpa_tl { \\ } }
          { \tl_put_right:Nn \l__matrix_tmpa_tl { & } }
      }
    \regex_replace_once:nnN { \c{\\}\Z } {} \l__matrix_tmpa_tl
    \tl_put_right:Nn \l__matrix_tmpa_tl
      { \end{array} \right) }
    \l__matrix_tmpa_tl
  }
```

(*End of definition for* `\__matrix_data_typeset:Nnnn`.)

`\__matrix_data_log:Nnnn` 在日志中查看 fparray 内容。#1 数组，#2 起点，#3 长度，#4 每行列数。

```
\cs_new:Nn \__matrix_data_log:Nnnn
  {
    \int_set:Nn \l__matrix_tmpa_int { 1 }
    \tl_set:Ne \l__matrix_tmpa_tl
      { <\int_use:N \l__matrix_tmpa_int>:~ }
    \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
      {
        \tl_put_right:Ne \l__matrix_tmpa_tl
          { \fparray_item:Nn #1 {##1} }
```

```
        \int_compare:nNnTF
          { \int_mod:nn { ##1 - #2 + 1} { #4 } } = { 0 }
          {
            \iow_log:V  \l__matrix_tmpa_tl
            \int_incr:N \l__matrix_tmpa_int
            \tl_set:Ne  \l__matrix_tmpa_tl
              { <\int_use:N \l__matrix_tmpa_int>:~ }
          }
          { \tl_put_right:Nn \l__matrix_tmpa_tl  { ,~ } }
      }
  }
```

(*End of definition for* `\__matrix_data_log:Nnnn`.)

`\__matrix_data_log:Nnnnn` 在日志中查看 fparray 内容。#1 数组，#2 起点，#3 长度，#4 每行列数, #5 空间长度。

```
\cs_new:Nn \__matrix_data_log:Nnnnn
  {
    \__matrix_data_log:Nnnn #1 {#2} {#3} {#4}
    \tl_set:Nn \l__matrix_tmpa_tl { <unused>: }
    \int_step_inline:nnn { #3 } { #5 - 1 }
      {
        \tl_put_right:Ne \l__matrix_tmpa_tl
          { ~\fparray_item:Nn #1 { #2 + ##1 }, }
      }
    \int_compare:nNnTF {#3} = {#5}
      { \tl_put_right:Nn \l__matrix_tmpa_tl { ~none. }  }
      { \regex_replace_once:nnN { ,\Z } { . } \l__matrix_tmpa_tl  }
    \iow_log:V \l__matrix_tmpa_tl
  }
```

(*End of definition for* `\__matrix_data_log:Nnnnn`.)

6 矩阵命令

6.1 创建与初始化矩阵

6.1.1 创建矩阵

`\matrix_new:Nnn` 定义新矩阵。

`\matrix_new:cnn`

```
\cs_new_protected:Nn \matrix_new:Nnn
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn {#2} < {0} }
```

```
      { \int_compare_p:nNn {#3} < {0} }
      { \msg_error:nnn { matrix } { negative-dimension } {#2x#3} }
      {
        \tl_new:N #1  % 防止变量已使用
        \int_set:Nn \l__matrix_length_int { #2 * #3 }
        \int_set:Nn \l__matrix_row_int {#2}
        \int_set:Nn \l__matrix_col_int {#3}
        \int_compare:nNnTF
          { \l__matrix_length_int } = { 0 }
          { \int_set:Nn \l__matrix_index_int { 0 } } % Empty Matrices
          { % Non-Empty Matrices
            \__matrix_space_allocate:
            \__matrix_space_query:
            \__matrix_data_gzero:
          }
        \__matrix_meta_gset:N #1
      }
  }
\cs_generate_variant:Nn \matrix_new:Nnn { c }
```

(*End of definition for* `\matrix_new:Nnn`. *This function is documented on page* 9.)

`\matrix_const:Nnnn`
`\matrix_const:cnnn`

定义新矩阵。

```
\cs_new_protected:Nn \matrix_const:Nnnn
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn {#2} < {0} }
      { \int_compare_p:nNn {#3} < {0} }
      { \msg_error:nnn { matrix } { negative-dimension } {#2x#3} }
      {
        \int_set:Nn \l__matrix_length_int { #2 * #3 }
        \int_set:Nn \l__matrix_row_int {#2}
        \int_set:Nn \l__matrix_col_int {#3}
        \int_compare:nNnTF
          { \clist_count:n {#4} } > { \l__matrix_length_int }
          { \msg_error:nnn { matrix } { too-many-elements } {#4} }
          {
            \tl_new:N #1  % 防止变量已使用
            \int_compare:nNnTF
              { \l__matrix_length_int } = { 0 }
              { \int_set:Nn \l__matrix_index_int { -1 } } % Empty Matrices
              { % Non-Empty Matrices
                \__matrix_space_allocate:
```

```
                \__matrix_space_query:
                \__matrix_data_gset:n {#4}
              }
            \__matrix_meta_gset:N #1
            \seq_gput_right:NV \g__matrix_const_seq #1
          }
      }
  }
\cs_generate_variant:Nn \matrix_const:Nnnn { c }
```

(*End of definition for* `\matrix_const:Nnnn`. *This function is documented on page 9.*)

6.1.2 设置矩阵

`\matrix_gzero:N` 矩阵归零。

`\matrix_gzero:c`

```
\cs_new_protected:Nn \matrix_gzero:N
  {
    \seq_if_in:NVTF \g__matrix_const_seq #1
      { \msg_error:nnn { matrix } { constant-matrix } { \token_to_str:N #1 } }
      {
        \__matrix_meta_parse:N #1
        \int_compare:nNnT
          { \l__matrix_length_int } > { 0 }
          { \__matrix_space_query: \__matrix_data_gzero: }
      }
  }
\cs_generate_variant:Nn \matrix_gzero:N { c }
```

(*End of definition for* `\matrix_gzero:N`. *This function is documented on page 10.*)

`\matrix_gset:Nn` 设置矩阵的值。

`\matrix_gset:cn`

```
\cs_new_protected:Nn \matrix_gset:Nn
  {
    \seq_if_in:NVTF \g__matrix_const_seq #1
      { \msg_error:nnn { matrix } { constant-matrix } { \token_to_str:N #1 } }
      {
        \__matrix_meta_parse:N #1
        \int_compare:nNnTF
          { \clist_count:n {#2} } > { \l__matrix_length_int }
          { \msg_error:nnn { matrix } { too-many-elements } {#2} }
          { \__matrix_space_query: \__matrix_data_gset:n {#2} }
      }
  }
\cs_generate_variant:Nn \matrix_gset:Nn { c }
```

(*End of definition for* `\matrix_gset:Nn`. *This function is documented on page* 10.)

6.1.3 显示矩阵

`\matrix_use:N` 显示矩阵。

`\matrix_use:c`

```
\cs_new_protected:Nn \matrix_use:N
  {
    \__matrix_meta_parse:N #1
    \int_compare:nNnTF
      { \l__matrix_length_int } = { 0 }
      { \left(~<\int_use:N \l__matrix_row_int>~x~<\int_use:N \l__matrix_col_int>~\right) }
      {
        \__matrix_space_query:
        \__matrix_data_typeset:Nnnn \l__matrix_fparray
          { \l__matrix_base_int } { \l__matrix_length_int } { \l__matrix_col_int }
      }
  }
\cs_generate_variant:Nn \matrix_use:N { c }
```

(*End of definition for* `\matrix_use:N`. *This function is documented on page* 14.)

`\matrix_log:N` 查看矩阵。

```
\cs_new_protected:Nn \matrix_log:N
  {
    \__matrix_meta_parse:N #1
    \iow_log:n { The~matrix's~base~information:~ }
    \iow_log:V #1
    \int_compare:nNnTF { \l__matrix_length_int } > { 0 }
      {
        \__matrix_space_query:nn { \l__matrix_length_int } { \l__matrix_index_int }
        \iow_log:e { The~matrix's~data: <base:~\int_use:N \l__matrix_base_int>~ }
        \__matrix_data_log:Nnnnn \l__matrix_fparray
          { \l__matrix_base_int } { \l__matrix_length_int }
          { \l__matrix_col_int  } { \l__matrix_space_size_int }
      }
      { \iow_log:n { The~matrix~does~not~has~any~data~! } }
  }
\cs_generate_variant:Nn \matrix_log:N { c }
```

(*End of definition for* `\matrix_log:N`. *This function is documented on page* 14.)

6.1.4 预定义矩阵

\g_tmpa_matrix
\g_tmpb_matrix

本宏包永不使用的临时变量。

```
\matrix_new:Nnn  \g_tmpa_matrix { 0 } { 0 }
\matrix_new:Nnn  \g_tmpb_matrix { 0 } { 0 }
```

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

\c_empty_matrix
\c_identity_two_matrix
\c_identity_three_matrix
\c_identity_four_matrix

定义长度为零的特殊矩阵，零索引不指向任何实际数据。

```
\matrix_const:Nnnn \c_empty_matrix
  { 0 } { 0 } {}
\matrix_const:Nnnn \c_identity_two_matrix
  { 2 } { 2 } { 1, 0, 0, 1 }
\matrix_const:Nnnn \c_identity_three_matrix
  { 3 } { 3 } { 1, 0, 0, 0, 1, 0, 0, 0, 1 }
\matrix_const:Nnnn \c_identity_four_matrix
  { 4 } { 4 }
  {
    1, 0, 0, 0,
    0, 1, 0, 0,
    0, 0, 1, 0,
    0, 0, 0, 1
  }
```

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

6.2 映射与运算

6.2.1 运算专用变量

\l__matrix_opa_row_int
\l__matrix_opb_row_int
\l__matrix_opa_col_int
\l__matrix_opb_col_int
\l__matrix_old_length_int
\l__matrix_old_index_int

矩阵运算时使用的临时元数据。

```
\int_new:N  \l__matrix_opa_row_int
\int_new:N  \l__matrix_opb_row_int
\int_new:N  \l__matrix_opa_col_int
\int_new:N  \l__matrix_opb_col_int
\int_new:N  \l__matrix_old_length_int
\int_new:N  \l__matrix_old_index_int
```

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

\l__matrix_opa_fparray
\l__matrix_opb_fparray
\l__matrix_opa_base_int
\l__matrix_opb_base_int

矩阵运算时使用的临时数组信息。

```
\cs_new:Npn \l__matrix_opa_fparray {}
\cs_new:Npn \l__matrix_opb_fparray {}
\int_new:N  \l__matrix_opa_base_int
\int_new:N  \l__matrix_opb_base_int
```

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

6.2.2 运算准备与收尾

`\__matrix_left_operand_prepare:N` 操作数准备。

`\__matrix_right_operand_prepare:N`

```
\cs_new:Nn \__matrix_left_operand_prepare:N
  {
    \__matrix_meta_parse:N #1
    \int_set_eq:NN \l__matrix_opa_row_int    \l__matrix_row_int
    \int_set_eq:NN \l__matrix_opa_col_int    \l__matrix_col_int
    \__matrix_space_query:
    \cs_set_eq:NN  \l__matrix_opa_fparray    \l__matrix_fparray
    \int_set_eq:NN \l__matrix_opa_base_int   \l__matrix_base_int
  }
\cs_new:Nn \__matrix_right_operand_prepare:N
  {
    \__matrix_meta_parse:N #1
    \int_set_eq:NN \l__matrix_opb_row_int    \l__matrix_row_int
    \int_set_eq:NN \l__matrix_opb_col_int    \l__matrix_col_int
    \__matrix_space_query:
    \cs_set_eq:NN   \l__matrix_opb_fparray    \l__matrix_fparray
    \int_set_eq:NN  \l__matrix_opb_base_int   \l__matrix_base_int
  }
\cs_generate_variant:Nn \__matrix_left_operand_prepare:N  { c }
\cs_generate_variant:Nn \__matrix_right_operand_prepare:N { c }
```

(*End of definition for* `\__matrix_left_operand_prepare:N` *and* `\__matrix_right_operand_prepare:N`.)

`\__matrix_inplace_prepare:N` 原地修改准备，清空旧的空间信息以误回收。

```
\cs_new:Nn \__matrix_inplace_prepare:N
  {
    \seq_if_in:NVTF \g__matrix_const_seq #1
      {
        \msg_error:nnn { matrix } { constant-matrix } { \token_to_str:N #1 }
        \matrix_map_break:
      }
      {
        \__matrix_meta_parse:N #1
        \__matrix_space_query:
        \int_zero:N \l__matrix_old_length_int
        \int_zero:N \l__matrix_old_index_int
      }
  }
```

```
\cs_generate_variant:Nn \__matrix_inplace_prepare:N { c }
```

(*End of definition for* `\__matrix_inplace_prepare:N`.)

`\__matrix_result_prepare:Nnn` 结果准备，额外提供行列数，需要重新申请空间。旧空间不保留暂不回收。

```
\cs_new:Nn \__matrix_result_prepare:Nnn
  {
    \seq_if_in:NVTF \g__matrix_const_seq #1
      {
        \msg_error:nnn { matrix } { constant-matrix } { \token_to_str:N #1 }
        \matrix_map_break:
      }
      {
        \__matrix_meta_parse:N #1
        \int_set_eq:NN \l__matrix_old_length_int \l__matrix_length_int
        \int_set_eq:NN \l__matrix_old_index_int  \l__matrix_index_int
        \int_set:Nn \l__matrix_row_int {#2}
        \int_set:Nn \l__matrix_col_int {#3}
        \int_set:Nn \l__matrix_length_int { #2 * #3 }
        \__matrix_space_allocate:
        \__matrix_meta_gset:N #1
        \__matrix_space_query:
        \int_step_inline:nnn { \l__matrix_length_int } { \l__matrix_space_size_int - 1 }
          { \fparray_gset:Nnn \l__matrix_fparray { \l__matrix_base_int + ##1 } { 0 } }
      }
  }
\cs_generate_variant:Nn \__matrix_result_prepare:Nnn { c }
```

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

`\__matrix_operation_cleanup:` 运算收尾。

```
\cs_new:Nn \__matrix_operation_cleanup:
  {
    \__matrix_space_free:nn
      { \l__matrix_old_length_int } { \l__matrix_old_index_int }
    \int_zero:N \l__matrix_old_length_int
    \int_zero:N \l__matrix_old_index_int
  }
```

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

6.2.3 通用映射函数

\g__matrix_map_depth_int 映射嵌套深度变量。

```
\int_new:N  \g__matrix_map_depth_int
```

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

\g__matrix_map_intarray 有限的嵌套变量。

```
\intarray_new:Nn \g__matrix_map_intarray { 400 }
```

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

__matrix_map_state_save: 有限的嵌套变量的保存与恢复。
__matrix_map_state_restore:

```
\cs_new_protected:Nn \__matrix_map_state_save:
  {
    \int_set:Nn \l__matrix_tmpa_int { (\g__matrix_map_depth_int - 2) * 10 }
    \intarray_gset:Nnn \g__matrix_map_intarray
      { \l__matrix_tmpa_int + 1 } { \l__matrix_length_int }
    \intarray_gset:Nnn \g__matrix_map_intarray
      { \l__matrix_tmpa_int + 2 } { \l__matrix_row_int }
    \intarray_gset:Nnn \g__matrix_map_intarray
      { \l__matrix_tmpa_int + 3 } { \l__matrix_col_int }
    \intarray_gset:Nnn \g__matrix_map_intarray
      { \l__matrix_tmpa_int + 4 } { \l__matrix_index_int }
    \intarray_gset:Nnn \g__matrix_map_intarray
      { \l__matrix_tmpa_int + 5 } { \l__matrix_base_int }
  }
\cs_new_protected:Nn \__matrix_map_state_restore:
  {
    \int_set:Nn \l__matrix_tmpa_int { (\g__matrix_map_depth_int - 2) * 10 }
    \int_set:Nn \l__matrix_length_int
      { \intarray_item:Nn \g__matrix_map_intarray { \l__matrix_tmpa_int + 1 } }
    \int_set:Nn \l__matrix_row_int
      { \intarray_item:Nn \g__matrix_map_intarray { \l__matrix_tmpa_int + 2 } }
    \int_set:Nn \l__matrix_col_int
      { \intarray_item:Nn \g__matrix_map_intarray { \l__matrix_tmpa_int + 3 } }
    \int_set:Nn \l__matrix_index_int
      { \intarray_item:Nn \g__matrix_map_intarray { \l__matrix_tmpa_int + 4 } }
    \int_set:Nn \l__matrix_base_int
      { \intarray_item:Nn \g__matrix_map_intarray { \l__matrix_tmpa_int + 5 } }
  }
```

(*End of definition for* `\__matrix_map_state_save:` *and* `\__matrix_map_state_restore:`.)

__matrix_map_item:n
__matrix_map_push:
__matrix_map_pop:

当前映射元素处理函数。

```
\cs_new:Npn \__matrix_map_item:n #1 { }
```

保存当前元素处理函数。

```
\cs_new_protected:Nn \__matrix_map_push:
  {
    \int_gincr:N \g__matrix_map_depth_int
    \int_compare:nNnT
      { \g__matrix_map_depth_int } > { 1 }
      {
        \cs_gset_eq:cN
          { __matrix_map_item_ \int_use:N \g__matrix_map_depth_int :n }
          \__matrix_map_item:n
        \cs_gset_eq:cN
          { g__matrix_map_\int_use:N \g__matrix_map_depth_int _fparray }
          \l__matrix_fparray
        \__matrix_map_state_save:
      }
  }
```

恢复上一级元素处理函数。

```
\cs_new_protected:Nn \__matrix_map_pop:
  {
    \int_compare:nNnT
      { \g__matrix_map_depth_int } > { 1 }
      {
        \cs_gset_eq:Nc \__matrix_map_item:n
          { __matrix_map_item_ \int_use:N \g__matrix_map_depth_int :n }
        \cs_set_eq:Nc \l__matrix_fparray
          { g__matrix_map_\int_use:N \g__matrix_map_depth_int _fparray }
        \__matrix_map_state_restore:
      }
    \int_gdecr:N \g__matrix_map_depth_int
  }
```

(*End of definition for* `\__matrix_map_item:n`, `\__matrix_map_push:`, *and* `\__matrix_map_pop:`.)

\matrix_map_break:
\matrix_map_break:n

定义中断函数。

```
\cs_new_protected:Nn \matrix_map_break:
  { \prg_map_break:Nn \matrix_map_break: {  } }
\cs_new_protected:Nn \matrix_map_break:n
  { \prg_map_break:Nn \matrix_map_break: {#1} }
```

(*End of definition for* `\matrix_map_break:` *and* `\matrix_map_break:n`. *These functions are documented on page 14.*)

\matrix_map_function:NN 函数式映射函数。

```
\cs_new_protected:Nn \matrix_map_function:NN
  {
    \__matrix_map_push:
    \__matrix_meta_parse:N #1
    \int_compare:nNnT { \l__matrix_length_int } = { 0 }
      { \matrix_map_break: }
    \__matrix_space_query:
    \int_step_inline:nnn {0} { \l__matrix_length_int - 1 }
      {
        \exp_args:Ne #2
          { \fparray_item:Nn \l__matrix_fparray { \l__matrix_base_int + ##1 } }
      }
    \prg_break_point:Nn \matrix_map_break: { \__matrix_map_pop: }
  }
```

(*End of definition for* `\matrix_map_function:NN`. *This function is documented on page* 13.)

\matrix_map_inline:Nn 内联式映射函数。

```
\cs_new_protected:Nn \matrix_map_inline:Nn
  {
    \__matrix_map_push:
    \__matrix_meta_parse:N #1
    \cs_gset:Nn \__matrix_map_item:n {#2}
    \int_compare:nNnT { \l__matrix_length_int } = { 0 }
      { \matrix_map_break: }
    \__matrix_space_query:
    \int_step_inline:nnn {0} { \l__matrix_length_int - 1 }
      {
        \exp_args:Ne \__matrix_map_item:n
          { \fparray_item:Nn \l__matrix_fparray { \l__matrix_base_int + ##1 } }
      }
    \prg_break_point:Nn \matrix_map_break: { \__matrix_map_pop: }
  }
```

(*End of definition for* `\matrix_map_inline:Nn`. *This function is documented on page* 13.)

6.2.4 原地映射运算

__matrix_map_inplace:Nn 通过映射修改矩阵的元素。

```
\cs_new:Nn \__matrix_map_inplace:Nn
  {
    \__matrix_map_push:
```

```
    \__matrix_inplace_prepare:N #1
    \int_compare:nNnT { \l__matrix_length_int } = { 0 }
      { \matrix_map_break: }
    \cs_gset:Nn \__matrix_map_item:n {#2}
    \int_step_inline:nnn {0} { \l__matrix_length_int - 1 }
      {
        \fparray_gset:Nnn \l__matrix_fparray { \l__matrix_base_int + ##1 }
          {
            \exp_args:Ne \__matrix_map_item:n
              { \fparray_item:Nn \l__matrix_fparray { \l__matrix_base_int + ##1 } }
          }
      }
    \prg_break_point:Nn \matrix_map_break: { \__matrix_map_pop: }
  }
```

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

`\matrix_gnegate:N` `\matrix_gabs:N` `\matrix_gsign:N` `\matrix_ground:Nn`

求 $-A$、$|A|$、$sign(A)$ 与 $round(A,n)$。

```
\cs_new_protected:Nn \matrix_gnegate:N
  { \__matrix_map_inplace:Nn #1 { -##1 } }
\cs_new_protected:Nn \matrix_gabs:N
  { \__matrix_map_inplace:Nn #1 { abs(##1) }  }
\cs_new_protected:Nn \matrix_gsign:N
  { \__matrix_map_inplace:Nn #1 { sign(##1) } }
\cs_new_protected:Nn \matrix_ground:Nn
  { \__matrix_map_inplace:Nn #1 { round(##1, #2) } }
```

(*End of definition for* `\matrix_gnegate:N` *and others. These functions are documented on page* 11.)

`\matrix_gadd:Nn` `\matrix_gsub:Nn` `\matrix_gmul:Nn` `\matrix_gdiv:Nn` `\matrix_gpower:Nn`

求 $A+c$、$A-c$、cA、A/c 及 A^c。

```
\cs_new_protected:Nn \matrix_gadd:Nn
  { \__matrix_map_inplace:Nn #1 { ##1 + #2 } }
\cs_new_protected:Nn \matrix_gsub:Nn
  { \__matrix_map_inplace:Nn #1 { ##1 - #2 } }
\cs_new_protected:Nn \matrix_gmul:Nn
  { \__matrix_map_inplace:Nn #1 { ##1 * #2 } }
\cs_new_protected:Nn \matrix_gdiv:Nn
  { \__matrix_map_inplace:Nn #1 { ##1 / #2 } }
\cs_new_protected:Nn \matrix_gpower:Nn
  { \__matrix_map_inplace:Nn #1 { ##1 ^ #2 } }
```

(*End of definition for* `\matrix_gadd:Nn` *and others. These functions are documented on page* 11.)

6.2.5 单矩阵映射运算

__matrix_map_operate:NNn 通过映射逐元素计算，操作数只有一个矩阵。

```
\cs_new:Nn \__matrix_map_operate:NNn
  {
    \__matrix_map_push:
    \__matrix_left_operand_prepare:N #2
    \__matrix_result_prepare:Nnn #1 { \l__matrix_opa_row_int } { \l__matrix_opa_col_int }
    \int_compare:nNnT { \l__matrix_length_int } = { 0 }
      { \matrix_map_break: }
    \cs_gset:Nn \__matrix_map_item:n {#3}
    \int_step_inline:nnn {0} { \l__matrix_length_int - 1 }
      {
        \fparray_gset:Nnn \l__matrix_fparray { \l__matrix_base_int + ##1 }
          {
            \exp_args:Ne \__matrix_map_item:n
              { \fparray_item:Nn \l__matrix_opa_fparray { \l__matrix_opa_base_int + ##1 } }
          }
      }
    \prg_break_point:Nn \matrix_map_break:
      { \__matrix_map_pop: \__matrix_operation_cleanup: }
  }
```

(*End of definition for* __matrix_map_operate:NNn.)

\matrix_gset_eq:NN 矩阵赋值。

\matrix_gset_eq:Nc

\matrix_gset_eq:cN

```
\cs_new_protected:Nn \matrix_gset_eq:NN
  { \__matrix_map_operate:NNn #1 #2 {##1} }
\cs_generate_variant:Nn \matrix_gset_eq:NN { c, Nc }
```

(*End of definition for* \matrix_gset_eq:NN. *This function is documented on page* 10.)

\matrix_gnegate:NN 求 $Z=-A$、$Z=|A|$ 与 $Z=sign(A)$ 与 $Z=round(A,n)$。

\matrix_gabs:NN

\matrix_gsign:NN

\matrix_ground:NNn

```
\cs_new_protected:Nn \matrix_gnegate:NN
  { \__matrix_map_operate:NNn #1 #2 {-##1} }
\cs_new_protected:Nn \matrix_gabs:NN
  { \__matrix_map_operate:NNn #1 #2 { abs(##1) }  }
\cs_new_protected:Nn \matrix_gsign:NN
  { \__matrix_map_operate:NNn #1 #2 { sign(##1) } }
\cs_new_protected:Nn \matrix_ground:NNn
  { \__matrix_map_operate:NNn #1 #2 { round(##1, #3) } }
```

(*End of definition for* \matrix_gnegate:NN *and others. These functions are documented on page* 11.)

\matrix_gadd:Nn
\matrix_gsub:Nn
\matrix_gmul:Nn
\matrix_gdiv:Nn
\matrix_gpower:Nn

求 $Z = A + c$、$Z = A - c$、$Z = cA$、$Z = A/c$ 及 $Z = A^c$。

```
\cs_new_protected:Nn \matrix_gadd:NNn
  { \__matrix_map_operate:NNn #1 #2 { ##1 + #3 } }
\cs_new_protected:Nn \matrix_gsub:NNn
  { \__matrix_map_operate:NNn #1 #2 { ##1 - #3 } }
\cs_new_protected:Nn \matrix_gmul:NNn
  { \__matrix_map_operate:NNn #1 #2 { ##1 * #3 } }
\cs_new_protected:Nn \matrix_gdiv:NNn
  { \__matrix_map_operate:NNn #1 #2 { ##1 / #3 } }
\cs_new_protected:Nn \matrix_gpower:NNn
  { \__matrix_map_operate:NNn #1 #2 { ##1 ^ #3 } }
```

(*End of definition for* `\matrix_gadd:Nn` *and others. These functions are documented on page* 11.)

6.2.6 双矩阵映射运算

__matrix_map_operate:NNNn

通过映射逐元素计算，操作数有两个矩阵。

```
\cs_new:Nn \__matrix_map_operate:NNNn
  {
    \__matrix_map_push:
    \__matrix_left_operand_prepare:N  #2
    \__matrix_right_operand_prepare:N #3
    \bool_lazy_and:nnF
      { \int_compare_p:nNn \l__matrix_opa_row_int = \l__matrix_opb_row_int }
      { \int_compare_p:nNn \l__matrix_opa_col_int = \l__matrix_opb_col_int }
      {
        \msg_error:nn { matrix } { dimension-mismatch }
        \matrix_map_break:
      }
    \__matrix_result_prepare:Nnn #1 { \l__matrix_opa_row_int } { \l__matrix_opa_col_int }
    \int_compare:nNnT { \l__matrix_length_int } = { 0 }
      { \matrix_map_break: }
    \cs_gset:Npn \__matrix_map_item:n ##1 ##2 {#4}
    \int_step_inline:nnn {0} { \l__matrix_length_int - 1 }
      {
        \fparray_gset:Nnn \l__matrix_fparray { \l__matrix_base_int + ##1 }
          {
            \exp_args:Nee \__matrix_map_item:n
              { \fparray_item:Nn \l__matrix_opa_fparray { \l__matrix_opa_base_int + ##1 } }
              { \fparray_item:Nn \l__matrix_opb_fparray { \l__matrix_opb_base_int + ##1 } }
          }
      }
    \prg_break_point:Nn \matrix_map_break:
```

```
      { \__matrix_map_pop: \__matrix_operation_cleanup: }
  }
```

(*End of definition for* `\__matrix_map_operate:NNNn`.)

`\matrix_gadd:NNN` 求 $Z = A + B$、$Z = A - B$

`\matrix_gsub:NNN`

```
\cs_new_protected:Nn \matrix_gadd:NNN
  { \__matrix_map_operate:NNNn #1 #2 #3 { ##1 + ##2 } }
\cs_new_protected:Nn \matrix_gsub:NNN
  { \__matrix_map_operate:NNNn #1 #2 #3 { ##1 - ##2 } }
```

(*End of definition for* `\matrix_gadd:NNN` *and* `\matrix_gsub:NNN`. *These functions are documented on page 12.*)

`\matrix_gadd:NN` 求 $A + B$、$A - B$。

`\matrix_gsub:NN`

```
\cs_new_protected:Nn \matrix_gadd:NN
  { \__matrix_map_operate:NNNn #1 #1 #2 { ##1 + ##2 } }
\cs_new_protected:Nn \matrix_gsub:NN
  { \__matrix_map_operate:NNNn #1 #1 #2 { ##1 - ##2 } }
```

(*End of definition for* `\matrix_gadd:NN` *and* `\matrix_gsub:NN`. *These functions are documented on page 12.*)

6.3 非映射运算

`\__matrix_multiply_aux:nn` 计算 $Z_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$。注意，为减少运算，此处代码坐标从 0 计。

```
\cs_new:Nn \__matrix_multiply_aux:nn
  {
    \fp_zero:N \l__matrix_tmpa_fp
    \int_step_inline:nnn { 0 } { \l__matrix_opa_col_int - 1 }
      {
        \fp_add:Nn \l__matrix_tmpa_fp
          {
            \fparray_item:Nn \l__matrix_opa_fparray
              { \l__matrix_opa_base_int +  #1 * \l__matrix_opa_col_int + ##1 }
            *
            \fparray_item:Nn \l__matrix_opb_fparray
              { \l__matrix_opb_base_int + ##1 * \l__matrix_opb_col_int +  #2 }
          }
      }
    \fparray_gset:Nnn \l__matrix_fparray
      { \l__matrix_base_int + #1 * \l__matrix_col_int + #2 }
      { \l__matrix_tmpa_fp }
  }
```

(*End of definition for* `\__matrix_multiply_aux:nn`.)

\matrix_gmul:NNN 计算 $Z = AB$。

```
\cs_new_protected:Nn \matrix_gmul:NNN
  {
    \group_begin:
    \__matrix_left_operand_prepare:N  #2
    \__matrix_right_operand_prepare:N #3
    \int_compare:nNnF
      { \l__matrix_opa_col_int } = { \l__matrix_opb_row_int }
      {
        \msg_error:nn { matrix } { dimension-mismatch }
        \matrix_map_break:
      }
    \__matrix_result_prepare:Nnn #1 { \l__matrix_opa_row_int } { \l__matrix_opb_col_int }
    \int_compare:nNnT { \l__matrix_length_int } = { 0 }
      { \matrix_map_break: }
    \int_step_inline:nnn { 0 } { \l__matrix_opa_row_int - 1 }
      {
        \int_step_inline:nnn { 0 } { \l__matrix_opb_col_int - 1 }
          {
            \__matrix_multiply_aux:nn {##1} {####1}
          }
      }
    \prg_break_point:Nn \matrix_map_break:
      { \__matrix_operation_cleanup: }
    \group_end:
  }
```

(*End of definition for* `\matrix_gmul:NNN`. *This function is documented on page* 13.)

\matrix_gmul:NN 求 AB。

```
\cs_new_protected:Nn \matrix_gmul:NN
  { \matrix_gmul:NNN #1 #1 #2 }
```

(*End of definition for* `\matrix_gmul:NN`. *This function is documented on page* 13.)

(*End of definition for* .)

```
\ExplSyntaxOff
```

⟨/package⟩

Index

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

Symbols

\\ 314, 317

B

\begin 307

bool commands:

\bool_lazy_and:nnTF 724

\bool_lazy_or:nnTF 358, 381

C

\c 317

clist commands:

\clist_count:n 390, 427

\clist_map_inline:nn 280

constant-matrix 13

cs commands:

\cs_generate_variant:Nn 6, 48, 173, 211, 241, 378, 407, 419, 432, 445, 461, 506, 507, 522, 544, 700

\cs_gset:Nn 633, 650, 686

\cs_gset:Npn 734

\cs_gset_eq:NN 589, 592, 603

\cs_new:Nn 37, 49, 69, 74, 113, 146, 150, 174, 189, 200, 204, 212, 221, 227, 242, 260, 266, 271, 277, 289, 296, 304, 322, 342, 488, 497, 508, 523, 545, 644, 679, 719, 755

\cs_new:Npn 57, 111, 484, 485, 582

\cs_new_protected:Nn 356, 379, 408, 420, 433, 446, 554, 568, 583, 598, 611, 613, 615, 629, 661, 663, 665, 667, 669, 671, 673, 675, 677, 698, 701, 703, 705, 707, 709, 711, 713, 715, 717, 747, 749, 751, 753, 773, 798

\cs_set:Npn 248

\cs_set_eq:NN . 233, 254, 494, 503, 605

\cs_undefine:N 209

D

dimension-mismatch 13

E

\end 319

exp commands:

\exp_args:Nc 162, 185

\exp_args:Ne 624, 639, 655, 691

\exp_args:Nee 739

\exp_last_unbraced:NV 71, 76

\ExplSyntaxOff 800

\ExplSyntaxOn *2*, *9*

F

fp commands:

\fp_add:Nn 760

\fp_new:N 9, 10

\fp_zero:N 757

fparray commands:

\fparray_gset:Nnn 269, 282, 287, 300, 541, 653, 689, 737, 769

\fparray_item:Nn 301, 311, 330, 349, 625, 640, 656, 692, 740, 741, 762, 765

\fparray_new:Nn 93, 94, 95, 96, 162, 185

I

identity commands:

\c_identity_four_matrix . . *6*, *14*, 464

\c_identity_three_matrix . *6*, *14*, 464

\c_identity_two_matrix . . . *6*, *14*, 464

int commands:

\int_compare:nNnTF 115, 117, 119, 135, 156, 191, 195, 208, 214, 245, 251, 312, 331, 351, 367,

389, 394, 414, 426, 436, 451, 586, 600, 619, 634, 648, 684, 732, 778, 785
\int_compare_p:nNn 359, 360, 382, 383, 725, 726
\int_const:Nn 83, 84, 85, 86, 87, 89, 91
\int_div_truncate:nn 160, 231
\int_gdecr:N 609
\int_gincr:N 166, 179, 585
\int_gset:Nn 101, 102, 103, 104
\int_incr:N 161, 232, 283, 335
\int_mod:nn 157, 237, 313, 332
\int_new:N 7, 8, 33, 34, 35, 36, 97, 98, 99, 100, 110, 112, 478, 479, 480, 481, 482, 483, 486, 487, 552
\int_set:Nn 64, 65, 66, 67, 122, 126, 131, 138, 142, 159, 170, 183, 192, 216, 217, 230, 235, 253, 279, 324, 364, 365, 366, 369, 386, 387, 388, 396, 534, 535, 536, 556, 570, 571, 573, 575, 577, 579
\int_set_eq:NN . 154, 155, 178, 229, 491, 492, 495, 500, 501, 504, 532, 533
\int_step_inline:nnn 268, 285, 298, 308, 327, 346, 540, 622, 637, 651, 687, 735, 758, 787, 789
\int_to_roman:n 163, 186, 209, 234, 255
\int_use:N 238, 326, 337, 438, 454, 590, 593, 604, 606
\int_zero:N . . . 247, 518, 519, 549, 550

intarray commands:

\intarray_gset:Nnn 557, 559, 561, 563, 565
\intarray_item:Nn 572, 574, 576, 578, 580
\intarray_new:Nn 553

iow commands:

\iow_log:n 6, 334, 354, 449, 450, 454, 459

L

\left 307, 438

M

matrix commands:

\c_empty_matrix *6*, *14*, 464
\matrix_const:Nnnn *2*, *9*, 379, 379, 407, 464, 466, 468, 470
\matrix_gabs:N *3*, *11*, 661, 663
\matrix_gabs:NN *3*, *11*, 701, 703
\matrix_gadd:NN *5*, *12*, 751, 751
\matrix_gadd:Nn . . *4*, *11*, 669, 669, 709
\matrix_gadd:NNN *4*, *12*, 747, 747
\matrix_gadd:NNn *4*, *12*, 709
\matrix_gdiv:Nn . . *4*, *11*, 669, 675, 709
\matrix_gdiv:NNn *4*, *12*, 715
\matrix_gmul:NN *5*, *13*, 798, 798
\matrix_gmul:Nn . . *4*, *11*, 669, 673, 709
\matrix_gmul:NNN . *5*, *13*, 773, 773, 799
\matrix_gmul:NNn *4*, *12*, 713
\matrix_gnegate:N *3*, *11*, 661, 661
\matrix_gnegate:NN . . . *3*, *11*, 701, 701
\matrix_gpower:Nn *4*, *11*, 669, 677, 709
\matrix_gpower:NNn *4*, *12*, 717
\matrix_ground:Nn *4*, *11*, 661, 667
\matrix_ground:NNn . . . *4*, *11*, 701, 707
\matrix_gset:Nn . . *3*, *10*, 420, 420, 432
\matrix_gset_eq:NN *3*, *10*, 698, 698, 700
\matrix_gsign:N *3*, *11*, 661, 665
\matrix_gsign:NN *3*, *11*, 701, 705
\matrix_gsub:NN *5*, *12*, 751, 753
\matrix_gsub:Nn . . *4*, *11*, 669, 671, 709
\matrix_gsub:NNN *4*, *12*, 747, 749
\matrix_gsub:NNn *4*, *12*, 711
\matrix_gzero:N . . *3*, *10*, 408, 408, 419
\matrix_log:N *6*, *14*, 446, 446, 461
\matrix_map_break: *5*, *14*, 513, 528, 611, 611, 612, 614, 620, 627, 635, 642, 649, 659, 685, 695, 729, 733, 744, 782, 786, 794
\matrix_map_break:n . . . *5*, *14*, 611, 613
\matrix_map_function:NN *5*, *13*, 615, 615
\matrix_map_inline:Nn . *5*, *13*, 629, 629
\matrix_multiply:NN *5*, *13*
\matrix_multiply:NNN *5*, *13*

\matrix_new:Nnn *2*, *9*, 356, 356, 378, 462, 463
\matrix_use:N *6*, *14*, 433, 433, 445
\g_tmpa_matrix *6*, *14*, 462
\g_tmpb_matrix *6*, *14*, 462

matrix internal commands:

\g__matrix_a_i_fparray 93
\c__matrix_array_length_int 86, 88, 90, 92, 93, 94, 95, 164
\g__matrix_b_i_fparray 93
\l__matrix_base_int *26*, *27*, 111, 235, 247, 253, 274, 292, 442, 454, 456, 495, 504, 541, 566, 579, 625, 640, 653, 656, 689, 737, 770
\g__matrix_c_i_fparray 93
\l__matrix_col_int 33, 54, 80, 366, 388, 438, 442, 457, 492, 501, 535, 562, 575, 770
\g__matrix_const_seq 32, 403, 410, 422, 510, 525
\c__matrix_count_a_int 87
\c__matrix_count_b_int 87
\c__matrix_count_c_int 87
\g__matrix_d_i_fparray 93
__matrix_data_gcopy:NnNnn 296, 296
__matrix_data_gset:n 277, 289, 400, 429
__matrix_data_gset:Nnnn 277, 277, 291
__matrix_data_gzero: 266, 271, 373, 416
__matrix_data_gzero:Nnn 266, 266, 273
__matrix_data_log:Nnnn 322, 322, 344
__matrix_data_log:Nnnnn 342, 342, 455
__matrix_data_typeset:Nnnn 304, 304, 441
\l__matrix_fparray *26*, *27*, 111, 233, 248, 254, 273, 291, 441, 455, 494, 503, 541, 594, 605, 625, 640, 653, 656, 689, 737, 769
\g__matrix_free_a_seq 105
\g__matrix_free_b_seq 105
\g__matrix_free_c_seq 105
\g__matrix_free_d_seq . . 105, 176, 182
\l__matrix_index_int *24*, *25*, 33, 55, 81, 154, 157, 160, 170, 178, 183, 186, 192, 225, 264, 369, 396, 453, 533, 564, 577
__matrix_inplace_prepare:N 508, 508, 522, 647
__matrix_left_operand_prepare:N 488, 488, 506, 682, 722, 776
\l__matrix_length_int 33, 52, 78, 96, 148, 202, 224, 263, 364, 368, 386, 390, 395, 415, 427, 437, 442, 451, 453, 456, 532, 536, 540, 558, 571, 619, 622, 634, 637, 648, 651, 684, 687, 732, 735, 785
\g__matrix_map_depth_int 552, 556, 570, 585, 587, 590, 593, 601, 604, 606, 609
__matrix_map_inplace:Nn 644, 644, 662, 664, 666, 668, 670, 672, 674, 676, 678
\g__matrix_map_intarray 553, 557, 559, 561, 563, 565, 572, 574, 576, 578, 580
__matrix_map_item:n 582, 582, 591, 603, 633, 639, 650, 655, 686, 691, 734, 739
__matrix_map_operate:NNn 679, 679, 699, 702, 704, 706, 708, 710, 712, 714, 716, 718
__matrix_map_operate:NNNn 719, 719, 748, 750, 752, 754
__matrix_map_pop: 582, 598, 627, 642, 659, 696, 745
__matrix_map_push: 582, 583, 617, 631, 646, 681, 721
__matrix_map_state_restore: 554, 568, 607
__matrix_map_state_save: 554, 554, 595
__matrix_meta_gset:N 37, 49, 375, 402, 538

__matrix_meta_gset:Nnnnn 37, 37, 48, 51
__matrix_meta_parse:N 69, 74, 413, 425, 435, 448, 490, 499, 516, 531, 618, 632
__matrix_meta_parse:NNNNN . . 69, 69
__matrix_multiply_aux:nn 755, 755, 791
\g__matrix_next_a_int 97
\g__matrix_next_b_int 97
\g__matrix_next_c_int 97
\g__matrix_next_d_int . . . 97, 178, 179
\l__matrix_old_index_int 478, 519, 533, 548, 550
\l__matrix_old_length_int 478, 518, 532, 548, 549
\l__matrix_opa_base_int 484, 495, 692, 740, 763
\l__matrix_opa_col_int 478, 492, 683, 726, 731, 758, 763, 779
\l__matrix_opa_fparray 484, 494, 692, 740, 762
\l__matrix_opa_row_int 478, 491, 683, 725, 731, 784, 787
\l__matrix_opb_base_int 484, 504, 741, 766
\l__matrix_opb_col_int 478, 501, 726, 766, 784, 789
\l__matrix_opb_fparray 484, 503, 741, 765
\l__matrix_opb_row_int 478, 500, 725, 779
__matrix_operation_cleanup: 545, 545, 696, 745, 795
__matrix_parse_aux:w . . 57, 57, 72, 77
__matrix_result_prepare:Nnn 523, 523, 544, 683, 731, 784
__matrix_right_operand_prepare:N 488, 497, 507, 723, 777
\l__matrix_row_int . . . 33, 53, 79, 365, 387, 438, 491, 500, 534, 560, 573
\c__matrix_size_a_int 83, 88, 115, 142
\c__matrix_size_b_int 83, 90, 117, 131
\c__matrix_size_c_int 83, 92, 119, 126, 195, 208, 251
__matrix_space_allocate: 189, 200, 371, 398, 537
__matrix_space_allocate:n 189, 189, 202
__matrix_space_allocate_aux:n 150, 150, 173, 197
__matrix_space_allocate_aux_d:n 174, 174, 196
\l__matrix_space_class_tl . . 109, 121, 125, 130, 137, 141, 197, 207, 257
__matrix_space_free: 212, 221
__matrix_space_free:nn 212, 212, 223, 547
__matrix_space_free_aux:nn 204, 204, 211, 218
__matrix_space_query: 242, 260, 372, 399, 416, 429, 440, 493, 502, 517, 539, 621, 636
__matrix_space_query:nn 242, 242, 262, 453
__matrix_space_query_aux:nn 227, 227, 241, 257
__matrix_space_select: . . . 113, 146
__matrix_space_select:n 113, 113, 148, 194, 206, 244
\l__matrix_space_size_int *23*, 109, 122, 126, 131, 138, 142, 275, 293, 457, 540
\l__matrix_space_type_tl *23*
\l__matrix_tmpa_fp . . 7, 757, 760, 771
\l__matrix_tmpa_int 7, 155, 157, 160, 216, 218, 229, 231, 237, 279, 282, 283, 286, 324, 326, 335, 337, 556, 558, 560, 562, 564, 566, 570, 572, 574, 576, 578, 580
\l__matrix_tmpa_tl 7, 169, 170, 182, 183, 306, 310, 314, 315, 317, 318, 320, 325, 329, 334, 336, 339, 345, 348, 352, 353, 354

\l__matrix_tmpb_fp 7
\l__matrix_tmpb_int 7, 159, 161, 163, 217, 218, 230, 232, 234
\l__matrix_tmpb_tl 7
msg commands:
\msg_error:nn 728, 781
\msg_error:nnn 361, 384, 391, 411, 423, 428, 512, 527
\msg_new:nnn 13, 17, 21, 25

N

\NeedsTeXFormat 3
negative-dimension 13

P

prg commands:
\prg_break_point:Nn 627, 642, 659, 695, 744, 794
\prg_map_break:Nn 612, 614
\ProvidesExplPackage 4

R

regex commands:
\regex_replace_once:nnN . . . 317, 353
\right 319, 438

S

scan commands:
\scan_new:N 29, 30, 31
scan internal commands:
\s__matrix 29, 41, 58
\s__matrix_mark 29, 42, 43, 44, 59, 60, 61
\s__matrix_stop 29, 45, 62
seq commands:
\seq_gpop_left:NN 169, 182
\seq_gput_right:Nn 207, 403
\seq_if_empty:NTF 152, 176
\seq_if_in:NnTF . . . 410, 422, 510, 525
\seq_new:N 32, 105, 106, 107, 108

T

tl commands:
\tl_gset:Nn 39
\tl_new:N 11, 12, 109, 363, 393
\tl_put_right:Nn 310, 314, 315, 318, 329, 339, 348, 352
\tl_set:Nn 121, 125, 130, 137, 141, 306, 325, 336, 345
token commands:
\token_to_str:N . . . 411, 423, 512, 527
too-many-elements 13

Z

\Z 317, 353