

linesys

User Manual

Version 1.0

Linear systems and linear expressions from matrix-like input.

Write an augmented matrix to produce a system of linear equations or inequalities, or write a single coefficient row to produce a linear expression. **linesys** keeps both forms close to ordinary matrix notation while providing configurable variables, coefficient formatting, spacing, wrappers, and system-specific features such as delimiters and pivot highlighting.

Package author	Fernando de Lacerda Mortari
Affiliation	Department of Mathematics, Federal University of Santa Catarina (UFSC)
Contact	fernando.mortari@ufsc.br
Environments	<code>linesys</code> , <code>linexp</code>
Inline expression command	<code>\linexpr</code>
Package	<code>linesys</code>
Version	1.0
Engine used for this manual	pdfLaTeX

This manual documents the user-visible behavior of the package as supplied for version 1.0. Examples are compiled with the package itself rather than being simulated illustrations.

Contents

1 Overview	1
2 Installation and loading	1
3 The linesys environment	2
3.1 A first example	2
3.2 Rows and columns	2
4 The linexp environment	2
4.1 Inline form: <code>\linexpr</code>	3
5 Options at a glance	3
6 Alignment: <code>align</code>	4
7 Relation symbols: <code>relation</code> and <code>relations</code>	4
8 Variables	5
8.1 Explicit variable lists	6
8.2 Lowercase Roman variables: <code>roman</code>	6
8.3 Uppercase Roman variables: <code>Roman</code>	6
8.4 Lowercase Greek variables: <code>greek</code>	7
8.5 Indexed variables: templates, starting values, and steps	7
8.5.1 Index progression: <code>varindexstart</code> , <code>varindexstep</code> , and <code>varindexdecrease</code>	8
8.5.2 Choosing the interpretation: <code>varmode</code>	8
9 Lexical coefficient classification	9
9.1 Declaring custom transparent wrappers	10
10 Zero terms and empty cells	10
10.1 Hiding zeros: <code>hidezeros</code>	11
10.2 Empty entries: <code>emptyzeros</code>	11
11 Pivot highlighting	12
11.1 Manual pivot selection: <code>pivotcols</code>	12
11.2 RHS-pivot highlighting	13
11.3 Custom pivot highlighters	13
12 Full mode: <code>full</code>	14
12.1 Leading-negative coefficients	14
12.2 Full mode and pivot parentheses: <code>pivotparen</code>	15
13 Coefficient formatting and scalar multiplication	16
14 Delimiters: <code>delim</code>	17
15 Spacing: <code>rowsep</code> and <code>colsep</code>	18
16 Wrappers	18
16.1 Targets	19
16.2 Row and column filters	20
16.3 Multiple targets	20

16.4 Multiple wrappers and order	21
16.5 Optional arguments in wrappers	21
16.6 Entries versus coefficients and constants	21
16.7 Wrapper scoping	22
16.8 Temporarily disabling wrappers	22
17 Interaction between wrappers and pivots	22
17.1 Built-in pivot highlighter	22
17.2 Custom highlighter precedence	23
18 Persistent defaults	23
18.1 Defaults shared by both renderers	25
19 A compact option reference	26
20 Practical patterns	28
20.1 A textbook-style linear system	28
20.2 A mixed system of linear relations	28
20.3 An explicit algebra display	28
20.4 Selective emphasis	28
20.5 A styled linear combination	29
21 Validation and diagnostics	30
21.1 A note about coefficient interpretation	31
22 Recommended workflow	31
23 Quick reference	32

1 Overview

The **linesys** package provides two closely related environments. The **linesys** environment turns an augmented matrix into a system of linear equations or inequalities, while **linexp** turns one coefficient row into a linear expression. The inline companion `\linexpr` uses the same linear-expression renderer when an environment would be unnecessarily heavy. In all three forms the input is deliberately close to matrix notation.

For a linear system:

```
\begin{linesys}
  1&2&-1&4\\
  0&-3&2&5\\
  2&0&1&-1
\end{linesys}
```

The final column is interpreted as the right-hand side. All preceding columns are coefficient columns. With the default configuration, the result is

$$\begin{array}{rrcrcl} x & + & 2y & - & z & = & 4 \\ & & -3y & + & 2z & = & 5 \\ 2x & & & + & z & = & -1 \end{array}$$

A linear expression uses the same coefficient syntax but has no constant column:

```
\begin{linexp}
  1&2&-3
\end{linexp}
```

$$x + 2y - 3z$$

Environment options configure how a system or expression is presented without altering the supplied coefficient data.

Authorship and AI assistance. The concept, package design, specification, and authorship of **linesys** are by Fernando de Lacerda Mortari, Department of Mathematics, Federal University of Santa Catarina (UFSC). Code implementation, testing, and the design, drafting, and revision of this manual were developed with assistance from OpenAI’s ChatGPT. AI assistance was used as a development and editorial tool; final design decisions, validation, and responsibility for the released package and documentation remain with the author.

2 Installation and loading

Place `linesys.sty` somewhere in the TeX search path, then load it in the document preamble:

```
\usepackage{linesys}
```

The package loads `amsmath`, `xparse`, and `tikz`. The examples in this manual also use additional packages for documentation layout; those are not requirements of **linesys**.

The manual’s helper command `\Scale` is defined by the manual solely to demonstrate that `wrap` can apply arbitrary user-supplied commands. It is **not** a command provided by **linesys**.

3 The linesys environment

The complete environment syntax is:

```
\begin{linesys}[options]
  coefficient & coefficient & ... & constant \\
  coefficient & coefficient & ... & constant \\
  ...
\end{linesys}
```

The optional argument contains a comma-separated list of package options. Rows are separated with `\\`; cells are separated with `&`. A trailing `\\` is harmless: it does not create an additional mathematical row.

3.1 A first example

The minimal form

Code

```
\begin{linesys}
  2&-1&3\\
  0&4&-2
\end{linesys}
```

Result

$$\begin{array}{rcl} 2x & - & y = 3 \\ & & 4y = -2 \end{array}$$

3.2 Rows and columns

For a system with two variables, each row has three entries: two coefficients followed by one constant. For three variables, each row has four entries, and so on.

```
% two variables
\begin{linesys}
  1&2&3\\
  4&5&6
\end{linesys}

% three variables
\begin{linesys}
  1&2&3&4\\
  0&-1&2&5
\end{linesys}
```

Blank rows are ignored when the package counts rows. The first non-empty row is used to determine the matrix width. Every later non-empty row must contain the same number of entries. If a row has a different width, **linesys** reports an error identifying the rendered row number, the expected number of columns, and the number actually found. A trailing row separator creates only a blank row item and does not trigger this diagnostic.

4 The linexp environment

The `linexp` environment typesets one linear expression from one row of coefficients. Every input column is a coefficient column; there is no right-hand-side column. The complete syntax is:

```
\begin{linexp}[options]
  coefficient & coefficient & ... & coefficient
\end{linexp}
```

If the row contains N coefficients, exactly N variables are resolved. The environment accepts exactly one non-empty row. Blank row items are ignored, so a trailing `\` is harmless, but a second non-empty row produces a package error.

A basic linear expression

Code

```
\begin{linexp}
  1&2&3
\end{linexp}
```

Result

$$x + 2y + 3z$$

4.1 Inline form: `\linexpr`

For short expressions, the command form

```
\linexpr[options]{coefficient & coefficient & ... & coefficient}
```

uses the same renderer, options, persistent defaults, wrappers, and diagnostics as `linexp`. It may be used directly in ordinary text or inside an existing math environment. Its coefficient argument obeys the same one-non-empty-row rule as the environment.

An inline linear expression

Code

The expression `\linexpr{1&2&-3}`
can appear directly in a sentence.

Result

The expression $x + 2y - 3z$ can
appear directly in a sentence.

5 Options at a glance

These are the options for the `linesys` environment (rightmost column lists built-in defaults).

Option	Meaning	Default
<code>align</code>	Alignment of expression terms and the right-hand side	<code>r</code>
<code>relation</code>	Uniform relation symbol	<code>=</code>
<code>relations</code>	One relation symbol per non-empty row	<code>none</code>
<code>var</code>	Variable specification	<code>x,y,z,w,t</code>
<code>hidezeros</code>	Hide zero terms	<code>true</code>
<code>emptyzeros</code>	Treat empty cells as zero	<code>false</code>
<code>full</code>	Show all matrix terms	<code>false</code>
<code>pivots</code>	Highlight pivots	<code>false</code>
<code>pivotcols</code>	Choose automatic or manual pivot positions	<code>auto</code>
<code>pivotparen</code>	Parenthesize negative pivots in full mode	<code>true</code>
<code>rhspivots</code>	Highlight eligible RHS pivots	<code>true</code>
<code>scalarops</code>	Show an explicit multiplication symbol	<code>false</code>
<code>scalaropsymbol</code>	Multiplication symbol	<code>\cdot</code>
<code>delim</code>	Left/right delimiters	<code>...</code>
<code>rowsep</code>	Extra vertical row spacing	<code>0pt</code>
<code>colsep</code>	Array column spacing	<code>\arraycolsep</code>
<code>varindex</code>	Placeholder token in indexed templates	<code>i</code>
<code>varindexstart</code>	First index in generated indexed variables	<code>1</code>
<code>varindexstep</code>	Step between generated indices	<code>1</code>
<code>varindexdecrease</code>	Convenience setting for a step of -1	<code>false</code>
<code>varmode</code>	Variable-specification interpretation	<code>auto</code>
<code>wrap</code>	Component formatting rules	<code>none</code>

The `linesys` environment accepts the following subset, with independent persistent defaults:

Option	Meaning	Default
<code>var</code>	Variable specification	<code>x,y,z,w,t</code>
<code>hidezeros</code>	Hide zero terms	<code>true</code>
<code>emptyzeros</code>	Treat empty coefficients as zero	<code>false</code>
<code>full</code>	Show all coefficients explicitly	<code>false</code>
<code>scalarops</code>	Show an explicit multiplication symbol	<code>false</code>
<code>scalaropssymbol</code>	Multiplication symbol	<code>\cdot</code>
<code>colsep</code>	Array column spacing	<code>\arraycolsep</code>
<code>varindex</code>	Placeholder token in indexed templates	<code>i</code>
<code>varindexstart</code>	First index in generated indexed variables	<code>1</code>
<code>varindexstep</code>	Step between generated indices	<code>1</code>
<code>varindexdecrease</code>	Convenience setting for a step of -1	<code>false</code>
<code>varmode</code>	Variable-specification interpretation	<code>auto</code>
<code>wrap</code>	Component formatting rules	<code>none</code>

Boolean options accept the usual `true` and `false` values. For example, `full=true` is valid, as is `hidezeros=false`.

6 Alignment: `align`

`align` controls the alignment of the main mathematical-content columns: the coefficient/variable terms and the right-hand side. The package's internal operation-sign and relation columns remain centered. The option accepts exactly three values: `l` (left), `c` (center), and `r` (right). The default is `r`.

Right alignment (default)

Code

```
\begin{linesys}[align=r]
  1&2&36\\
  12&-4&5\\
  0&7&-2
\end{linesys}
```

Result

$$\begin{array}{rcl} x & + & 2y = 36 \\ 12x & - & 4y = 5 \\ & & 7y = -2 \end{array}$$

Centered and left alignment

Code

```
\begin{linesys}[align=c]
  1&2&36\\
  12&-4&5
\end{linesys}

\begin{linesys}[align=l]
  1&2&36\\
  12&-4&5
\end{linesys}
```

Result

$$\begin{array}{rcl} x & + & 2y = 36 \\ 12x & - & 4y = 5 \end{array}$$

$$\begin{array}{rcl} x & + & 2y = 36 \\ 12x & - & 4y = 5 \end{array}$$

7 Relation symbols: `relation` and `relations`

By default, every row of a `linesys` environment uses the equality symbol `=`. The `relation` option replaces that symbol uniformly, making the same matrix-like input suitable for systems of

inequalities or other linear relations.

A system of linear inequalities

Code

```
\begin{linesys}[relation=\le]
  1&2&5\\
  -1&3&4
\end{linesys}
```

Result

$$\begin{array}{rcrcrcrcl} x & + & 2y & \leq & 5 \\ -x & + & 3y & \leq & 4 \end{array}$$

The relation value is math material rather than a restricted choice. Commands such as `\le`, `\ge`, `\neq`, or another suitable relation symbol may therefore be supplied directly. The built-in default is `relation={=}`.

For mixed systems, `relations` accepts a comma-separated list containing exactly one relation symbol for each rendered non-empty row. Blank row items and a trailing `\\` do not consume list entries.

Different relations by row

Code

```
\begin{linesys}[
  relations={=,\le,\ge}
]
  1&2&3\\
  4&-1&5\\
  0&2&6
\end{linesys}
```

Result

$$\begin{array}{rcrcrcrcl} x & + & 2y & = & 3 \\ 4x & - & y & \leq & 5 \\ & & 2y & \geq & 6 \end{array}$$

The two options use ordinary left-to-right key semantics. Processing `relations={...}` activates per-row relation mode; processing a later `relation=...` clears that list and restores one uniform symbol. Thus

```
relations={\le,=}, relation=\neq
```

uses $\neq$ on every row, while

```
relation=\le, relations={=,\ge}
```

uses the two symbols from the later per-row list. Both options are supported by the persistent-defaults mechanism described in Section 18. When a stored per-row list is active, its length must match the number of non-empty rows in each system that inherits it. A local `relation=...` is a convenient way to return such a system to uniform-relation mode.

Relation symbols are generated output rather than matrix entries. They are not matched by an `entries` wrapper; instead, use the dedicated `relations` wrapper target described in the wrapper section.

8 Variables

The variable options in this section are shared by `linesys` and `linexp`. The `var` option determines the variable sequence. In the default `varmode=auto`, a specification may be

1. an explicit comma-separated list;
2. `roman` for $a, b, c, \dots$;
3. `Roman` for $A, B, C, \dots$;

4. `greek` for the supported lowercase Greek sequence;
5. an indexed token template containing the placeholder selected by `varindex`.

The `varmode` option can also force a specification to be interpreted as an explicit list or as an indexed template when automatic interpretation would be ambiguous.

8.1 Explicit variable lists

Variables are taken from the given list, in the order they were provided.

Explicit variables

Code

```
\begin{linesys}[var={u,v}]
  2&3&7\\
  -1&4&2
\end{linesys}
```

Result

$$\begin{array}{rcl} 2u & + & 3v = 7 \\ -u & + & 4v = 2 \end{array}$$

List can have more variables than needed

Code

```
\begin{linesys}[var={u,v,w,s}]
  1&2&3\\
  4&5&6
\end{linesys}
```

Result

$$\begin{array}{rcl} u & + & 2v = 3 \\ 4u & + & 5v = 6 \end{array}$$

Here only u and v are needed; w and s remain unused. Supplying fewer variables than needed results in an error. The built-in default `var={x,y,z,w,t}` is an explicit five-item list, not an open-ended generator; systems requiring more than five variables therefore need another `var` specification.

8.2 Lowercase Roman variables: `roman`

Automatic $a, b, c, \dots$

Code

```
\begin{linesys}[var=roman]
  1&2&3&4&5\\
  0&1&-1&2&3
\end{linesys}
```

Result

$$\begin{array}{rcl} a & + & 2b + 3c + 4d = 5 \\ & & b - c + 2d = 3 \end{array}$$

The generator provides as many variables as required, up to at most 26 variables.

8.3 Uppercase Roman variables: `Roman`

Automatic $A, B, C, \dots$

Code

```
\begin{linesys}[var=Roman]
  1&2&3&4\\
  0&-1&2&5
\end{linesys}
```

Result

$$\begin{array}{rcl} A & + & 2B + 3C = 4 \\ & & -B + 2C = 5 \end{array}$$

The uppercase generator also provides at most 26 variables.

8.4 Lowercase Greek variables: `greek`

The lowercase Greek generator uses this order:

$$\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \iota, \kappa, \lambda, \mu, \nu, \xi, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi, \omega.$$

It provides at most 23 variables.

Automatic Greek variables

Code

```
\begin{linesys}[var=greek]
  1&2&3&4\\
  0&-1&2&5
\end{linesys}
```

Result

$$\alpha + 2\beta + 3\gamma = 4$$

$$-\beta + 2\gamma = 5$$

8.5 Indexed variables: templates, starting values, and steps

Indexed variables are generated from a token template that includes a placeholder for the index. The token selected by `varindex` acts as the index placeholder; its default is `i`. In the default `varmode=auto`, a single specification containing that token is normally recognized as an indexed template, and every occurrence of the placeholder is replaced by the corresponding braced integer.

Thus the familiar form `x_i` produces $x_1, x_2, x_3, \dots$:

Default indexed variables

Code

```
\begin{linesys}[var=x_i]
  1&2&3&4\\
  0&-1&2&5
\end{linesys}
```

Result

$$x_1 + 2x_2 + 3x_3 = 4$$

$$-x_2 + 2x_3 = 5$$

The index placeholder need not occur immediately after an underscore. It may appear anywhere in the template, including inside grouped subscripts, superscripts, or more elaborate constructions.

General indexed templates

Code

```
\begin{linesys}[var=x_{2,i}]
  1&2&3&4\\
\end{linesys}
```

Result

$$x_{2,1} + 2x_{2,2} + 3x_{2,3} = 4$$

As another example, `var=x^(i)` generates $x^{(1)}, x^{(2)}, x^{(3)}, \dots$. Repeated occurrences of the index placeholder are all replaced, thus `var={\frac {(x-x_0)^{i}}{i!}}` produces

$$\frac{(x-x_0)^1}{1!}, \quad \frac{(x-x_0)^2}{2!}, \quad \frac{(x-x_0)^3}{3!}, \dots$$

when the default start and step are used.

To use another placeholder token, set `varindex`. For example:

Changing the index token

Code

```
\begin{linesys}[
  var=\alpha_j,
  varindex=j
]
1&2&3&4\\
0&-1&2&5
\end{linesys}
```

Result

$$\alpha_1 + 2\alpha_2 + 3\alpha_3 = 4$$

$$-\alpha_2 + 2\alpha_3 = 5$$

8.5.1 Index progression: `varindexstart`, `varindexstep`, and `varindexdecrease`

The first generated index is `varindexstart`; its default is 1. Successive indices differ by `varindexstep`, whose default is also 1. For the variable in position n , the generated index is

$$\text{varindexstart} + (n - 1) \text{varindexstep}.$$

The step may be positive, zero, or negative.

A custom index step

Code

```
\begin{linexp}[
  var=x_i,
  varindexstart=10,
  varindexstep=-2
]
3&4&5
\end{linexp}
```

Result

$$3x_{10} + 4x_8 + 5x_6$$

`varindexdecrease` is a boolean convenience setting for the common unit-step cases: `true` sets the step to -1 , while `false` sets it to 1. Omitting the value is equivalent to `true`. These settings are processed sequentially, so a later `varindexstep` or `varindexdecrease` setting determines the final step.

Generated indices are ordinary integers, so negative values require no special syntax.

8.5.2 Choosing the interpretation: `varmode`

The default `varmode=auto` uses the following interpretation order:

1. the exact values `roman`, `Roman`, and `greek` select their named generators;
2. a specification with more than one top-level comma-separated item is an explicit variable list;
3. a single specification exactly equal to the `varindex` placeholder is an explicit one-variable list;
4. otherwise, if the placeholder occurs anywhere in the specification, it is an indexed template;
5. otherwise the specification is treated as an explicit one-variable list.

TeX grouping therefore distinguishes an explicit list such as `var={i,j,k}` from a one-item template such as `var=x_{2,i}`: the commas in the latter are inside the grouped subscript and do not split the top-level variable list.

An explicit list containing the default placeholder

Code

```
\begin{linesys}[var={i,j,k}]
  1&2&3&4\\
\end{linesys}
```

Result

$$i + 2j + 3k = 4$$

For genuinely ambiguous cases, `varmode=list` and `varmode=indexed` bypass automatic inference and force the desired interpretation. In forced indexed mode, the specification must contain the configured `varindex` placeholder.

Forcing list and indexed interpretations

Code

```
\begin{linexp}[var=i^2,varmode=list]
  3
\end{linexp}

\begin{linexp}[
  var={(x_i,y_i)},
  varmode=indexed
]
  1&2&3
\end{linexp}
```

Result

$$3i^2$$

$$(x_1, y_1) + 2(x_2, y_2) + 3(x_3, y_3)$$

`varindex` selects a TeX token, not a character substring. A control sequence such as `\sin` is therefore not affected merely because its name contains the letter `i`. Literal occurrences of the configured placeholder token elsewhere in an indexed template are replaced; choose a different `varindex` token when such occurrences are meant to remain literal.

9 Lexical coefficient classification

Both environments classify coefficients lexically before deciding whether a term is zero, whether its magnitude is a unit coefficient, and whether it has a leading sign. Classification is performed on a temporary copy; the original coefficient tokens remain the source of the rendered mathematics.

For classification, the package can look through ordinary outer grouping and selected transparent formatting forms, including `\color` declarations and `\textcolor`. Thus harmless formatting does not by itself change whether a coefficient counts as zero, a unit, or a leading-negative coefficient. User-defined one-argument formatting commands can be added to this transparent set with `\linesysdeclaretransparentwrapper`, described below.

A single explicit leading sign is normalized for classification. Consequently, `0`, `+0`, and `-0` all classify as zero, while `1`, `+1`, and `-1` all have unit magnitude. A redundant leading `+` is removed in the rendered coefficient; a leading `-` continues to follow the ordinary sign and full-mode rules.

Formatting does not change zero classification

Code

```
\begin{linesys}[pivots=true]
  \textcolor{red}{0}&{+0}&2&3
\end{linesys}
```

Result

$$\textcircled{2}z = 3$$

The boundary remains deliberately lexical rather than algebraic. An expression such as $1-1$ is not simplified to zero, and arbitrary macros are not expanded in an attempt to discover their mathematical value.

9.1 Declaring custom transparent wrappers

When a document uses its own simple formatting command around coefficients, the command can be declared transparent to the lexical classifier:

```
\newcommand*\Tint}[1]{\textcolor{violet}{#1}}
\linesysdeclaretransparentwrapper{\Tint}
```

The declaration argument must be exactly one control sequence. Once registered, the lexical source form `\Tint{content}` is treated like the package's built-in transparent forms for zero, unit-magnitude, and leading-sign classification. The original wrapper is retained when visible coefficient content is rendered; the declaration changes classification, not presentation. The same declaration applies to `linesys`, `linexp`, and `\linexpr` because all three use the same coefficient classifier.

A user-declared transparent wrapper

Code

```
\newcommand*\Tint}[1]{%
  \textcolor{violet}{#1}}
\begingroup
\linesysdeclaretransparentwrapper{\Tint}
\begin{linesys}[pivots=true]
  \Tint{0}&\Tint{-1}&2
\end{linesys}
\endgroup
```

Result

$$\textcolor{violet}{-1}y = 2$$

Declarations obey ordinary TeX grouping and duplicate declarations are harmless. This makes local classifier extensions straightforward:

```
\begingroup
\linesysdeclaretransparentwrapper{\Tint}
% \Tint{...} is transparent here.
\endgroup
% \Tint{...} is opaque again here.
```

The registered source shape is intentionally narrow. A declaration recognizes only `\foo{content}`: an invocation such as `\foo[option]{content}` or `\foo{content}{extra}` remains opaque. Registered wrappers may be nested; classification peels them recursively while preserving the nested wrappers in visible output.

Declare only commands that are genuinely transparent to the mathematical value of their argument. The command tells **linesys** that it may classify the wrapped content as though the outer command were not present; it does not cause arbitrary macro expansion or algebraic simplification. Built-in handling of `\color` and `\textcolor` remains available independently of this declaration API.

10 Zero terms and empty cells

Both environments support the same `hidezeros` and `emptyzeros` settings.

10.1 Hiding zeros: `hidezeros`

With the default `hidezeros=true`, coefficients classified as zero are omitted. This includes transparent/grouped forms of literal zero and the signed forms `+0` and `-0`. In `linesys`, a row with all zero coefficients and constant k is handled specially and prints $0 = k$. In `linexp`, an all-zero coefficient row prints the lone expression 0 .

Hidden zeros (default)

Code

```
\begin{linesys}
  1&0&-2&5\\
  0&3&0&-1\\
  0&0&0&6\\
  0&0&0&0
\end{linesys}
```

Result

$$\begin{array}{rclcl} x & & - & 2z & = & 5 \\ & 3y & & & = & -1 \\ & & & 0 & = & 6 \\ & & & 0 & = & 0 \end{array}$$

An all-zero linear expression

Code

```
\begin{linexp}
  0&0&0
\end{linexp}
```

Result

$$0$$

Set `hidezeros=false` to retain zero terms:

Visible zeros

Code

```
\begin{linesys}[hidezeros=false]
  1&0&-2&5\\
  0&3&0&-1\\
  0&0&0&6\\
  0&0&0&0
\end{linesys}
```

Result

$$\begin{array}{rclcl} x & + & 0y & - & 2z & = & 5 \\ 0x & + & 3y & + & 0z & = & -1 \\ 0x & + & 0y & + & 0z & = & 6 \\ 0x & + & 0y & + & 0z & = & 0 \end{array}$$

10.2 Empty entries: `emptyzeros`

An empty coefficient gives an error by default; in `linesys`, the same is true of an empty constant. Use `emptyzeros=true` when an empty cell should be interpreted as zero.

Empty cells interpreted as zero

Code

```
\begin{linesys}[emptyzeros=true]
  1&&4\\
  &2&5
\end{linesys}
```

Result

$$\begin{array}{rcl} x & & = & 4 \\ & 2y & = & 5 \end{array}$$

This option changes the interpretation of empty cells; it does not by itself make zero terms visible. Combine it with `hidezeros=false` when desired.

Manual zero, no pivot, and RHS pivot

Code

```
\begin{linesys}[
  pivots=true,
  rhspivots=false,
  pivotcols={2,none,rhs}
]
3&0&4\\
5&6&7\\
0&0&0
\end{linesys}
```

Result

$$\begin{array}{rcl} 3x & + & \textcircled{0}y = 4 \\ 5x & + & 6y = 7 \\ & & 0 = \textcircled{0} \end{array}$$

The `pivots` option still controls only whether and how the selected pivot is highlighted. Manual pivot status therefore remains meaningful with `pivots=false`; for example, it still interacts with `pivotparen` in full mode. Similarly, `rhspivots` governs automatic RHS-pivot highlighting only: an explicit `rhs` selector is not vetoed by `rhspivots=false`.

Use `pivotcols=auto` to restore automatic first-nonzero/RHS detection, including when a surrounding persistent default supplies a manual pivot map.

11.2 RHS-pivot highlighting

The `rhspivots` option controls only whether an eligible RHS pivot is highlighted; it does not change variable-column pivot detection. Its default is `true`.

RHS pivot enabled

Code

```
\begin{linesys}[
  pivots=true,
  rhspivots=true
]
0&0&5\\
1&0&2
\end{linesys}
```

Result

$$\begin{array}{rcl} & 0 & = \textcircled{5} \\ \textcircled{1}x & & = 2 \end{array}$$

RHS pivot disabled

Code

```
\begin{linesys}[
  pivots=true,
  rhspivots=false
]
0&0&5\\
1&0&2
\end{linesys}
```

Result

$$\begin{array}{rcl} & 0 & = 5 \\ \textcircled{1}x & & = 2 \end{array}$$

11.3 Custom pivot highlighters

Instead of `true`, the `pivots` option accepts a command. That command receives the pivot as its single mandatory argument.

For example:

```
\newcommand*\myPivot{1}{\fbox{\ensuremath{#1}}}%

\begin{linesys}[pivots=\myPivot]
  2&1&4\\
  0&-3&5
\end{linesys}
```

Custom highlighter

Code

```
\newcommand*\myPivot{1}{%
  \fbox{\ensuremath{#1}}}%
\begin{linesys}[pivots=\myPivot]
  2&1&4\\
  0&-3&5
\end{linesys}
```

Result

$$\boxed{2}x + \boxed{-3}y = 4$$

$$\boxed{-3}y = 5$$

A custom highlighter is treated as user code and is applied with precedence over coefficient/entry wrappers in the pivot case (more on wrappers below). This makes constructions such as a custom color highlighter useful even when a coefficient wrapper also supplies a different color.

12 Full mode: full

`full=true` is available in both environments and requests a fully explicit coefficient representation. It implies the visible-zero behavior and changes coefficient/sign formatting:

- zero coefficients remain visible;
- coefficients 1 and -1 are not hidden;
- coefficient signs are not split into a separate operation sign and magnitude;
- generated plus signs are retained as explicit term separators, while a redundant leading $+$ supplied as part of a coefficient is normalized away;
- a coefficient classified as leading-negative is parenthesized; `linesys` can alter this only for pivot coefficients through `pivotparen`, described below.

Full mode

Code

```
\begin{linesys}[full=true]
  1&-2&3\\
  0&-1&4
\end{linesys}
```

Result

$$1x + (-2)y = 3$$

$$0x + (-1)y = 4$$

12.1 Leading-negative coefficients

The negative test uses the same lexical classifier described earlier. The package can look through its supported transparent formatting forms, but it does not algebraically simplify an expression. Consequently, expressions such as $-1+\sqrt{2}$ and $-\sqrt{3}$ are treated as leading-negative coefficients. An explicit leading $+$, including one inside transparent formatting, is normalized away rather than printed as part of the coefficient.

Parenthesized negative coefficients in full mode

Code

```
\begin{linesys}[full=true]
  -2&-1+\sqrt{2}&3\\
  -\sqrt{3}&4&5
\end{linesys}
```

Result

$$\begin{array}{rcl} (-2)x & + & (-1 + \sqrt{2})y = 3 \\ (-\sqrt{3})x & + & 4y = 5 \end{array}$$

12.2 Full mode and pivot parentheses: pivotparen

The boolean option `pivotparen` controls whether a leading-negative *pivot coefficient* receives the usual full-mode parentheses. Its default is `true`. The option is relevant only when `full=true`; outside full mode it has no effect.

With the default setting, a negative pivot is parenthesized just like any other negative coefficient:

Default pivot parentheses in full mode

Code

```
\begin{linesys}[full=true]
  -1&2&3
\end{linesys}
```

Result

$$(-1)x + 2y = 3$$

Literal `pivots=true` selects the built-in circular highlighter and, when that key is processed, also sets `pivotparen=false`. This preserves the compact built-in rendering in which the circle itself visually separates a negative pivot from the surrounding row. Other negative coefficients remain parenthesized.

Full mode with built-in pivot highlighting

Code

```
\begin{linesys}[full=true,pivots=true]
  -2&-3&4\\
  0&-4&5
\end{linesys}
```

Result

$$\begin{array}{rcl} \textcircled{-2}x & + & (-3)y = 4 \\ 0x & + & \textcircled{-4}y = 5 \end{array}$$

Option order is significant when `pivots=true` and `pivotparen` are both supplied. A later explicit value overrides the earlier one:

Circle around (-1)

Code

```
\begin{linesys}[
  full=true,
  pivots=true,
  pivotparen=true
]
  -1&2&3
\end{linesys}
```

Result

$$\textcircled{(-1)}x + 2y = 3$$

Circle around -1

Code

```
\begin{linesys}[
  full=true,
  pivotparen=true,
  pivots=true
]
-1&2&3
\end{linesys}
```

Result

$$\textcircled{-1}x + 2y = 3$$

A custom pivot highlighter does not change `pivotparen`; it uses the current state. This is useful for highlighters such as framed boxes, where parentheses may or may not be desirable.

Custom highlighter with pivot parentheses disabled

Code

```
\newcommand*{\myPivot}[1]{\fbox{\ensuremath{\#1}}}
\begin{linesys}[
  full=true,
  pivots=\myPivot,
  pivotparen=false
]
-1&2&3
\end{linesys}
```

Result

$$\boxed{-1}x + 2y = 3$$

The option is independent of highlighting. Thus `full=true,pivots=false,pivotparen=false` leaves a leading-negative row pivot unparenthesized even though no pivot marker is drawn. Non-pivot negative coefficients keep their ordinary full-mode parentheses.

The `linexp` environment has no pivot concept and therefore no `pivotparen` option. In its full mode, every lexically leading-negative coefficient follows the ordinary parenthesizing rule.

Full mode in a linear expression

Code

```
\begin{linexp}[full=true]
-1&-2&3
\end{linexp}
```

Result

$$(-1)x + (-2)y + 3z$$

13 Coefficient formatting and scalar multiplication

In both environments, unit coefficients are suppressed in the ordinary (non-full) representation. Unit magnitude is determined by the shared lexical classifier, so transparent/grouped forms of 1, +1, and -1 behave consistently. Thus $-1y$ is printed as $-y$, while a coefficient such as 2 remains visible.

A leading + is treated as unary coefficient syntax rather than as part of the coefficient's printed content. Transparent formatting is preserved around the remaining magnitude.

Transparent signs and unit coefficients

Code

```
\begin{linexp}[scalarops=true]
  \textcolor{red}{+1}&\{\color{blue}
  \}-1}&+2
\end{linexp}
```

Result

$$1 \cdot x - 1 \cdot y + 2 \cdot z$$

`scalarops=true` changes the ordinary presentation by inserting an explicit multiplication symbol between a coefficient and its variable. The default symbol is $\cdot$ (made with `\cdot`).

Explicit scalar multiplication

Code

```
\begin{linesys}[scalarops=true]
  2&-3&4\\
  -1&1&5
\end{linesys}
```

Result

$$\begin{array}{rcl} 2 \cdot x & - & 3 \cdot y = 4 \\ -1 \cdot x & + & 1 \cdot y = 5 \end{array}$$

With `scalarops=true`, unit magnitudes are not hidden. You can replace $\cdot$ using `scalaropssymbol`.

Custom multiplication symbol

Code

```
\begin{linesys}[
  scalarops=true,
  scalaropssymbol=\ast
]
  2&-3&4\\
  -1&1&5
\end{linesys}
```

Result

$$\begin{array}{rcl} 2 \ast x & - & 3 \ast y = 4 \\ -1 \ast x & + & 1 \ast y = 5 \end{array}$$

In full mode, the explicit full-mode coefficient presentation takes precedence over the ordinary hiding rules; `scalarops` can still insert its multiplication symbol.

14 Delimiters: `delim`

The `delim` option belongs to `linesys`; `linexp` does not add surrounding delimiters. `delim` takes a comma-separated pair:

```
delim={left,right}
```

A dot $.$ means “no visible delimiter” on that side. The default is $.,.$ (no delimiter on either side).

Curly braces on left only

Code

```
\begin{linesys}[delim={\{,}]
  1&2&3\\
  4&5&6
\end{linesys}
```

Result

$$\left\{ \begin{array}{rcl} x & + & 2y = 3 \\ 4x & + & 5y = 6 \end{array} \right.$$

Brackets and a right floor

Code

```
\begin{linesys}[delim={[, \rfloor]}
  1&2&3\\
  4&5&6
\end{linesys}
```

Result

$$\left[\begin{array}{ccc} x & + & 2y = 3 \\ 4x & + & 5y = 6 \end{array} \right]$$

The delimiters are applied with LaTeX's `left` and `right`, so they scale with the height of the system.

15 Spacing: `rowsep` and `colsep`

`rowsep` is a `linesys`-only option. It adds vertical space after each rendered row and is a dimension.

Extra row spacing

Code

```
\begin{linesys}[rowsep=8pt]
  1&2&3\\
  4&5&6\\
  7&8&9
\end{linesys}
```

Result

$$\begin{array}{ccc} x & + & 2y = 3 \\ 4x & + & 5y = 6 \\ 7x & + & 8y = 9 \end{array}$$

`colsep` is shared by both environments and changes the local `arraycolsep` used by the rendered array. Each environment stores its own default, initially taken from `arraycolsep`.

Tighter columns

Code

```
\begin{linesys}[colsep=1pt]
  1&22&333\\
  4&55&666
\end{linesys}
```

Result

$$\begin{array}{ccc} x + 22y = 333 \\ 4x + 55y = 666 \end{array}$$

The spacing changes are local to the current `linesys` or `linexp` array.

16 Wrappers

The wrapper facility is the most flexible formatting feature in **linesys**. Both environments use the same public wrapper grammar and nesting model, while exposing targets and filters appropriate to their output. It lets you apply arbitrary LaTeX commands to selected rendered components.

The public grammar is:

```
wrap={TARGETS}[FILTERS]{WRAPPERS}
```

For example:

```
wrap={coefficients}{\color{blue}}
wrap={variables}[columns=2]{\mathbf}
wrap={entries}[rows=1,columns=2]{\fbox}
```

16.1 Targets

For `linesys`, there are five targets:

entries	matrix entries: coefficients and constants.
coefficients	coefficient components only.
constants	right-hand-side constants only.
variables	variable symbols only.
relations	generated relation symbols only.

The special target **entries** matches both coefficient and constant components, but not variables or relation symbols. In `linexp`, the only valid targets are **coefficients** and **variables**.

Formatting all coefficients

Code

```
\begin{linesys}[
  wrap={coefficients}{\color{blue}}
]
2&-3&4\\
5&1&6
\end{linesys}
```

Result

$$\begin{array}{rcl} 2x & - & 3y = 4 \\ 5x & + & y = 6 \end{array}$$

Formatting variables and constants

Code

```
\begin{linesys}[
  wrap={variables}{\mathbf},
  wrap={constants}{\color{blue}}
]
2&-3&4\\
5&1&6
\end{linesys}
```

Result

$$\begin{array}{rcl} 2\mathbf{x} & - & 3\mathbf{y} = 4 \\ 5\mathbf{x} & + & \mathbf{y} = 6 \end{array}$$

Formatting relation symbols

Code

```
\begin{linesys}[
  relations={=,\le,\neq},
  wrap={relations}{\color{blue}}
]
1&2&3\\
4&5&6\\
7&8&9
\end{linesys}
```

Result

$$\begin{array}{rcl} x & + & 2y = 3 \\ 4x & + & 5y \leq 6 \\ 7x & + & 8y \neq 9 \end{array}$$

The **relations** target is row-level output. Row filters therefore select relation symbols naturally, while a column filter has no coordinate meaning; if supplied, it is ignored with a package warning.

16.2 Row and column filters

Filters are optional. In `linesys`, the canonical filter names are `rows` and `columns`; singular aliases `row` and `column` are also accepted. Selectors are positive integer lists. In `linexp`, only `columns` (or the singular alias `column`) is valid; a row filter is an error because a linear expression contains only one mathematical row.

Each `wrap` declaration has its own filter state: omitting a supported selector means “all” for that declaration, and selectors from another wrapper declaration are not inherited.

```
wrap={coefficients}[rows={1,3}]{\color{red}}
wrap={variables}[columns=2]{\mathbf}
wrap={entries}[rows=1,columns=2]{\fbox}
```

A row selector is based on the rendered, non-empty row number. A column selector refers to matrix columns for `entries` and `coefficients`, and to variable columns for `variables`. Constants and relations are row-level special cases: a column selector supplied with either `constants` or `relations` is ignored and a package warning is issued.

Selectors combine with **AND**: a rule with both a row and column selector applies only where both conditions match. Here’s an example:

```
\begin{linesys}[
  wrap={coefficients}[rows=2,columns={1,3}]{\color{red}},
  wrap={variables}[columns=2]{\mathbf}
]
1&2&3&4\\
5&6&7&8\\
9&10&11&12
\end{linesys}
```

$$\begin{array}{rrrrrr} x & + & 2y & + & 3z & = & 4 \\ \color{red}{5}x & + & 6y & + & \color{red}{7}z & = & 8 \\ 9x & + & 10y & + & 11z & = & 12 \end{array}$$

Only the first and third coefficients in the second row are being colored red.

For `linexp`, column selectors refer directly to coefficient/variable positions in the expression:

```
\begin{linexp}[
  wrap={variables}[columns=2]{\mathbf},
  wrap={coefficients}[columns={1,3}]{\color{red}}
]
2&-3&4
\end{linexp}
```

$$\color{red}{2}x - 3y + \color{red}{4}z$$

16.3 Multiple targets

The target list is comma-separated. One declaration can therefore affect several component types:

```
\begin{linesys}[wrap={coefficients,variables}[rows=1]{\mathbf}]
2&-3&4\\
5&1&6
\end{linesys}
```

$$\begin{array}{rcl} 2\mathbf{x} & - & 3\mathbf{y} = 4 \\ 5x & + & y = 6 \end{array}$$

16.4 Multiple wrappers and order

The wrapper list is also comma-separated. Wrappers are applied in declaration order by nesting each later matching wrapper around the result of the earlier one. Consequently, the visible precedence of stateful declarations such as `\color` follows normal TeX grouping rules rather than a simple “last wrapper wins” rule.

```
\begin{linesys}[wrap={coefficients}[rows=1]{\textbf,\color{magenta}}]
  2&-3&4\\
  5&1&6
\end{linesys}
```

$$\begin{array}{rcl} \textcolor{magenta}{2}x & - & \textcolor{magenta}{3}y = 4 \\ 5x & + & y = 6 \end{array}$$

Multiple wrapper declarations are likewise retained in declaration order. This makes it possible to build a predictable sequence of local formatting operations.

16.5 Optional arguments in wrappers

Wrapper commands may take optional arguments. The wrapper list parser accepts the usual LaTeX form, for example:

```
wrap={coefficients}{\Scale[1.5]}
wrap={coefficients,variables}[rows=1]{\textbf,\Scale[1.2]}
```

The manual’s `\Scale` helper demonstrates this feature:

```
\begin{linesys}[wrap={coefficients}[rows=1]{\Scale[1.35]}]
  2&-3&4\\
  5&1&6
\end{linesys}
```

$$\begin{array}{rcl} 2x & - & 3y = 4 \\ 5x & + & y = 6 \end{array}$$

The wrapper parser splits the wrapper list at commas. TeX grouping protects ordinary mandatory arguments such as `\color{red}`. Optional arguments are accepted in the usual LaTeX form. The package does not claim special parsing for arbitrary comma-containing optional arguments.

16.6 Entries versus coefficients and constants

An `entries` wrapper can style either kind of matrix entry. This is useful when a single rule should cover the complete augmented matrix while leaving variables untouched.

All entries, but not variables

Code

```
\begin{linesys}[
  wrap={entries}{\color{blue}}
]
2&-3&4\\
5&1&6
\end{linesys}
```

Result

$$\begin{array}{rcl} 2x & - & 3y = 4 \\ 5x & + & y = 6 \end{array}$$

16.7 Wrapper scoping

Each wrapper application is local to its target component. Formatting does not leak into neighboring coefficients, variables, or constants.

16.8 Temporarily disabling wrappers

The commands `\linesyswrapdisable` and `\linesyswrapenable` provide a package-wide, group-local switch for the visual application of wrapper rules in both `linesys` and `linexp`. Wrappers are enabled by default. While wrapping is disabled, `wrap` declarations are still parsed and validated, but matching wrapper commands are not applied to the rendered system.

Group-local wrapper suppression

Code

```
\begin{group}
\linesyswrapdisable
\begin{linesys}[
  wrap={coefficients}{\color{red}}
]
2&-1&3\\
4&5&6
\end{linesys}
\end{group}
```

Result

$$\begin{array}{rcl} 2x & - & y = 3 \\ 4x & + & 5y = 6 \end{array}$$

Both commands obey ordinary TeX grouping. A nested `\linesyswrapenable` can therefore restore wrapper rendering temporarily inside a region in which wrappers have been disabled; leaving the nested group restores the surrounding disabled state.

17 Interaction between wrappers and pivots

The package distinguishes the built-in and custom pivot highlighters when combining them with wrappers.

17.1 Built-in pivot highlighter

With `pivots=true`, the built-in circle is applied around the normally wrapped pivot. For example, coefficient wrappers can style the pivot inside the circle.

Built-in pivot plus coefficient wrapper

Code

```
\begin{linesys}[
  pivots=true,
  wrap={coefficients}{\color{blue}}
]
2&-3&4\\
0&5&6
\end{linesys}
```

Result

$$\textcircled{2}x - \textcolor{blue}{3}y = 4$$

$$\textcircled{5}y = 6$$

17.2 Custom highlighter precedence

A custom pivot highlighter is nested inside the ordinary coefficient/entry wrappers rather than outside them. This nesting allows an inner stateful declaration, such as a custom pivot color, to remain effective against a conflicting outer wrapper; for example, a blue pivot can override a red coefficient wrapper.

Custom pivot color overrides coefficient color

Code

```
\begin{linesys}[
  pivots=\color{blue},
  wrap={coefficients}{\color{red}}
]
2&-3&4\\
0&5&6
\end{linesys}
```

Result

$$\textcolor{blue}{2}x - \textcolor{red}{3}y = 4$$

$$\textcolor{blue}{5}y = 6$$

The same precedence rule applies to a custom highlighter used for an RHS pivot.

The package also defines `\linesyscircled` as the command used by the built-in pivot highlighter. It takes one argument and places that argument inside a TikZ circle; this command is available independently of the environment.

```
\[
\linesyscircled{p}
\qqquad
\linesyscircled{-2}
\]
```

$$\textcircled{p} \quad \textcircled{-2}$$

18 Persistent defaults

Use `linesyssetdefaultoptions` to change the defaults used by subsequent `linesys` environments:

```
\linesyssetdefaultoptions{
  var={x,y},
  hidezeros=false,
  rowsep=4pt
}
```

Only the supplied settings change. Later environments inherit those values, while an option in an individual environment still takes precedence. Relation settings are persistent as well: `relation` stores one uniform symbol and clears any stored per-row list, while `relations` stores one symbol per rendered row. Their left-to-right override behavior is the same in the `defaults` command as in an individual environment. The `pivotcols` option is defaultable as well: a manual selector list can be stored persistently, while a local `pivotcols=auto` restores automatic pivot detection for one environment. The `pivotparen` option is part of this defaults system and follows the same sequential-key behavior as in an individual environment: literal `pivots=true` sets it to `false` when processed, while a later `pivotparen=true` can set it back to `true`. The indexed-variable settings are persistent in the same way: `varindexstep` stores the numeric step, while `varindexdecrease` is a convenience setting that writes either `-1` or `1` to that step. The `varmode` interpretation mode is also defaultable.

Wrapper declarations are intentionally environment-local: `wrap` is not a supported key of `\linesyssetdefaultoptions` and cannot be stored as a persistent default. The separate `\linesyswrapdisable` and `\linesyswrapenable` commands are likewise group-local switches, not stored default keys.

Persistent defaults

Code

```
\begingroup
\linesyssetdefaultoptions{
  var={u,v},
  hidezeros=false
}
\begin{linesys}
  1&0&3\\
  0&2&4
\end{linesys}
\endgroup
```

Result

$$\begin{array}{rcl} u & + & 0v = 3 \\ 0u & + & 2v = 4 \end{array}$$

The command is local to the surrounding TeX group. In particular,

```
\begingroup
  \linesyssetdefaultoptions{...}
  % changed defaults here
\endgroup
% original defaults are restored here
```

is a convenient pattern for temporary document-wide configuration.

The `linexp` environment has a separate persistent-default command:

```
\linexpsetdefaultoptions{
  var=x_i,
  varindexstart=2,
  varindexstep=3,
  scalarops=true
}
```

Its supported keys are exactly the non-wrapper `linexp` options listed above, including all variable-generation controls: `varindex`, `varindexstart`, `varindexstep`, `varindexdecrease`, and `varmode`. The two defaults systems are independent: changing `linesys` defaults does not change `linexp` defaults, and vice versa. As with `linesyssetdefaultoptions`, `wrap` is intentionally not a persistent default.

18.1 Defaults shared by both renderers

When the same persistent configuration should apply to both `linesys` and `linexp` (and therefore also to `\linexpr`), use `\linesyssetcommonoptions` :

```
\linesyssetcommonoptions{
  var=x_i,
  varindexstart=0,
  varindexstep=2,
  hidezeros=false,
  scalarops=true
}
```

The command updates only options that exist with the same meaning in both renderers. The supported keys are `var`, `hidezeros`, `emptyzeros`, `full`, `scalarops`, `scalaropssymbol`, `colsep`, `varindex`, `varindexstart`, `varindexstep`, `varindexdecrease`, and `varmode`. System-specific settings such as relations, pivots, delimiters, and row spacing are untouched. Wrapper declarations remain environment-local and are not accepted here.

The command writes to the two established persistent-default families rather than creating a third layer of runtime options. Consequently, later calls to `linesyssetdefaultoptions` or `linexpsetdefaultoptions` can still override one renderer independently, and individual environment or `\linexpr` options still take final precedence. Like the other defaults commands, `\linesyssetcommonoptions` is local to the surrounding TeX group.

Shared persistent defaults

Code

```
\begingroup
\linesyssetcommonoptions{
  var=x_i,
  varindexstart=0,
  hidezeros=false
}
\begin{linesys}
  1&0&5
\end{linesys}
\;
\linexpr{1&0}
\endgroup
```

Result

$$x_0 + 0x_1 = 5 \quad x_0 + 0x_1$$

19 A compact option reference

This section collects the exact user-facing forms in one place.

Core presentation

<code>align=l c r</code>	<code>linesys</code> only: alignment of expression terms and the right-hand side. Default <code>r</code> .
<code>hidezeros=true false</code>	Hide zero coefficients. Default <code>true</code> .
<code>emptyzeros=true false</code>	Treat empty cells as zero. Default <code>false</code> .
<code>full=true false</code>	Fully explicit coefficient presentation. Default <code>false</code> .
<code>pivotparen=true false</code>	<code>linesys</code> only: in full mode, control parentheses around leading-negative pivot coefficients. Default <code>true</code> ; literal <code>pivots=true</code> sets it to <code>false</code> when processed.
<code>scalarops=true false</code>	Insert multiplication symbols. Default <code>false</code> .
<code>scalaropsymbol=<code></code>	Symbol inserted by <code>scalarops</code> . Default <code>\cdot</code> .

Relations (`linesys` only)

<code>relation=<code></code>	Use one relation symbol for every rendered row. Default <code>=</code> . Processing this key clears any active per-row <code>relations</code> list.
<code>relations=<list></code>	Supply exactly one relation symbol for each non-empty row. A later <code>relation</code> restores uniform-relation mode.

Variables

<code>var=<spec></code>	Explicit list, <code>roman</code> , <code>Roman</code> , <code>greek</code> , or an indexed token template.
<code>varindex=<token></code>	Placeholder token for indexed templates. Default <code>i</code> .
<code>varindexstart=<integer></code>	First generated index. Default <code>1</code> .
<code>varindexstep=<integer></code>	Difference between successive generated indices. Default <code>1</code> .
<code>varindexdecrease</code>	Boolean convenience setting: <code>true</code> sets the index step to <code>-1</code> , while <code>false</code> sets it to <code>1</code> .
<code>varmode</code>	Values <code>auto</code> , <code>list</code> , or <code>indexed</code> ; choose automatic, forced-list, or forced-indexed interpretation of <code>var</code> . Default <code>auto</code> .

Pivots (`linesys` only)

<code>pivots=true</code>	Built-in circular highlighter.
<code>pivots=false</code>	Disable pivot highlighting.
<code>pivots=<command></code>	Custom highlighter receiving one argument.
<code>pivotcols=auto</code>	Use automatic first-nonzero/RHS pivot detection. Default <code>auto</code> .
<code>pivotcols=<selectors></code>	Manually select one pivot state per non-empty row; each selector is a positive variable-column number, <code>rhs</code> , or <code>none</code> .

rhspivots=true|false Highlight an eligible automatically detected nonzero RHS pivot when no variable pivot occurs. Default **true**; an explicit manual **rhs** selection is not vetoed by this option.

Geometry

delim=<left,right> **linesys** only: scalable delimiters; **.** means invisible. Default **.,..**

rowsep=<dimension> **linesys** only: extra row spacing. Default **0pt**.

colsep=<dimension> Local array column spacing. Default **\arraycolsep** .

Wrappers

wrap Form: **wrap={TARGETS}[FILTERS]{WRAPPERS}**. Apply formatting commands to selected components. **linesys** supports the five targets documented in the Wrappers section; **linexp** supports only **coefficients** and **variables**. Column filters are ignored for the row-level **constants** and **relations** targets. The option is environment-local and is not accepted by any of the persistent-default commands.

\linesyswrapdisable Suppress visual wrapper application within the surrounding TeX group; wrapper syntax is still parsed and validated.

\linesyswrapenable Re-enable visual wrapper application within the surrounding TeX group.

Commands and persistent defaults

\linexpr [options]{...} Inline companion to **linexp**; accepts the same coefficient row, options, defaults, wrappers, and diagnostics.

\linesyssetdefaultoptions {...} Set persistent defaults for **linesys**.

\linexpsetdefaultoptions {...} Set persistent defaults for **linexp** and **\linexpr** .

\linesyssetcommonoptions {...} Set only the persistent options shared by both renderers.

\linesysdeclaretransparentwrapper {\foo } Declare a one-argument source form **\foo{content}** transparent to lexical coefficient classification. Declarations are TeX-group-local and duplicate registrations are harmless.

Environment-specific options

linesys Additionally supports the relation and pivot controls described above, together with **align**, **delim**, and **rowsep**.

linexp Supports only the shared coefficient/variable options listed above; it has no RHS, pivot, delimiter, row-spacing, or alignment options.

20 Practical patterns

The following combinations cover common presentation tasks.

20.1 A textbook-style linear system

```
\begin{linesys}[
  var=x_i,
  pivots=true,
  delim={\{,.\},
  rowsep=3pt
]
1&2&-1&4\\
0&3&2&7\\
0&0&5&-2
\end{linesys}
```

$$\left\{ \begin{array}{rclcl} \textcircled{1}x_1 & + & 2x_2 & - & x_3 & = & 4 \\ & & \textcircled{3}x_2 & + & 2x_3 & = & 7 \\ & & & & \textcircled{5}x_3 & = & -2 \end{array} \right.$$

20.2 A mixed system of linear relations

```
\begin{linesys}[
  relations={=,\le,\ge},
  delim={\{,.\}
]
1&2&5\\
3&-1&4\\
0&2&6
\end{linesys}
```

$$\left\{ \begin{array}{rcl} x & + & 2y = 5 \\ 3x & - & y \leq 4 \\ & & 2y \geq 6 \end{array} \right.$$

20.3 An explicit algebra display

```
\begin{linesys}[
  full=true,
  scalarops=true
]
-2&3&5\\
0&-1&4
\end{linesys}
```

$$\begin{array}{rcl} (-2) \cdot x & + & 3 \cdot y = 5 \\ 0 \cdot x & + & (-1) \cdot y = 4 \end{array}$$

20.4 Selective emphasis

```
\begin{linesys}[
  pivots=true,
  wrap={variables}[columns=2]{\mathbf},
  wrap={constants}[rows={1,3}]{\color{blue}},
]
```

```
wrap={coefficients}[rows=2]{\color{red}}
]
2&1&0&4\\
0&-3&2&5\\
0&0&7&-1
\end{linesys}
```

$$\begin{array}{rcl} 2x & + & y = 4 \\ -3y & + & 2z = 5 \\ & & 7z = -1 \end{array}$$

20.5 A styled linear combination

```
\begin{linexp}[
  full=true,
  scalarops=true,
  scalaropssymbol=\ast,
  var=\vec{v}_j,
  varindex=j,
  varindexstart=2,
  wrap={coefficients}[columns={1,3}]{\color{blue}}
]
1&2&3
\end{linexp}
```

$$1 * \vec{v}_2 + 2 * \vec{v}_3 + 3 * \vec{v}_4$$

21 Validation and diagnostics

The package performs several structural checks before opening its internal array. The important user-facing cases are listed below.

Situation	Package response
One-column or empty-width matrix	Error: the system must have at least one variable column.
Inconsistent non-empty row width	Error identifying the rendered row number, expected number of columns, and number found.
Manual <code>pivotcols</code> selector count does not match the non-empty row count	Error identifying both counts.
Invalid <code>pivotcols</code> selector	Error identifying the selector and row; use a positive variable-column number, <code>rhs</code> , or <code>none</code> .
Manual pivot column exceeds the number of variable columns	Error identifying the invalid column, row, and column limit.
Too few explicit variables in <code>linesys</code>	Error identifying the required and supplied counts.
Forced indexed mode without its placeholder (either environment)	Error identifying the missing <code>varindex</code> token.
More than 26 Roman variables requested	Error.
More than 26 uppercase Roman variables requested	Error.
More than 23 Greek variables requested	Error.
Empty coefficient/constant with <code>emptyzeros=false</code>	Error.
Invalid transparent-wrapper declaration	Error requiring exactly one control sequence.
Invalid <code>linesys</code> wrapper target	Error listing the five valid targets.
Per-row <code>relations</code> count does not match the non-empty row count	Error identifying both counts.
Empty uniform or per-row relation symbol	Error requiring math material such as <code>=</code> , <code>\le</code> , or <code>\ge</code> .
Invalid <code>linesys</code> wrapper row/column selector	Error identifying the invalid index and the system limit.
Column selector on <code>constants</code> or <code>relations</code> in <code>linesys</code>	Warning; selector is ignored.
Malformed <code>linesys</code> wrapper syntax	Error with the canonical wrapper grammar.
<code>linexp</code> or <code>\linexpr</code> with zero or multiple non-empty rows	Error requiring exactly one non-empty coefficient row.
Too few explicit variables in <code>linexp</code> or <code>\linexpr</code>	Error identifying the required and supplied counts.
Empty <code>linexp</code> coefficient with <code>emptyzeros=false</code>	Error.
Invalid <code>linexp</code> wrapper target	Error; valid targets are <code>coefficients</code> and <code>variables</code> .
Row filter in a <code>linexp</code> wrapper	Error; only column filters are supported.
Invalid <code>linexp</code> wrapper column selector	Error identifying the invalid index and expression limit.
Malformed <code>linexp</code> wrapper syntax	Error with the <code>linexp</code> wrapper grammar.

The package reports these conditions through its own diagnostics rather than intentionally continuing into an incomplete array. This is especially useful for invalid variable specifications and wrapper selectors. The inline `\linexpr` command delegates to `linexp`, so it reports the same expression-level diagnostics rather than maintaining a separate validation path.

21.1 A note about coefficient interpretation

Coefficient handling is lexical, not algebraic. Zero, unit-magnitude, and leading-sign tests share the transparent classification rules described earlier, so outer grouping, supported built-in formatting, and registered one-argument transparent wrappers do not change those basic lexical properties. A single leading $+$ or $-$ is ignored when testing literal zero or unit magnitude; the sign itself is still handled by the normal rendering rules.

The package does not evaluate mathematical expressions or expand arbitrary macros to discover their value. In particular, $1-1$ is not classified as zero merely because it is mathematically equal to zero. The original source formatting is preserved for rendered coefficient content wherever that content remains visible.

22 Recommended workflow

For most documents, the following progression is enough:

1. Choose `linesys` for an augmented-matrix system, `linexp` for one displayed linear expression, or `\linexpr` when the same expression renderer is more convenient inline; let the default variables x, y, z, w, t do the work.
2. Set `var` when the system or expression uses different variable names.
3. In `linesys`, set `relation` for a uniform inequality or other relation, or `relations` when the rows use different symbols.
4. Use `hidezeros=false` or `full=true` when a more explicit coefficient presentation is needed.
5. In `linesys`, use `pivots=true` when pivot positions should be visible; use `pivotcols` when the pivot positions should be chosen manually rather than detected automatically. In full mode, use `pivotparen` when you want explicit control over parentheses around negative pivots.
6. In automatic pivot mode, add `rhspivots=false` if eligible RHS pivots should not be highlighted.
7. If custom one-argument formatting commands should be transparent to coefficient classification, register them with `\linesysdeclaretransparentwrapper` in the scope where that behavior is wanted.
8. Use `wrap` for selective emphasis rather than manually rebuilding rendered rows.
9. If the same supported non-wrapper configuration is repeated throughout a document, use the persistent-defaults mechanism described in Section 18. Use `\linesyssetcommonoptions` when the shared settings should apply to both system and expression rendering, and the environment-specific defaults commands when they should diverge.

A useful mental model: the matrix body supplies the data; the options choose how that data is interpreted and displayed; wrappers provide a final, local formatting layer. Keeping those roles separate makes complicated examples much easier to maintain.

23 Quick reference

Task	Use
Load package	<code>\usepackage \{linesys\}</code>
Basic system	<code>\begin {linesys} ... \end {linesys}</code>
Basic linear expression	<code>\begin {linexp} ... \end {linexp}</code>
Inline linear expression	<code>\linexpr [options]\{...\}</code>
Choose variables	<code>var=...</code>
Choose variable interpretation	<code>varmode=auto list indexed</code>
Choose index step	<code>varindexstep=...</code>
Decrease generated indices by one	<code>varindexdecrease=true</code>
Hide/show zeros	<code>hidezeros=true/false</code>
Treat blanks as zeros	<code>emptyzeros=true</code>
Show full coefficients	<code>full=true</code>
Choose one relation for all rows	<code>relation=\le</code>
Choose relations row by row	<code>relations={=,\le,\ge}</code>
Highlight pivots	<code>pivots=true</code>
Custom pivot command	<code>pivots=<command></code>
Choose manual pivots	<code>pivotcols={1,2,rhs,none,...}</code>
Return to automatic pivots	<code>pivotcols=auto</code>
Disable automatic RHS-pivot highlighting	<code>rhspivots=false</code>
Explicit multiplication	<code>scalarops=true</code>
Choose multiplication symbol	<code>scalaropsymbol=...</code>
Choose delimiters	<code>delim={left,right}</code>
Vertical spacing	<code>rowsep=...</code>
Horizontal spacing	<code>colsep=...</code>
Selective formatting	<code>wrap={targets}[filters]{wrappers}</code>
Temporarily disable/enable wrappers	<code>\linesyswrapdisable / \linesyswrapenable</code>
Declare transparent wrapper	<code>\linesysdeclaretransparentwrapper \{foo \}</code>
Persistent system configuration	<code>\linesyssetdefaultoptions \{...\}</code>
Persistent expression configuration	<code>\linexpsetdefaultoptions \{...\}</code>
Persistent configuration shared by both	<code>\linesyssetcommonoptions \{...\}</code>