Subfigure, amsmath, and new environment in LaTeX

Prof. Pranay Kumar Saha

March 29, 2025

Including Figures & Subfigures

Use `graphicx` for `\includegraphics` and `subcaption` for subfigures.

```
\begin{figure}[htbp] % Float placement suggestion
  \centering % Center contents
  \begin{subfigure}[b]{0.48\textwidth} % Subfigure 1
      \includegraphics[width=\textwidth]{example-image-a}
      \caption{Caption for Subfigure A}
      \label{fig:sub_a_example}
  \end{subfigure}
  \hfill % Space between subfigures
  \begin{subfigure}[b]{0.48\textwidth} % Subfigure 2
      \includegraphics[width=\textwidth]{example-image-b}
      \caption{Caption for Subfigure B}
      \label{fig:sub_b_example}
  \end{subfigure}
  \caption{Main caption for the whole figure} % Goes in LoF
  \label{fig:main_example} % Label for the whole figure
\end{figure}
```

Including Figures & Subfigures

- `\begin{subfigure}[pos]{width}`: Creates a subfigure box.
- `\caption` inside `subfigure`: Captions the sub-part (e.g., (a), (b)).
- Main `\caption`: Appears in the List of Figures.
- Use separate `\labels` for main figure and subfigures.

`amsmath`: The Foundation

Essential for Serious Math Typesetting

While basic LaTeX provides the `equation` environment, the `amsmath` package offers much more flexibility and power for complex mathematical expressions.

Key Benefits

- Environments for multi-line equations (aligned, grouped).
- Environments for matrices and piecewise functions.
- Commands for text within math mode.
- Enhanced control over equation numbering.
- Improved spacing and general appearance of math.

Many other math packages (like `amsthm`) rely on `amsmath`.

Inline vs. Display Mathematics

Integrating Math with Text

LaTeX distinguishes between math that flows within text (inline) and math shown on its own lines (display).

Inline Math: $ ... $

Use single dollar signs `$...$` to embed math directly within a paragraph. It uses a more compact style (e.g., limits on sums/integrals appear as subscripts/superscripts).

```
The function $f(x) = x^2 - 2x + 1$ can be simplified.
Consider the sum $\sum_{i=1}^n i = \frac{n(n+1)}{2}$.
```

Output Appearance: The function f(x) = $f(x) = x^2 - 2x + 1$ can be simplified. Consider the sum $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ (Note: Symbols appear smaller and integrated into the line).

Inline vs. Display Mathematics

Integrating Math with Text

Display Math: \[... \] or Environments

Use `\[ ... \]` for unnumbered displayed equations or environments like `equation`, `align`, `gather` (often from `amsmath`) for numbered/aligned display math. These appear centered on separate lines with more generous spacing.

```
The function can be simplified:
\[ f(x) = x^2 - 2x + 1 \]
The sum is:
\begin{equation} \label{eq:sum_n}
  \sum_{i=1}^{n_i} = \frac{n(n+1)}{2}
\end{equation}
See \cref{eq:sum_n}.
```

Inline vs. Display Mathematics

The function can be simplified:

$$f(x) = x^2 - 2x + 1$$

The sum is:

$$\sum_{i=1}^{n_i} = \frac{n(n+1)}{2} \tag{1}$$

`\cref` See eq. (1). `\ref` see 1

Output Appearance: The equations appear centered on new lines, visually separate from the text. Limits on the sum appear above/below. The ‘equation‘ environment adds a number (e.g., (1)).

Aligned Equations: `align` and `align*`

Aligning Multiple **Display** Equations

Use `align` for numbered equations or `align*` for unnumbered ones. Use `&` to mark the alignment point and `\\` to separate lines. **These are display environments.**

Code (`align`)

```
\begin{align}
  f(x) &= (x+a)(x+b) \label{eq:expand} \\
       &= x^2 + (a+b)x + ab \\ % No label needed here
  g(x) &= (x+c)^2 \label{eq:square} \\
       &= x^2 + 2cx + c^2
\end{align}
Referencing \cref{eq:expand} and \cref{eq:square}.
Inline math like $a=1$ can be used in surrounding text.
```

Aligned Equations: `align` and `align*`

Aligning Multiple **Display** Equations

Expected Output

$$
\begin{aligned}
f(x) &= (x+a)(x+b) && (2)\\
&= x^2 + (a+b)x + ab && (3)\\
g(x) &= (x+c)^2 && (4)\\
&= x^2 + 2cx + c^2 && (5)
\end{aligned}
$$

Referencing eq. (2) and eq. (4). Inline math like $a = 1$ can be used in surrounding text.

Grouped Equations: `gather` and `gather*`

Centering Multiple **Display** Equations

Use `gather` (numbered) or `gather*` (unnumbered) when you have multiple equations to display together, but they don't need alignment relative to each other. Each equation is centered independently. **These are display environments.**

Code (`gather`)

```
Pythagoras: $a^2 + b^2 = c^2$. Euler's identity is amazing:
\begin{gather}
  a^2 + b^2 = c^2 \label{eq:pythagoras} \\
  e^{i\pi} + 1 = 0 \label{eq:euler}
\end{gather}
See \cref{eq:pythagoras,eq:euler}.
```

Grouped Equations: `gather` and `gather*`

Centering Multiple **Display** Equations

Expected Output

Pythagoras: $a^2 + b^2 = c^2$. Euler's identity is amazing:

$$a^2 + b^2 = c^2 \tag{6}$$

$$e^{i\pi} + 1 = 0 \tag{7}$$

See eqs. (6) and (7).

Long Single Equations: `multline` and `multline*`

Breaking a Single **Display** Equation

Use `multline` (numbered) or `multline*` (unnumbered) for a *single* equation that is too long for one line. **This is a display environment.**

Code (`multline`)

```
A very long sum starting with $x = a+b+c+ \dots$
\begin{multline} \label{eq:long_sum}
  x = a + b + c + d + e + f + g + h + i \\
    + j + k + l + m + n + o + p + q \\
    + r + s + t + u + v + w + y + z
\end{multline}
Check \cref{eq:long_sum}.
```

Long Single Equations: `multline` and `multline*`

Breaking a Single **Display** Equation

- The text starts with an inline math snippet '$x = a + b + c + \ldots$'.
- The 'multline' equation is split across three lines, formatted as described before (left, center, right alignment).
- A single equation number appears.

Expected Output

A very long sum starting with $x = a + b + c + \ldots$

$$x = a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u + v + w + y + z \quad (8)$$

Check eq. (8).

Splitting Within Environments: `split`

Aligning Parts of a Single **Display** Equation

The `split` environment is used *inside* another display math environment (like `equation`, `align`, `gather`) to break a single logical equation with alignment points (`&`). It uses the number of the surrounding **display environment**.

Code (`split` inside `equation`)

```
The Hamiltonian $H_c$ is given by:
\begin{equation} \label{eq:split_example}
\begin{split}
  H_c &= \frac{1}{2n} \sum_{l=0}^{n-1} \left( l^2 + l + \frac{1}{3} \right) \
      &\quad + \frac{1}{2n} \sum_{l=0}^{n-1} \left( \dots \right) \\
      &= \text{Something simpler}
\end{split}
\end{equation} See \cref{eq:split_example}.
```

Splitting Within Environments: `split`

Aligning Parts of a Single **Display** Equation

- Text uses inline math 'H_c'.
- The multi-line formula appears as a single display block, aligned as specified.
- The entire block receives only one equation number.

Expected Output

The Hamiltonian H_c is given by:

$$\begin{aligned} H_c &= \frac{1}{2n} \sum_{l=0}^{n-1} \left(l^2 + l + \frac{1}{3} \right) \\ &\quad + \frac{1}{2n} \sum_{l=0}^{n-1} (\dots) \\ &= \text{Something simpler} \end{aligned} \tag{9}$$

Matrices: `pmatrix`, `bmatrix`, etc.

Typesetting Arrays **within Math Mode**

`amsmath` provides environments for common matrix delimiters. Use `&` to separate columns and `\\` to end rows. These are used *inside* math mode (e.g., within inline '$A = \ldots$' or display `\[ ... \]` or `equation`).

Code (Display Math Example)

```
% Displayed Matrices
\[
  \mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}
  \quad % Some space
  \mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]
% Inline Matrix Example
The matrix $M = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ is invertible.
```

Matrices: `pmatrix`, `bmatrix`, etc.

Typesetting Arrays **within Math Mode**

Expected Output

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The matrix $M = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ is invertible.

Piecewise Functions: `cases`

Defining Functions by Cases **within Math Mode**

The `cases` environment creates a structure with a large left brace. Use `&` to separate the function value from the condition. Use inside display math (`\[...\]`, `equation`) or inline (`$...$` - though less common for multi-line cases).

```
The absolute value $|x|$ is defined as:
\[ % Display mode is common for cases
  |x| =
  \begin{cases}
    x,  & \text{if } x \ge 0 \\
    -x, & \text{if } x < 0
  \end{cases}
\]
The function $f(x)$ (see \cref{eq:piecewise_f}) is continuous.
```

```
\begin{equation} \label{eq:piecewise_f}
f(x) =
  \begin{cases}
    \sin(x)/x, & x \neq 0 \\
    1,         & x = 0
  \end{cases}
\end{equation}
```

Piecewise Functions: cases

Expected Output

The absolute value $|x|$ is defined as:

$$|x| = \begin{cases} x, & \text{if } x \geq 0 \\ -x, & \text{if } x < 0 \end{cases}$$

The function $f(x)$ (see eq. (10)) is continuous.

$$f(x) = \begin{cases} \sin(x)/x, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{10}$$

Controlling Numbering: `\nonumber`

Suppressing Equation Numbers for Specific **Display** Lines

Within environments like `align` or `gather`, you can prevent a specific line from being numbered using `\nonumber` (or its synonym `\notag`). **This applies to lines in numbered display environments.**

```
\begin{align}
  y &= mx + c \\
  E &= mc^2 \nonumber \\ % This line won't be numbered
  F &= ma
\end{align}
```

Controlling Numbering: `\nonumber`

Suppressing Equation Numbers for Specific **Display** Lines

Expected Output

$$y = mx + c \tag{11}$$
$$E = mc^2$$
$$F = ma \tag{12}$$

Special Symbols: amssymb

Blackboard Bold, Calligraphic, and More **in Math Mode**

The amssymb package (which requires amsfonts) provides many common mathematical symbols for use **inside inline (‘...‘) or display math**.

Common Commands

- Blackboard Bold: `\mathbb{R}` ($\mathbb{R}$), `\mathbb{C}` ($\mathbb{C}$), `\mathbb{N}` ($\mathbb{N}$), `\mathbb{Z}` ($\mathbb{Z}$)
- Calligraphic: `\mathcal{F}` ($\mathcal{F}$), `\mathcal{L}` ($\mathcal{L}$)
- Fraktur: `\mathfrak{g}` ($\mathfrak{g}$), `\mathfrak{H}` ($\mathfrak{H}$)
- Operators/Relations: `\lesssim` ($\lesssim$), `\therefore` ($\therefore$)

Example: The set of real numbers $\mathbb{R}$ is uncountable.

Special Symbols: `amssymb`

Blackboard Bold, Calligraphic, and More **in Math Mode**

Finding Symbols

- Use online tools like Detexify (draw the symbol).
- Consult the "Comprehensive LaTeX Symbol List" document (`texdoc symbols-a4` on command line).

Structuring Proofs: amsthm

Definitions, Theorems, Lemmas, Proofs

The amsthm package provides tools for creating structured, numbered (or unnumbered) theorem-like environments.

Setup (in Preamble)

1. Load the package: `\usepackage{amsthm}`
2. Define theorem styles (optional): `\theoremstyle{style_name}` (‘plain‘, ‘definition‘, ‘remark‘)
3. Define environments: `\newtheorem{env_name}{Title}[counter]` or `\newtheorem{env_name}[shared_counter]{Title}`

Structuring Proofs: amsthm

Definitions, Theorems, Lemmas, Proofs

Example Definitions (Preamble Code)

```
% Load package
\usepackage{amsthm}

% Define styles and environments
\theoremstyle{plain}
\newtheorem{theorem}{Theorem}[section] % Numbers like 1.1, 1.2 in Sec 1
\newtheorem{lemma}[theorem]{Lemma}     % Shares counter with theorem

\theoremstyle{definition}
\newtheorem{definition}[theorem]{Definition}

\theoremstyle{remark}
\newtheorem{remark}{Remark} % Separate counter, not reset by section
```

Using Theorem Environments

Writing Theorems, Lemmas, Proofs (with Inline Math)

Once defined in the preamble, use the environments like any other LaTeX environment. Mathematical expressions within the text of theorems/proofs use inline math ‘...‘. The ‘proof‘ environment is predefined by ‘amsthm‘.

Code (in Document Body)

```
\section{Main Results} % Assume this is section 1

\begin{definition} \label{def:prime}
A natural number $p > 1$ is called \emph{prime}
if its only positive divisors are $1$ and $p$.
\end{definition}
```

Using Theorem Environments

Writing Theorems, Lemmas, Proofs (with Inline Math)

```
\begin{lemma}[Euclid's Lemma] \label{lem:euclid}
If a prime $p$ divides the product $ab$, then $p$
divides $a$ or $p$ divides $b$. (Here $a, b \in \mathbb{Z}$.)
\end{lemma}
```

Using Theorem Environments

Writing Theorems, Lemmas, Proofs (with Inline Math)

```
\begin{theorem} \label{thm:inf_primes}
There are infinitely many prime numbers.
The set $\{p \mid p \text{ is prime}\}$ is infinite.
\end{theorem}
```

Using Theorem Environments

Writing Theorems, Lemmas, Proofs (with Inline Math)

```
\begin{proof} % Predefined environment
Suppose there are finitely many primes $p_1, \dots, p_n$.
Consider $N = p_1 p_2 \dots p_n + 1$.
$N$ must have a prime divisor $p$. By assumption, $p=p_i$
for some $i \in \{1, \dots, n\}$. Then $p$ divides $N$ and $p$
divides $N-1 = p_1 \dots p_n$, so $p$ must divide their
difference $N - (N-1) = 1$. This is a contradiction,
as no prime divides 1 ($p \nmid 1$).
Therefore, the set of primes must be infinite.
% QED symbol is added automatically
\end{proof}
See \cref{def:prime}, \cref{lem:euclid}, and \cref{thm:inf_primes}.
```

Why Define New Environments?

Simplifying Repetitive Structures

Often, you find yourself typing the same complex structure repeatedly. Examples:

- A specific type of theorem or definition box.
- A consistently formatted warning or note block.
- A container for code examples with specific formatting.

Benefits

- **Reduce Repetition (DRY):** Write the complex code once.
- **Ensure Consistency:** All instances look the same.
- **Improve Readability:** Replace complex code with a meaningful environment name like `\begin{warning}`.
- **Simplify Maintenance:** Change the definition in one place to update all instances.

The `\newenvironment` Command

Syntax Breakdown

You define new environments in the **preamble** of your document using `\newenvironment`.

Syntax

```
\newenvironment{envname}[numargs][optargdefault]{begincode}{endcode}
```

- `{envname}`: The name of your new environment (e.g., `warning`, `mytheorem`). Must be unique!
- `[numargs]`: Optional. A number from 1 to 9 indicating how many **mandatory** arguments the environment takes. If omitted, it takes zero mandatory arguments.

The `\newenvironment` Command

Syntax Breakdown

- `[optargdefault]`: Optional. If present, the environment takes **one optional** argument as its *first* argument. `optargdefault` is the value used if the optional argument is not provided by the user. Note: This syntax means it takes *one* optional, plus the number of mandatory args specified by '[numargs]'.
- `{begincode}`: The LaTeX code executed when `\begin{envname}` is encountered. Mandatory arguments are accessed using `#1`, `#2`, ..., `#numargs`.
- `{endcode}`: The LaTeX code executed when `\end{envname}` is encountered. Cannot directly use arguments (`#1`, etc.).

Definitions using `\newenvironment` typically go in the document preamble (before `\begin{document}`).

Example 1: Simple Environment

No Arguments

Let's create an environment for highlighting important notes using Beamer's `alertblock`.

Definition (in Preamble)

```
\newenvironment{highlightnote}% Environment name (0 args)
  {\begin{alertblock}{Important Note}}% Begin code
  {\end{alertblock}}% End code
```

Usage (in Document Body)

```
Some regular text...
\begin{highlightnote}
  This point is crucial for understanding the next slide.
  Remember to compile multiple times!
\end{highlightnote}
More regular text...
```

Example 1: Simple Environment

No Arguments

Expected Output

The text "This point is crucial..." appears inside a standard Beamer 'alertblock' (often visually distinct, e.g., red background/border) with the title "Important Note".

Important Note

This point is crucial for understanding the next slide. Remember to compile multiple times!

Example 2: Mandatory Argument

Defining a Custom Definition Box

Let's create an environment that takes the term being defined as an argument.

Definition (in Preamble)

```
\newenvironment{mydefinition}[1]% Env name, [1] = one mandatory arg
  {\begin{block}{Definition: #1}\itshape}% Begin code, uses #1
  {\end{block}}% End code
```

Note: `#1` *in the 'begincode' is replaced by the argument provided by the user.* `\itshape` *makes the body italic.*

Usage (in Document Body)

```
\begin{mydefinition}{Kerning}
  The adjustment of space between pairs of letters
  to improve visual appearance.
\end{mydefinition}
```

Example 2: Mandatory Argument

Defining a Custom Definition Box

Expected Output

A standard Beamer ‘block’ appears with the title ”Definition: Kerning”. The body text (”The adjustment of space...”) is displayed inside the block in italics.

Definition: Kerning

The adjustment of space between pairs of letters to improve visual appearance.

Example 3: Optional Argument

Creating a Block with an Optional Title

Let's create a block where the user *can* provide a title, but it defaults to "Default Title" if they don't.

Definition (in Preamble)

```
\newenvironment{titledblock}[1][Default Title]% [1] Optional arg
                                              % Default="Default Title"
  {\begin{block}{#1}}% Begin code, #1 gets provided value or default
  {\end{block}}% End code
```

Note: The `[1]` *indicates one optional argument.* `#1` *refers to this argument's value.*

Usage (in Document Body)

```
% Case 1: No optional argument provided
\begin{titledblock}
  Some content using the default title.
\end{titledblock}
```

Example 3: Optional Argument

Creating a Block with an Optional Title

Another Example (in Document Body)

```
% Case 2: Optional argument IS provided
\begin{titledblock}[My Custom Title]
  Some content using a specified title.
\end{titledblock}
```

Example 3: Optional Argument

Creating a Block with an Optional Title

Expected Output

Two Beamer blocks appear:

- The first block has the title ”Default Title”.
- The second block has the title ”My Custom Title”.

Default Title

Some content using the default title.

My Custom Title

Some content using a specified title.

Important Considerations

Things to Keep in Mind

- **Naming Conflicts:** Environment names must be unique. Avoid redefining existing LaTeX or package environments unless you know exactly what you're doing. Use prefixes (e.g., `my...`, `tut...`) to reduce clashes.
- **`\renewenvironment`:** If you *intentionally* want to change an existing environment (use with extreme caution!), use `\renewenvironment` instead of `\newenvironment`. It has the same syntax but will overwrite the old definition.
- **Scope:** Definitions are typically global when placed in the preamble.
- **Arguments in `endcode`:** The mandatory arguments (`#1`, etc.) are **not** directly available in the `endcode`. If needed, you'd have to save them in the `begincode` using temporary macros (more advanced).
- **Complexity:** Overly complex environments can become hard to debug.

Subfigure, amsmath, and new environment in LaTeX

Thank You for Listening!