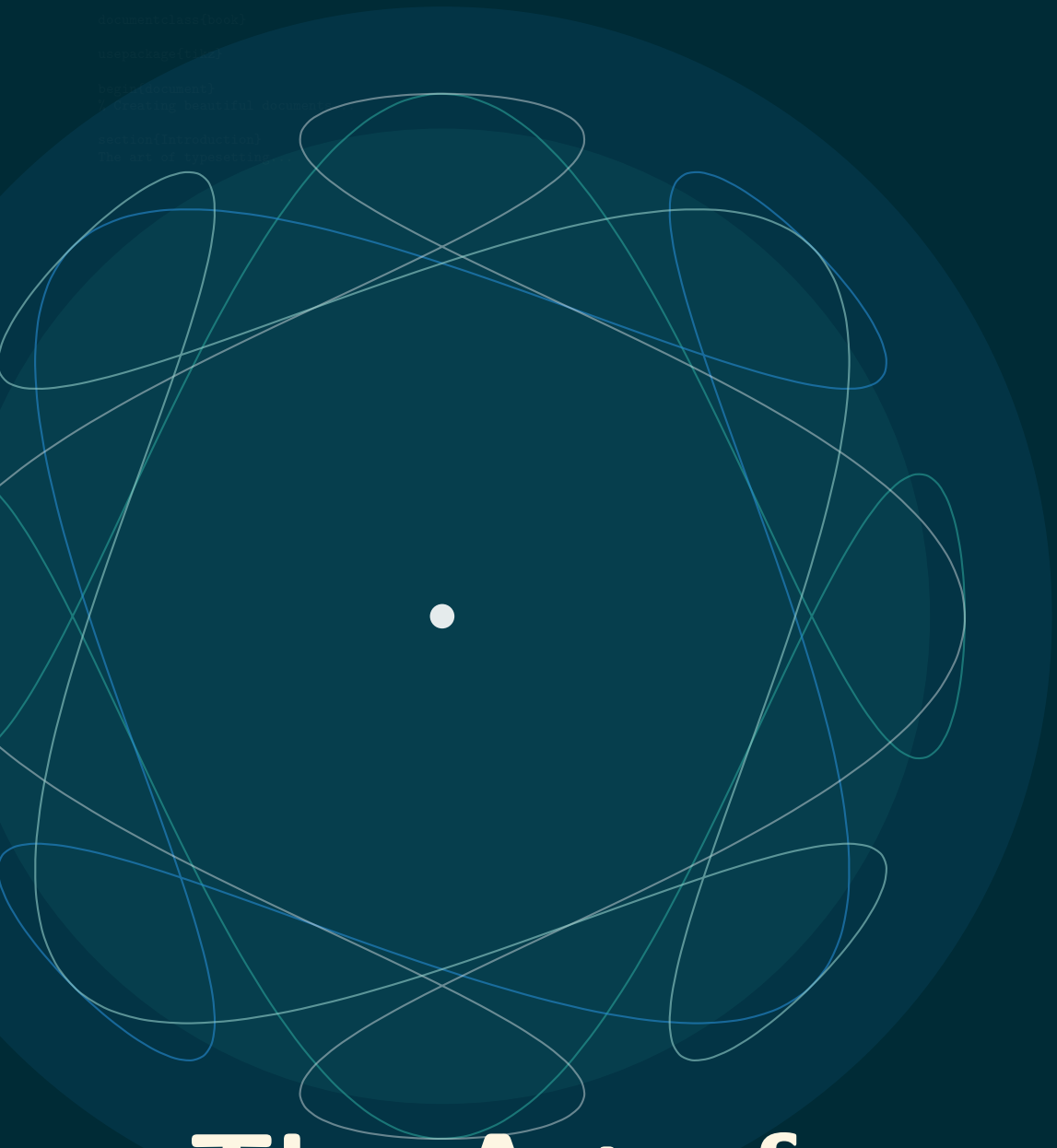


LECTURE NOTES

The Art of Technical LaTeX

Modern Typography and Vector Graphics

From Standard Documents to Advanced TikZ

Ji, Yonghyeon

A Comprehensive Guide to LaTeX

The Art of Technical LaTeX

Modern Typography and Vector Graphics

Ji, Yonghyeon

February 19, 2026

WINTER 2026

Copyright

Changelog

v1.2	2026-02-29	Add “Vector Graphic: TikZ”.
v1.1	2026-01-19	Add installation and poster.
v1.0	2026-01-08	Initial release.

Contents

0 Installing LaTeX with TeXworks and TeXstudio

A working LaTeX setup has two parts:

- a **LaTeX distribution** (compiler + packages), such as **MiKTeX** or **TeX Live** (Windows/Linux) and **MacTeX** (macOS),
- an **editor** to write and compile `.tex` files, such as **TeXworks** or **TeXstudio**.

0.1 Windows Installation

TBA

0.2 Linux Installation

Debian/Ubuntu:

```
sudo apt update
sudo apt install texlive texlive-latex-extra texlive-fonts-recommended texlive-science
```

Fedora:

```
sudo dnf install texlive-scheme-basic texlive-collection-latexextra
```

Arch Linux:

```
sudo pacman -S texlive-core texlive-latexextra texlive-fontsextra
```

Install TeXworks (Linux)

TeXworks is typically available as a package:

```
# Debian/Ubuntu
sudo apt install texworks
```

Install TeXstudio (Linux)

```
# Debian/Ubuntu
sudo apt install texstudio
```

0.3 macOS Installation

MacTeX is a macOS distribution based on TeX Live.

1. Download and install **MacTeX** using the official `.pkg` installer.
2. After installation, ensure the TeX binaries are available in your shell environment.

A common check in Terminal is:

```
pdflatex --version
```


1 Introduction to LaTeX

LaTeX (often pronounced "Lay-tek" or "Lah-tek") is a high-quality typesetting system widely used for professional scientific and technical documents.

Instead of formatting text by hand (like in a word processor), we write a plain-text source file and then *compile* it into a polished PDF.

1.1 What is LaTeX?

What is LaTeX?

LaTeX is a document preparation system based on TeX, designed for producing beautifully typeset documents, especially those containing complex mathematical expressions.

A helpful way to think about LaTeX is:

- We write *content* and *structure* (sections, theorems, figures, references).
- The compiler applies consistent typography and layout rules automatically.

This separation makes long documents easier to maintain: if we change the structure (add sections, figures, equations), numbering and references can update automatically when compiled again.

What TeX and LaTeX Do

- TeX is the underlying typesetting engine (it focuses on high-quality layout and spacing).
- LaTeX is a set of macros and conventions on top of TeX that provides document structure: chapters/sections, figures, tables, bibliographies, and many more.

We usually write LaTeX source and let the engine produce the final PDF.

Why Use LaTeX? People choose LaTeX because it is strong where word processors often become difficult:

- **Professional-quality typography** with consistent formatting across the whole document.
- **Excellent mathematical rendering**, for example $a^2 + b^2 = c^2$ and complex formulas.
- **Automatic numbering and references** for equations, figures, sections, and bibliographies.
- **Scales well** to long reports, theses, and books: structure stays manageable.

When LaTeX Is a Good Fit

- Math-heavy documents (homework, lecture notes, papers).
- Reports with many figures/tables and cross-references.
- Theses and books with chapters and bibliographies.

When It Might Not Be Necessary

- Very short, simple documents where structure and references do not matter.
- Layout-heavy marketing designs (unless you already know advanced packages).

1.1.1 Compiling a Document

A LaTeX file is typically saved with the extension `.tex`. To produce a PDF, you run a compiler (engine) that reads the source and outputs the final document.

The Basic Compile Command A common engine is `pdflatex`. The typical workflow is:

```
pdflatex file.tex
pdflatex file.tex % run twice to update references
```

Running twice matters because references (like section numbers, equation numbers, and the table of contents) are resolved across compilation passes.

What Happens During Compilation When you compile, LaTeX:

- reads your structure (chapters, sections, lists),
- calculates layout (line breaks, page breaks),
- assigns numbers (figures, tables, equations),
- writes auxiliary files (used to resolve references),
- produces a PDF output.

Recommended Tools We can write and compile LaTeX using:

- **Overleaf**: online editor; no installation required.
- **TeX Live**: a full TeX distribution commonly used on Linux/macOS.
- **MiKTeX**: a popular distribution on Windows.

1.2 Document Structure and Classes

A LaTeX source file is a plain text file, usually saved with the extension `.tex`. When you compile it, LaTeX produces a beautifully typeset output (most commonly a PDF).

1.2.1 Setup and Content

A LaTeX document is built from two main parts:

- **Preamble** (before `\begin{document}`): chooses the document class and loads packages.
- **Document body** (between `\begin{document}` and `\end{document}`): contains the content you want printed (text, math, figures, tables, etc.).

This "preamble + body" split is the standard structure shown in the reference material.

Example 1.1 (Minimal Structure)**.**

```
\documentclass{article}

\begin{document}
   This is a LaTeX document.
\end{document}
```

1.2.2 Document Classes

The first line of a LaTeX file is often the most important:

`\documentclass[options]{class}`

It sets the foundation for the entire document.

`article`	best for **shorter documents** such as papers, essays, homework, and reports.
`report`	used for **longer documents** that need chapters (e.g., theses).
`book`	designed for **books**; supports chapters and is set up for two-sided printing.
`beamer`	used for **presentations**; the output becomes slide-based rather than page-based.

Class options customize the class behavior.

`10pt`, `11pt`, `12pt`	base font size (default is commonly `10pt`).
`a4paper`, `letterpaper`	paper size
`twocolumn`	two-column layout.
`oneside`, `twoside`	single-sided or double-sided layout.

Example 1.2. `\documentclass[12pt, a4paper, twoside]{article}`

Remark. A beginner-friendly recommendation:

- Use `11pt` or `12pt` for readable notes.
- Use the correct paper size for your country/institution.
- Avoid `twocolumn` until you are comfortable with layout.

1.2.3 Packages

Packages are extensions that add new features and commands. We load them in the preamble using `\usepackage{...}`.

```
\usepackage{amsmath, amssymb}
\usepackage{graphicx}
\usepackage{hyperref}
```

`amsmath`, `amssymb`	improved math environments and symbols.
`graphicx`	enables `\includegraphics` for inserting images.
`hyperref`	clickable links and clickable cross-references in the PDF.

Example 1.3 (Margins with `geometry`)**.**

```
\usepackage[left=1.5in, right=1.5in, top=1in, bottom=1in]{geometry}
```

1.2.4 Skeleton Template

A typical skeleton combines class selection, packages, metadata (title/author/date), and document body commands. The reference provides a standard example.

Example 1.4 (Typical Structure)**.**

```
\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath, amssymb}
\usepackage{graphicx}
\usepackage{hyperref}

\title{My First Document}
\author{Student Name}
\date{\today}

\begin{document}
    \maketitle

    Hello, LaTeX World!

\end{document}
```


1.3 Text Formatting and Layout

1.3.1 Basic Text Commands

LaTeX offers two main approaches to text styling:

- **Direct styling commands** such as `\textbf` and `\textit`.
- **Logical emphasis** using `\emph`, which adapts to context.

The most common beginner commands are:

- `\textbf{bold}` → **bold**
- `\textit{italic}` → *italic*
- `\underline{underline}` → underline
- `\emph{emphasized}` → *emphasized*

A useful rule:

- Use `\emph` when you mean "emphasize this concept."
- Use `\textit` or `\textbf` when you want a specific visual style.

1.3.2 Paragraphs and Line Breaks

In LaTeX, a new paragraph is created by leaving a blank line in your source. A manual line break can be forced using `\\`.

Example 1.5.

```
This is the first line.\\
This is the second line.
```

1.3.3 Quotes and Quotations

- `quote`: short quotations (indented, typically one paragraph).
- `quotation`: longer quotations (indented, with paragraph formatting).

Example 1.6.

```
\begin{quote}
    A well-structured document is easier to write and easier to read.
\end{quote}
```


1.3.4 Text Alignment

Alignment environments are helpful for small blocks of text:

- `\begin{center}...\end{center}` → centered text
- `\begin{flushleft}...\end{flushleft}` → left-aligned text
- `\begin{flushright}...\end{flushright}` → right-aligned text

Example 1.7.

centered text

left-aligned text

right-aligned text

1.3.5 Font Sizes

LaTeX provides relative font size switches that affect the enclosed group:

- `\tiny`, `\scriptsize`, `\footnotesize`
- `\small`, `\normalsize`, `\large`, `\Large`
- `\LARGE`, `\huge`, `\Huge`

Example 1.8.

1.3.6 Special Characters

Some characters are reserved because LaTeX uses them for commands or syntax.

`\%`	% (percent)
`\$`	$ (dollar)
`\#`	# (hash)
`\_`	_ (underscore)
`\&`	& (ampersand)
`\{ \}`	{ } (braces)

Use `%` to write comments that will not appear in the output PDF:

```
% This is a comment and will not appear in the output
```

1.4 Lists and Tables

1.4.1 Unordered Lists (Bulleted)

Use the `itemize` environment to create bulleted lists. Each list item begins with `\item`.

Example 1.9.

```
\begin{itemize}
    \item Apples
    \item Bananas
    \item Cherries
\end{itemize}
```

- Apples
- Bananas
- Cherries

Example 1.10. We can nest `itemize` to create sub-bullets.

```
\begin{itemize}
    \item Fruit
    \begin{itemize}
        \item Apples
        \item Bananas
    \end{itemize}
    \item Vegetables
\end{itemize}
```

- Fruit
  - Apples
  - Bananas
- Vegetables

1.4.2 Ordered Lists (Numbered)

Use the `enumerate` environment for numbered lists. LaTeX automatically handles numbering, which is exactly the beginner-friendly point emphasized in the reference notes.

Example 1.11.

```
\begin{enumerate}
    \item First
    \item Second
    \item Third
\end{enumerate}
```

1. First
2. Second
3. Third

Example 1.12. We can nest `enumerate` just like `itemize`:

```
\begin{enumerate}
    \item Main step
    \begin{enumerate}
        \item Sub-step
        \item Sub-step
    \end{enumerate}
    \item Next main step
\end{enumerate}
```

1. Main step
    (a) Sub-step
    (b) Sub-step
2. Next main step

1.4.3 Description Lists

Use the `description` environment when each item needs a label.

Example 1.13.

```
\begin{description}
    \item[Dog] A friendly animal.
    \item[Cat] A curious animal.
\end{description}
```

Dog A friendly animal.

Cat A curious animal.

1.4.4 Tables

Tables in LaTeX are typically built in two layers:

- `tabular`: the grid itself (rows/columns).
- `table`: an optional floating wrapper for captions and labels.

Example 1.14.

```
\begin{tabular}{|l|c|r|}
   \hline
   Name & Age & Score \\
   \hline
   Alice & 24 & 95 \\
   Bob & 22 & 88 \\
   \hline
\end{tabular}
```

Name	Age	Score
Alice	24	95
Bob	22	88

- `&` separates columns.
- `\\` ends a row.
- `\hline` draws a horizontal line.

Alignment Options in Tables The column specification controls alignment:

- `l` – left aligned
- `c` – centered
- `r` – right aligned
- `|` – vertical line between columns

Beyond `l`, `c`, `r`, we will often need fixed-width columns:

- `p{3cm}`: a paragraph column of width 3cm (text wraps automatically).

Example 1.15 (Fixed-width column)**.**

```
\begin{tabular}{|l|p{6cm}|}
   \hline
   Term & Explanation \\
   \hline
   LaTeX & A document preparation system for professional typesetting. \\
   \hline
\end{tabular}
```

Term	Explanation
LaTeX	A document preparation system for professional typesetting.

1.4.5 Floating Tables

A `table` environment is a *float*: LaTeX may move it to an optimal location for page layout.

Example 1.16. Standard Pattern: `table` + `tabular` + caption/label

```
\begin{table}[h]
    \centering
    \begin{tabular}{lrr}
        \hline
        Item & Qty & Price \\
        \hline
        Apples & 3 & 1.20 \\
        Bananas & 5 & 0.80 \\
        \hline
    \end{tabular}
    \caption{Fruit Inventory}
    \label{tab:fruit}
\end{table}
```

Item	Qty	Price
Apples	3	1.20
Bananas	5	0.80

Table 1: Fruit Inventory

Placement Options The optional argument `[h]` suggests "place here". Common options:

`h`	here
`t`	top of page
`b`	bottom of page
`p`	float page

- `h`: here
- `t`: top of page
- `b`: bottom of page
- `p`: float page

A common practical choice is `[htbp]` to give LaTeX flexibility.

Captions, Labels, and Referencing A best practice:

- Put `\label` after `\caption`.
- Refer to the table using `Table~\ref{tab:fruit}`. (Table 1)

1.5 Mathematical Typesetting

Mathematics is one of the strongest reasons to use LaTeX. Compared to ordinary word processors, LaTeX typesets formulas with consistent spacing, professional symbol shapes, and stable numbering. Once we learn the patterns, we can write math quickly and keep it readable in both short notes and long theses.

A major advantage is that LaTeX treats math as a structured language: we build expressions from commands (like `\frac` or `\sum`) rather than trying to align symbols manually.

1.5.1 Inline and Display Mathematics

We distinguish two modes: inline math (inside a sentence) and display math (on its own line).

Inline Mathematics Inline math is used inside a sentence.

Example 1.17.

```
The quadratic polynomial $ax^2 + bx + c$ has degree $2$.
```

```
The quadratic polynomial \(ax^2 + bx + c\) has degree \(2\).
```

The quadratic polynomial $ax^2 + bx + c$ has degree 2.

Displayed Mathematics Displayed math is used for important equations that appear on their own line. We list display math forms such as `\[...\]` (and also mentions `$$...$$`).

Example 1.18.

```
   The quadratic polynomial $$ax^2 + bx + c$$ has degree $2$.
```

```
   The quadratic polynomial \[ax^2 + bx + c\] has degree \(2\).
```

The quadratic polynomial

$$ax^2 + bx + c$$

has degree 2.

1.5.2 Numbered vs. Unnumbered Display Math

- Use `equation` when you want a number you can reference later.
- Use an unnumbered display (commonly `\[...\]`) for one-off formulas you will not reference.

Example 1.19 (Numbered display)**.**

```
\begin{equation}
    \int_0^\infty e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2}
\end{equation}
```

$$\int_0^\infty e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2} \tag{1}$$

Example 1.20 (Unnumbered display)**.**

```
\[
\int_0^\infty e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2}
\]
```

$$\int_0^\infty e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2}$$

1.5.3 Superscripts and Subscripts

Superscripts and subscripts are fundamental.

- Superscripts use `^`: x^2, a^{n+1}.
- Subscripts use `_`: a_1, b_{ij}.

Example 1.21.

```
The sequence $a_n = n^2$ grows quadratically.
```

The sequence $a_n = n^2$ grows quadratically.

Remark.

- `a_12` means $a_1 2$ (subscript applies only to 1).
- `a_{12}` means a_{12} with subscript 12.

1.5.4 Fractions and Roots

Fractions and roots are also highlighted as essential commands.

Example 1.22.

```
The solution is $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.
```

The solution is $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

1.6 Theorems, Definitions, and Proofs

1.6.1 Package: `amsthm`

Mathematical writing often needs structured statements such as definitions, theorems, lemmas, corollaries, and remarks.

Statement type	Typical role
Definition	Introduces terminology; fixes conventions and notation.
Theorem	Principal result; proved in full detail.
Lemma	Auxiliary/technical result used in proofs.
Corollary	Immediate consequence of a theorem.
Proposition	Intermediate result; often specialized or less central than a theorem.
Claim	Local assertion inside a proof; proved on the spot.
Remark	Intuition, caveat, discussion of scope or sharpness.
Example	Illustrative instance; may justify hypotheses.
Notation	Declares symbols and standing conventions.
Assumption	Standing hypotheses for a section/chapter.
Observation	Minor fact clarifying structure.

Note. The naming hierarchy is conventional and varies by author; prioritize clarity.

In LaTeX, the most standard tool for this is the `amsthm` package, which provides:

- theorem-like environments with automatic numbering,
- consistent formatting (bold theorem headings, italic theorem body, etc.),
- a built-in `proof` environment that automatically prints a QED symbol.

Example 1.23. In our preamble, we define theorem-style environments like this:

```
\usepackage{amsthm}

\newtheorem{theorem}{Theorem}[section]
\newtheorem{lemma}[theorem]{Lemma}
\newtheorem{definition}[theorem]{Definition}
\newtheorem{corollary}[theorem]{Corollary}

\theoremstyle{remark}
\newtheorem*{remark}{Remark}
```

- `\usepackage{amsthm}` loads the theorem framework.
- `\newtheorem{theorem}{Theorem}[chapter]` creates an environment named `theorem`. Its printed title is "Theorem", and it resets numbering each chapter. So you get Theorem 1.1, 1.2, then Theorem 2.1, etc.

- `\newtheorem{lemma}[theorem]{Lemma}` makes `lemma` share the same counter as `theorem`. That means a lemma can be Lemma 2.3 right after Theorem 2.2.
- `\theoremstyle{remark}` switches to a different style (usually non-italic body text).
- `\newtheorem*{remark}{Remark}` creates an unnumbered environment `remark`. The star `*` means "no numbering".

`\theoremstyle{...}`	**Heading**	**Body**
`plain`	Bold (upright)	Italic
`definition`	Bold (upright)	Upright (roman)
`remark`	Italic	Upright (roman)

Environment (examples)	**Style**	**Rationale**
Theorem, Lemma, Proposition, Corollary	`plain`	Results/proofs read well with italic body.
Definition, Assumption, Notation	`definition`	Declarative text is usually upright.
Remark, Observation, Example	`remark`	Commentary often uses italic heading and upright body.

Numbered vs. Unnumbered Statements

- Use numbered statements for items you will reference later.
- Use unnumbered statements for short comments, informal notes, or one-off remarks.

1.6.2 Basic Environments

A good theorem section begins with clear definitions. Definitions tell the reader what words mean before you prove anything about them.

Example 1.24 (Definition Statements)**.**

```
\begin{definition}[Even Number]
   An integer $n$ is called \emph{even} if there exists an integer $k$ such that $n = 2k$.
\end{definition}
```

Definition 1.1 (Even Number)**.** *An integer n is called* even *if there exists an integer k such that $n = 2k$.*

Example 1.25 (Theorem Statements)**.** A theorem states a mathematical fact we will justify with a proof.

```
\begin{theorem}[Sum of Evens]
   The sum of two even integers is even.
\end{theorem}
\begin{proof}
   Let $a = 2m$ and $b = 2n$ for some integers $m$ and $n$.
   Then $a + b = 2m + 2n = 2(m + n)$, which is divisible by $2$.
   Therefore, $a + b$ is even.
\end{proof}
```

Theorem 1.2 (Sum of Evens)**.** *The sum of two even integers is even.*

Proof. Let $a = 2m$ and $b = 2n$ for some integers m and n. Then $a + b = 2m + 2n = 2(m + n)$, which is divisible by 2. Therefore, $a + b$ is even. □

1.6.3 Referencing Theorems and Definitions

In real writing, we should label key statements and refer back to them later. Use `\label` inside the theorem-like environment, then refer using `\ref`.

Example 1.26 (Label and reference)**.**

```
\begin{theorem}\label{thm:sum-even}
   The sum of two even integers is even.
\end{theorem}

By Theorem~\ref{thm:sum-even}, we conclude that ...
```

Theorem 1.3. *The sum of two even integers is even.*

By Theorem 1.3, we conclude that ...

1.7 Figures and Graphics

1.7.1 Packages: `graphicx`

To insert images in LaTeX, you typically load the `graphicx` package. This package provides the command `\includegraphics`, which is the main tool for importing external images.

```
\usepackage{graphicx}
```

A practical beginner rule:

- Put image files in the same folder as our `.tex` file, or in a dedicated folder like `figures/`.
- If using a folder, include it in the path (e.g., `figures/plot.pdf`).

Common File Formats Most workflows accept:

- `.png`, `.jpg` for raster images (screenshots, photos),
- `.pdf` for vector graphics (plots, diagrams exported from tools).

1.7.2 Basic Usage: `\includegraphics`

The most common professional pattern is to place an image inside a `figure` environment, center it, add a caption, add a label, and then reference it in the text.

Example 1.27.

```
\begin{figure}[h]
    \centering
    \includegraphics[width=.6\linewidth]{example-image-a}
    \caption{A sample figure}
    \label{fig:sample}
\end{figure}
See Figure~\ref{fig:sample}.
```

- `\begin{figure}...\end{figure}` creates a *float* (LaTeX may move it for better layout).
- `\centering` centers the image.
- `\caption` creates a numbered caption.
- `\label` creates a reference key.
- `\ref` prints the correct figure number (and updates automatically if numbering changes).

Example 1.28 (Controlling Image Size)**.**

- `width=...` e.g., `width=0.6\textwidth`
- `height=...` e.g., `height=4cm`
- `scale=...` e.g., `scale=0.8`

```
...
\includegraphics[scale=0.5]{diagram}
...
```

Example 1.29. For consistent layout, `width=...` is usually the best choice. A common pattern is:

- `width=\linewidth` inside a minipage or narrow container,
- `width=0.6\textwidth` for a medium figure in normal text.

If we specify both width and height, the image can stretch. A safer pattern is:

```
...
\includegraphics[width=0.7\textwidth, keepaspectratio]{plot}
...
```

1.7.3 Positioning Figures

Figures are floats, so LaTeX chooses a good placement by default. We can provide placement hints with:

- `h` — here (approximately)
- `t` — top of page
- `b` — bottom of page
- `p` — on a separate float page

1.7.4 Referencing Figures

Use `\label` and `\ref` to refer to figures. The reference notes explicitly demonstrate referencing with `Figure~\ref{...}`.

```
See Figure~\ref{fig:sample} for a visual illustration.
```

Use prefixes:

- `fig:...` for figures,
- `tab:...` for tables,
- `eq:...` for equations.

1.7.5 Side-by-Side Images

Side-by-side layouts are useful for comparisons (before/after, method A vs. method B). A simple approach is to use `minipage` blocks inside one `figure` float.

Example 1.30.

```
\begin{figure}[h]
    \centering
    \begin{minipage}{0.48\textwidth}
        \centering
        \includegraphics[width=\linewidth]{img1}
        \caption{Image 1}
        \label{fig:img1}
    \end{minipage}
    \hfill
    \begin{minipage}{0.48\textwidth}
        \centering
        \includegraphics[width=\linewidth]{img2}
        \caption{Image 2}
        \label{fig:img2}
    \end{minipage}
\end{figure}
```

The above produces two captions in one float, which can be acceptable but may look unusual in some styles. A more "textbook/paper" solution is to use subfigures (commonly via the `subcaption` package), so we get one main caption plus (a)/(b) subcaptions.

```
\begin{figure}[h]
    \centering
    % subfigure A here
    % subfigure B here
    \caption{Comparison of two methods}
    \label{fig:comparison}
\end{figure}
```


1.8 Cross-Referencing and Labels

Cross-referencing is one of the most important "professional writing" features in LaTeX. Instead of hard-coding numbers like "see Section 3" or "see Figure 2", you create references that update automatically.

Cross-referencing allows you to:

- refer to sections, figures, tables, theorems, and equations dynamically,
- automatically update numbering when you insert, delete, or reorder content,
- improve navigability in long documents (especially with clickable links in the PDF),
- reduce mistakes (no outdated references after edits).

This is exactly the workflow emphasized in the reference notes: label items with `\label{...}` and refer with `\ref` or `\eqref`.

Example 1.31 (Section Reference)**.**

```
\section{Methodology}
\label{sec:method}

As discussed in Section~\ref{sec:method}, we use...
```

Example 1.32.

```
\begin{figure}[h]
    \centering
    \includegraphics[width=0.5\textwidth]{plot}
    \caption{Data trend}
    \label{fig:trend}
\end{figure}

See Figure~\ref{fig:trend}.
```

This matches the reference approach: caption, label, then `Figure~\ref{...}` in text.

Example 1.33.

```
\begin{table}[h]
    \centering
    \begin{tabular}{lrr}
        \hline
        Item & Qty & Price \\
        \hline
        Apples & 3 & 1.20 \\
        Bananas & 5 & 0.80 \\
        \hline
    \end{tabular}
    \caption{A neat table}
    \label{tab:neat}
\end{table}

See Table~\ref{tab:neat}.
```


Example 1.34. Equations are a major reason cross-referencing matters: we often need to cite a specific formula later. The reference notes show a numbered equation with `\label` and a reference using `Eq.~\eqref{...}`.

```
\begin{equation}
    a^2 + b^2 = c^2
    \label{eq:pythagoras}
\end{equation}

From Eq.~\eqref{eq:pythagoras}, we know...
```

Example 1.35.

```
\begin{theorem}\label{thm:addition}
    Addition of even numbers is even.
\end{theorem}

Theorem~\ref{thm:addition} proves the statement.
```

Example 1.36. To make cross-references clickable in the PDF, load the `hyperref` package. The reference notes also show using `\href{...}{...}` for an external link.

- `\href{URL}{text}` — clickable external link text.
- `\url{URL}` — prints a clickable URL in a monospace-like style.

```
...
\usepackage{hyperref}
...

\begin{document}
    ...
    \href{https://www.latex-project.org}{Visit the LaTeX Project}
    ...
\end{document}
```

This mirrors the reference example of linking to `latex-project.org` via `\href`.

Remark. Use consistent prefixes:

- `sec:` for sections,
- `fig:` for figures,
- `tab:` for tables,
- `eq:` for equations,
- `thm:` for theorems.

1.9 Bibliography and Citations

Citations are a core part of academic and technical writing.

- give credit to original authors (avoids plagiarism),
- support claims with reliable evidence,
- help readers locate the sources for verification or further study.

A good citation system should be:

- consistent (same style everywhere),
- complete (enough details to find the source),
- easy to maintain (especially in large documents).

1.9.1 Manual Bibliography with `thebibliography`

This is the simplest approach and works well for:

- short assignments,
- one-file handouts,
- documents with only a few references.

The reference slides show the standard pattern: `\begin{thebibliography}{...}` and one `\bibitem{key}` per entry.

Example 1.37.

```
\begin{thebibliography}{9}

   \bibitem{knuth}
   D. E. Knuth,
   \emph{The \TeX book},
   Addison-Wesley, 1984.

   \bibitem{lamport}
   L. Lamport,
   \emph{\LaTeX: A Document Preparation System},
   2nd ed., Addison-Wesley, 1994.

\end{thebibliography}
```

In `\begin{thebibliography}{9}`, the `9` helps LaTeX reserve space for label widths. For small bibliographies (under 10 items), `9` is common. For up to 99 items, use `99`.

Example 1.38 (Citing a Manual Entry)**.** To cite a reference inside our text, use `\cite{key}`:

```
As shown in \cite{knuth}, the TeX system is powerful.
```


1.9.2 Using BibTeX (Recommended for Larger Projects)

For large projects, BibTeX separates bibliography data into a `.bib` file. Benefits:

- clean separation between writing and reference data,
- easy reuse of a bibliography database across multiple documents,
- automatic formatting via bibliography styles.

Example 1.39. Write BibTeX entries in a file such as `references.bib`. (Shown here as plain text you place in the `.bib` file.)

```
@book{knuth,
    author = {Donald E. Knuth},
    title = {The TeXbook},
    year = {1984},
    publisher = {Addison-Wesley}
}

@article{einstein,
    author = {Albert Einstein},
    title = {Does the Inertia of a Body Depend Upon Its Energy Content?},
    journal = {Annalen der Physik},
    year = {1905}
}
```

Near the end of your document body (where you want the bibliography to appear):

```
\bibliographystyle{plain}
\bibliography{references}
```

Cite inside the text as usual:

```
Einstein explained this in \cite{einstein}.
```

Remark. A typical compile sequence is:

- run `pdflatex` (creates citation requests),
- run `bibtex` (builds bibliography from `.bib`),
- run `pdflatex` twice (resolves references and formatting).

1.10 Custom Commands and Macros

As our documents grow, we will repeat the same notation, symbols, and formatting patterns many times. Custom commands (macros) let you write once and reuse everywhere.

Suppose we write the real numbers many times as $\mathbb{R}$. If we define a macro `\R`, then our source becomes shorter and clearer:

- without macros: `$\mathbb{R}$`
- with macros: `$\R$`

1.10.1 Defining New Commands

Use `\newcommand` in the preamble to define your own commands. A command name starts with a backslash and usually uses letters only (e.g., `\R`, `\vect`).

Syntax

```
\newcommand{\name}{replacement text}
\newcommand{\name}[n]{replacement with #1 to #n}
```

- `\name` is the command we will type in the document.
- `{replacement text}` is what LaTeX will insert when the command is used.
- `[n]` declares the number of required arguments.
- `#1`, `#2`, ... are placeholders for those arguments.

Example 1.40.

```
\newcommand{\R}{\mathbb{R}} % real numbers
\newcommand{\vect}[1]{\mathbf{#1}} % bold vectors
\newcommand{\abs}[1]{\left|#1\right|} % absolute value
```

Once defined, you can use them in math mode:

- $\mathbb{R}$ denotes the set of real numbers.
- $\mathbf{v}$ denotes a vector v typeset in bold.
- $|x|$ produces the absolute value of x.

1.10.2 A Note on Math Mode

Sometimes we want a command with a default argument. we can define an optional argument using `\newcommand` by putting a default value in brackets.

Example 1.41.

```
\newcommand{\seq}[2][n]{\{#2_1, #2_2, \dots, #2_#1\}}
```

- `\seq{a}` produces $\{a_1, a_2, \ldots, a_n\}$.
- `\seq[3]{x}` produces $\{x_1, x_2, x_3\}$.

1.10.3 Declaring Math Operators

Functions like $\sin(x)$, $\log(x)$, and lim are typeset as operators with correct spacing. If we write a function name as plain letters in math mode (like $Var(X)$), spacing and font may look wrong.

Example 1.42. To define operators with proper spacing and upright font, use `\DeclareMathOperator`:

```
\DeclareMathOperator{\Var}{Var}
\DeclareMathOperator*{\argmax}{arg\,max}
```

Inline usage:

- $\Var(X)$
- $\argmax_{x \in A} f(x)$

Display usage (showing typical operator styling):

$$\argmax_{x \in A} f(x) \qquad \Var(X)$$

Remark.

- `\DeclareMathOperator` behaves like sin: subscripts appear to the side in display math.
- `\DeclareMathOperator*` behaves like lim: subscripts can appear underneath in display math.

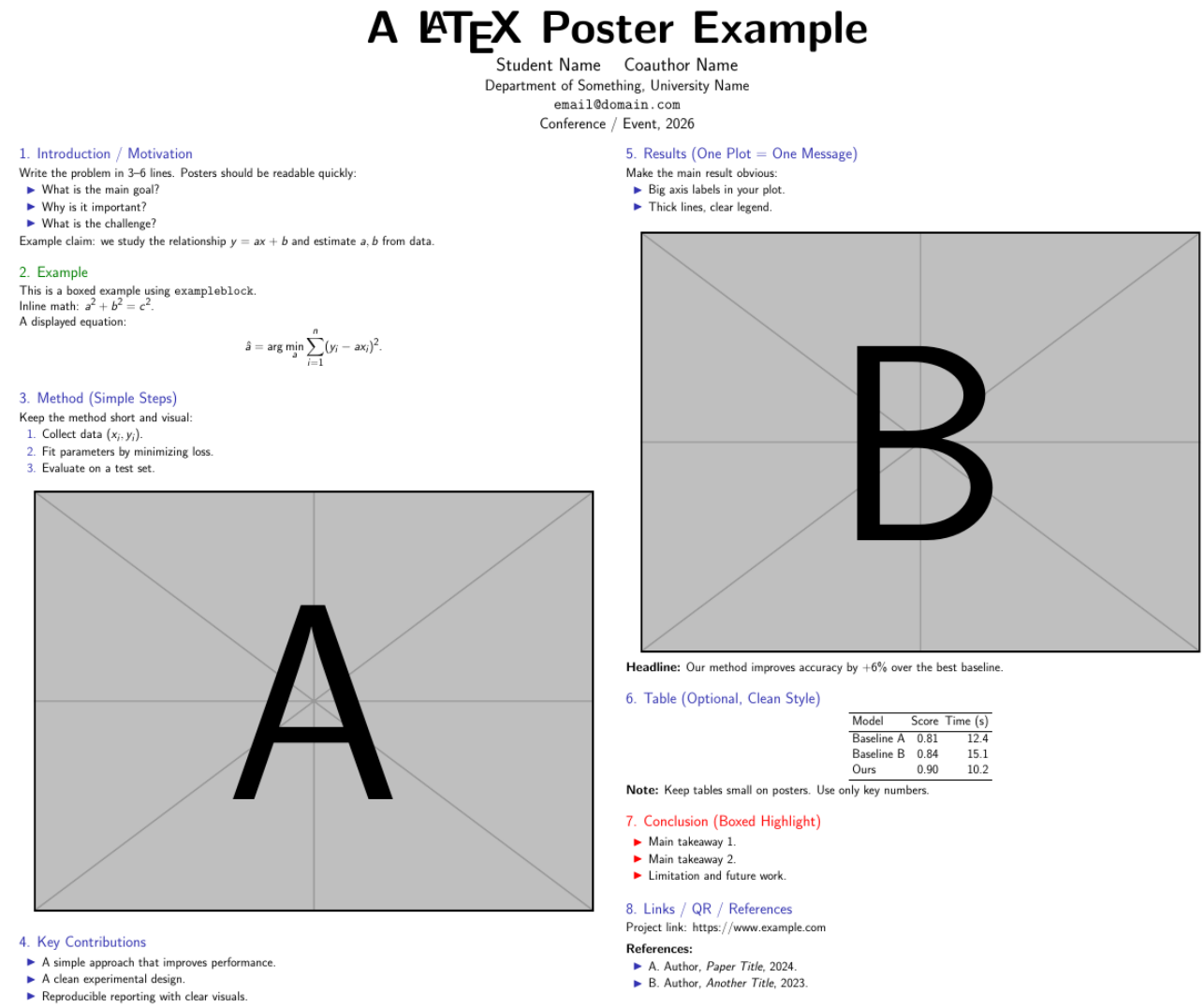
2 Posters: Beamerposter

A poster is a large-format document (often A0, A1, or 36in × 48in) designed for fast reading at a distance. A good poster is **not** a paper on a big page. It is a visual summary:

- a clear title and author line,
- 3–6 main blocks (Problem, Method, Results, Conclusion),
- a few strong figures and one key table,
- minimal text, large fonts, and consistent spacing.

For beginners, use **beamerposter** because it is stable, well-documented, and uses the familiar `beamer` "block" style.

2.1 The Poster Template



```
\documentclass[final]{beamer}

\usepackage[size=a0,orientation=portrait,scale=1]{beamerposter}

\usetheme{default}
\usecolortheme{seahorse}

\usepackage{graphicx}
\usepackage{booktabs}
\usepackage{amsmath,amssymb}
\usepackage{hyperref}

\title{A \LaTeX{} Poster Example}
\author{Student Name \and Coauthor Name}
\institute{Department of Something, University Name \\ \texttt{email@domain.com}}
\date{Conference / Event, 2026}

\setlength{\parskip}{0.35em}

\begin{document}
    \begin{frame}[t]
        % header columns ...
        % main body columns ...
    \end{frame}
\end{document}
```

2.2 Step 1: Document Class

```
\documentclass[final]{beamer}
```

- `beamer` is usually for slides.
- With `beamerposter`, one `frame` becomes one large-format poster page.
- `final` reduces draft-style marks and is appropriate for submission/printing.

2.3 Step 2: Make It Vertical (Portrait)

```
\usepackage[size=a0,orientation=portrait,scale=1]{beamerposter}
```

- 'size=a0': the poster paper size (A0 is common for conferences).
- 'orientation=portrait': vertical poster (taller than wide).
- 'scale=1': global scaling of fonts and layout elements.

Option	Example Value(s)	Meaning / Effect
size	a0, a1, a2, a3, a4, letter	Sets the poster paper size.
orientation	portrait, landscape	Controls layout direction.
scale	0.95, 1.0, 1.05, 1.1	Global scaling of fonts and layout elements.

Table 2: Common beamerposter options for size, orientation, and scale

The 'size' option sets the poster paper size. Common values include:

- a0, a1, a2, a3, a4 (ISO paper sizes)
- letter (common US sizes)

The 'orientation' option controls whether the poster is vertical or horizontal:

- orientation=portrait (vertical, tall)
- orientation=landscape (horizontal, wide)

Category	Setting	Description
Size (ISO)	size=a0	Very large poster size
Size (ISO)	size=a1	Large poster size
Size (ISO)	size=a2	Medium poster size
Size (ISO)	size=a3	Small poster size
Size (ISO)	size=a4	Page size (not typical for posters)
Size (US)	size=letter	US letter size
Orientation	orientation=portrait	Vertical poster (taller than wide)
Orientation	orientation=landscape	Horizontal poster (wider than tall)

Table 3: Typical values for size and orientation in beamerposter

2.4 Step 3: Theme and Colors

```
\usetheme{default}
\usecolortheme{seahorse}
```

- The theme controls the general style of blocks and typography.
- The color theme changes the colors of block headers and accents.
- Beginners should keep themes simple and prioritize readability.

Type	Command	Typical Use / Look
Theme	`\usetheme{default}`	Clean, minimal, safe for posters
Theme	`\usetheme{Madrid}`	Clear navigation style, common in slides
Theme	`\usetheme{Boadilla}`	Simple, modern look
Theme	`\usetheme{Warsaw}`	Darker style, strong contrast
Theme	`\usetheme{AnnArbor}`	Bold headings, more colorful
Color theme	`\usecolortheme{seahorse}`	Soft, readable tones
Color theme	`\usecolortheme{dolphin}`	Stronger contrast
Color theme	`\usecolortheme{whale}`	Darker, heavier style
Color theme	`\usecolortheme{orchid}`	More colorful accents
Color theme	`\usecolortheme{beaver}`	Red/brown academic look

Table 4: Examples of Beamer themes and color themes for posters

2.5 Step 4: Packages

```
\usepackage{graphicx}
\usepackage{booktabs}
\usepackage{amsmath,amssymb}
\usepackage{hyperref}
```

- ‘`graphicx`’: insert images with ‘`\includegraphics`’.
- ‘`booktabs`’: clean tables using ‘`\toprule`’, ‘`\midrule`’, ‘`\bottomrule`’.
- ‘`amsmath,amssymb`’: better math support (align, symbols, ‘`\arg\min`’ patterns).
- ‘`hyperref`’: clickable links via ‘`\href`’ and ‘`\url`’.

2.6 Step 5: Title, Author, Institute, Date

```
\title{A \LaTeX{} Poster Example}
\author{Student Name \and Coauthor Name}
\institute{Department ... \\ \texttt{email@domain.com}}
\date{Conference / Event, 2026}
```


2.7 Step 6: One Frame = One Poster Page

```
\begin{document}
    \begin{frame}[t]
        ...
    \end{frame}
\end{document}
```

- A poster is usually one single `frame`.
- The option `[t]` top-aligns the content, which is helpful for column layouts.

2.8 Step 7: The Header

Our header uses `columns` but only one full-width column:

```
\begin{columns}[t]
    \begin{column}{1.0\textwidth}
        \begin{block}{}
            \centering
            {\VeryHuge \textbf{\inserttitle}\par}
            \vspace{0.35em}
            {\Large \insertauthor\par}
            {\large \insertinstitute\par}
            {\large \insertdate\par}
        \end{block}
    \end{column}
\end{columns}
```

- `\inserttitle` prints the value set by `\title`.
- `\insertauthor` prints the value set by `\author`.
- `\insertinstitute` prints the value set by `\institute`.
- `\insertdate` prints the value set by `\date`.
- `\VeryHuge` is useful for posters because titles must be readable from far away.

2.9 Step 8: The Main Body Layout (Two Columns)

```
\begin{columns}[t]
    \begin{column}{0.49\textwidth}
        % left column blocks...
    \end{column}
    \begin{column}{0.49\textwidth}
        % right column blocks...
    \end{column}
\end{columns}
```

Why '`0.49\textwidth`'?

- It leaves a small gap so the columns do not collide.
- Using `0.50 + 0.50` often causes overflow in posters.

2.10 Step 9: Block Boxes

Beamer provides three useful boxed environments:

- `block`: normal content
- `exampleblock`: examples/demos
- `alertblock`: emphasis (main result, conclusion, warning)

Example patterns:

```
\begin{block}{Title} ... \end{block}
\begin{exampleblock}{Example} ... \end{exampleblock}
\begin{alertblock}{Key Message} ... \end{alertblock}
```


3 Vector Graphics: TikZ

3.1 Drawing a 2D Lattice

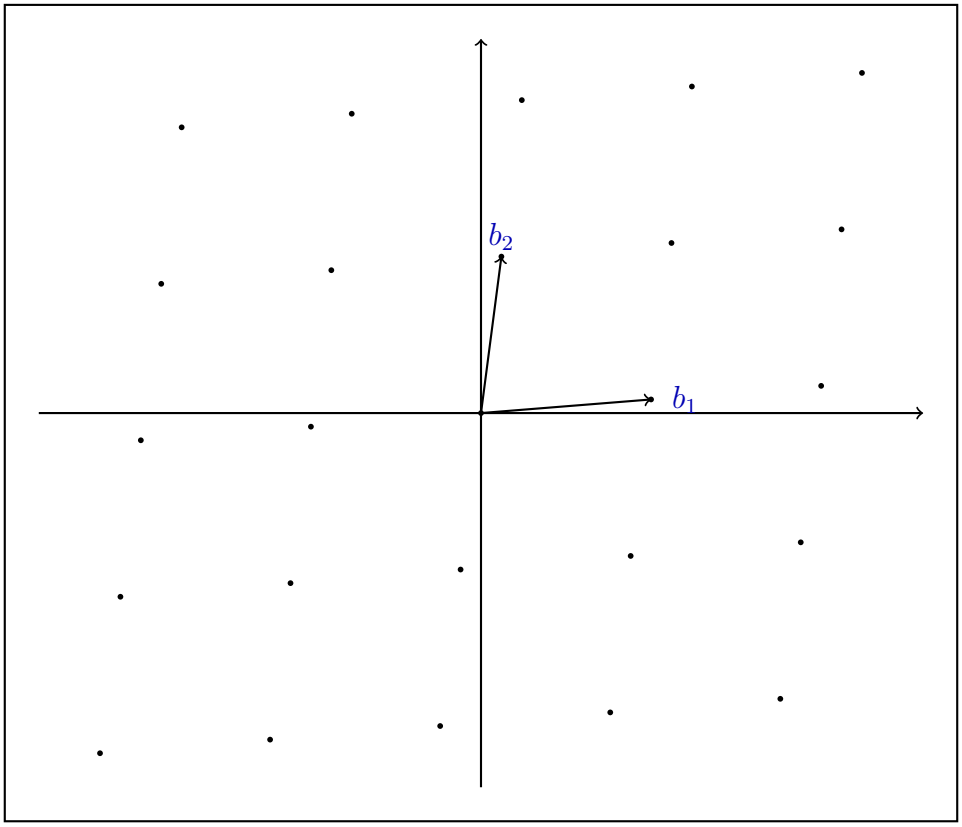
Learning Goals We will be able to:

- Draw a simple coordinate system (axes) using TikZ.
- Define vectors using `\coordinate`.
- Generate lattice points using nested `\foreach` loops.
- Label vectors and add mathematical captions.

What is a Lattice? Given two vectors $b_1, b_2 \in \mathbb{R}^2$, the **2D lattice generated by** (b_1, b_2) is:

$$\mathcal{L}(b_1, b_2) = \{x_1 b_1 + x_2 b_2 : x_1, x_2 \in \mathbb{Z}\}.$$

That means: start at the origin, move x_1 times along b_1 and x_2 times along b_2 (forward or backward), and mark every point you can reach.



$$\mathcal{L}(b_1, b_2) = \{\sum x_i b_i : x_i \in \mathbb{Z}\}$$

TikZ Idea (What are we drawing?) We will draw:

1. A border rectangle (just to frame the figure).
2. The x-axis and y-axis.
3. Two basis vectors b_1 and b_2 from the origin.
4. Points of the form $ib_1 + jb_2$ for integers i, j in a small range.
5. Labels and a formula under the diagram.

```
\begin{tikzpicture}[scale=0.9]

   % 1) draw a border box
   \draw[thick] (-7,-6) rectangle (7,6);

   % 2) draw x- and y-axes
   \draw[->, thick] (-6.5,0) -- (6.5,0);
   \draw[->, thick] (0,-5.5) -- (0,5.5);

   % 3) choose two basis vectors b1 and b2
   \coordinate (b1) at (2.5,0.2);
   \coordinate (b2) at (0.3,2.3);

   % 4) draw lattice points i*b1 + j*b2
   % (small range so it stays simple)
   \foreach \i in {-2,-1,0,1,2}{
       \foreach \j in {-2,-1,0,1,2}{
           \fill ({\i*2.5+\j*0.3},{\i*0.2+\j*2.3}) circle (1.2pt);
       }
   }

   % 5) draw the basis vectors as arrows from the origin
   \draw[->,thick] (0,0) -- (b1);
   \draw[->,thick] (0,0) -- (b2);

   % 6) label the vectors
   \node[blue!70!black, font=\large] at ($(b1)+(0.5,0)$) {$b_1$};
   \node[blue!70!black, font=\large] at ($(b2)+(0,0.3)$) {$b_2$};

   % 7) write the formula under the picture
   \node at (0,-7)
   {$\mathcal{L}(b_1,b_2)=\left\{\sum x_i b_i : x_i\in\mathbb{Z}\right\}$};

\end{tikzpicture}
```

3.1.1 The TikZ environment

- `\begin{tikzpicture}[scale=0.9]` starts the drawing.
- `scale=0.9` shrinks the entire picture to 90% of its original size.

3.1.2 Step 1: Border rectangle

```
\draw[thick] (-7,-6) rectangle (7,6);
```

- `\draw` draws lines/shapes.
- `[thick]` makes the line thicker.
- `(-7,-6)` is the bottom-left corner.
- `(7,6)` is the top-right corner.
- `rectangle` draws the rectangle using these opposite corners.

3.1.3 Step 2: Axes

```
\draw[->, thick] (-6.5,0) -- (6.5,0);
\draw[->, thick] (0,-5.5) -- (0,5.5);
```

- `--` draws a straight line segment.
- `->` adds an arrowhead at the end.
- The first line is the x-axis from $x = -6.5$ to $x = 6.5$.
- The second line is the y-axis from $y = -5.5$ to $y = 5.5$.

3.1.4 Step 3: Defining basis vectors with `\coordinate`

```
\coordinate (b1) at (2.5,0.2);
\coordinate (b2) at (0.3,2.3);
```

- `\coordinate (name) at (x,y);` stores a point with a name.
- Here, `(b1)` and `(b2)` are two vectors from the origin.
- Geometrically:

$$b_1 = (2.5, 0.2), \qquad b_2 = (0.3, 2.3).$$

3.1.5 Step 4: Lattice points using nested loops

The key idea is that every lattice point has the form:

$$(i, j) \mapsto i\, b_1 + j\, b_2.$$

We loop over integers i and j:

```
\foreach \i in {-2,-1,0,1,2}{ ... }
\foreach \j in {-2,-1,0,1,2}{ ... }
```

Inside the loops:

```
\fill ({\i*2.5+\j*0.3},{\i*0.2+\j*2.3}) circle (1.2pt);
```


3.1.6 Step 5: Drawing the basis vectors

```
\draw[->,thick] (0,0) -- (b1);    \draw[->,thick] (0,0) -- (b2);
```

- Because `(b1)` and `(b2)` are named coordinates, TikZ knows where to draw the arrows.

3.1.7 Step 6: Label placement

```
\node[...] at ($(b1)+(0.5,0)$) {$b_1$};
```

- `\node at (point) {text};` places text.
- `($(b1)+(0.5,0)$)` means "start at `b1` and shift right by 0.5".
- This uses the `calc` library: `\usetikzlibrary{calc}`.

3.1.8 Step 7: Adding the lattice definition

```
\node at (0,-7) {$\mathcal{L}(b_1,b_2)=\cdots$};
```

- This places a math formula below the box.

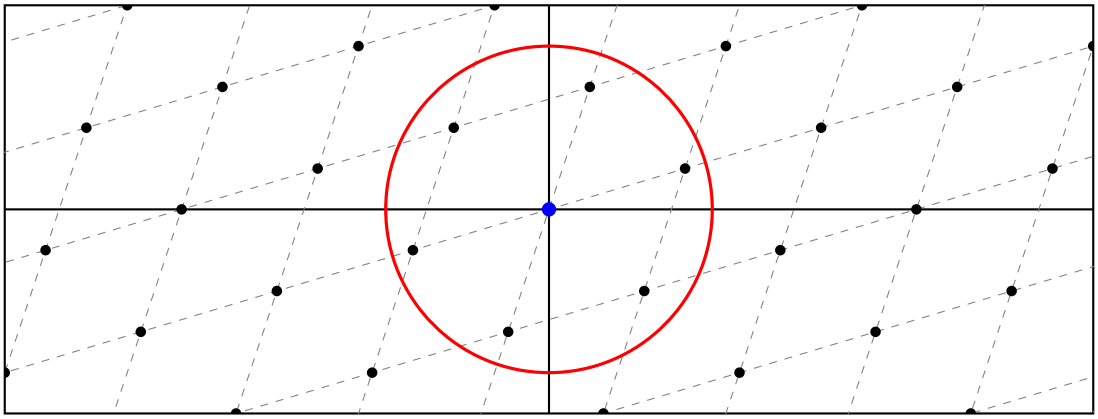
3.2 Drawing a Skew Lattice with Parallel Line Families

Learning Goals After this lesson, you will be able to:

- Use `\def` to create adjustable parameters for a TikZ figure.
- Clip drawings to a rectangular window using `\clip`.
- Use `\foreach` loops to generate repeated geometric objects.
- Draw two families of dashed parallel lines determined by two basis vectors.
- Compute lattice points $ib_1 + jb_2$ using `\pgfmathsetmacro`.
- Combine all elements into a clear lattice diagram with a circle highlight.

```
\begin{tikzpicture}[scale=0.9]

   % ---- (A) parameters you can easily change ----
   \def\xmin{-8}
   \def\xmax{ 8}
   \def\ymin{-3}
   \def\ymax{ 3}

   % basis vectors (change these to change the slant)
   \def\bax{2.0} % b1 = (bax, bay)
   \def\bay{0.6}
   \def\bbx{0.6} % b2 = (bbx, bby)
   \def\bby{1.8}

   % circle radius
   \def\R{2.4}

   % ---- (B) border (optional) ----
   \draw[thick] (\xmin,\ymin) rectangle (\xmax,\ymax);

   % clip everything to the box
   \begin{scope}
       \clip (\xmin,\ymin) rectangle (\xmax,\ymax);

       % ---- (C) axes ----
       \draw[thick] (\xmin,0) -- (\xmax,0);
       \draw[thick] (0,\ymin) -- (0,\ymax);

       % ---- (D) dashed parallel lines (two directions) ----
       % Lines parallel to b1, shifted by multiples of b2
       \foreach \k in {-6,-5,...,6}{
           % a point on the line: k*b2
           \pgfmathsetmacro{\px}{\k*\bbx}
           \pgfmathsetmacro{\py}{\k*\bby}
           % draw long segment in direction b1 through that point
           \draw[gray, dashed]
           ({\px-20*\bax},{\py-20*\bay}) -- ({\px+20*\bax},{\py+20*\bay});
       }
```

```
        % Lines parallel to b2, shifted by multiples of b1
        \foreach \k in {-6,-5,...,6}{
            % a point on the line: k*b1
            \pgfmathsetmacro{\qx}{\k*\bax}
            \pgfmathsetmacro{\qy}{\k*\bay}
            % draw long segment in direction b2 through that point
            \draw[gray, dashed]
            ({\qx-20*\bbx},{\qy-20*\bby}) -- ({\qx+20*\bbx},{\qy+20*\bby});
        }

        % ---- (E) lattice points: i*b1 + j*b2 ----
        \foreach \i in {-6,-5,...,6}{
            \foreach \j in {-6,-5,...,6}{
                \pgfmathsetmacro{\x}{\i*\bax+\j*\bbx}
                \pgfmathsetmacro{\y}{\i*\bay+\j*\bby}
                \fill (\x,\y) circle (2.2pt);
            }
        }

        % ---- (F) red circle and blue center point ----
        \draw[red, very thick] (0,0) circle (\R);
        \fill[blue] (0,0) circle (3pt);

    \end{scope}
\end{tikzpicture}
```



3.2.1 Step 1: Parameters you can easily change

This block defines variables:

- Window limits: `\xmin,\xmax,\ymin,\ymax`.
- Basis vector components:

$$b_1 = (\texttt{bax}, \texttt{bay}), \qquad b_2 = (\texttt{bbx}, \texttt{bby}).$$

- Circle radius `\R`.

3.2.2 Step 2: Border box and Clipping to the box

```
\draw[thick] (\xmin,\ymin) rectangle (\xmax,\ymax);
```

This draws a rectangle that frames the picture.

```
\clip (\xmin,\ymin) rectangle (\xmax,\ymax);
```

Clipping means: anything drawn outside the rectangle is not shown. This is useful because we will draw *very long* dashed lines; clipping keeps them neat.

3.2.3 Step 3: Axes

```
\draw[thick] (\xmin,0) -- (\xmax,0);    \draw[thick] (0,\ymin) -- (0,\ymax);
```

These draw the x- and y-axes across the window. (If you want arrowheads, use `[->,thick]`.)

3.2.4 Step 4: Dashed parallel lines (two families)

This is the key geometric idea.

Family 1: Lines parallel to b_1

For each integer k, we take a point kb_2:

$$kb_2 = (k\,\texttt{bbx},\, k\,\texttt{bby})$$

and draw a line through it in the direction of b_1.

In code:

- Compute a point on the line:

```
\px = k*\bbx,    \py = k*\bby
```

- Draw a long segment in direction b_1:

$$(p_x, p_y) \pm 20\, b_1$$

The number 20 is chosen so the segment is long enough to cover the window.

Family 2: Lines parallel to b_2

Similarly, for each integer k we take a point kb_1 and draw a line in direction b_2.

Why do we need two families? Together, these two families form a skew "grid" aligned with the lattice basis directions.

3.2.5 Step 5: Lattice points

This part draws the lattice:

$$(i, j) \mapsto ib_1 + jb_2.$$

We compute coordinates:

$$x = i\,\texttt{bax} + j\,\texttt{bbx}, \qquad y = i\,\texttt{bay} + j\,\texttt{bby}.$$

Then we plot a small filled circle.

Why use `\pgfmathsetmacro`**?** It stores computed numeric values (like `\x` and `\y`) so you can use them cleanly in `\fill (\x,\y)`.

3.2.6 Step 6: Circle and center point

```
\draw[red, very thick] (0,0) circle (\R);
```

draws a circle of radius `\R` centered at the origin.

```
\fill[blue] (0,0) circle (3pt);
```

draws a blue dot at the origin.

3.3 Drawing Rooted Trees

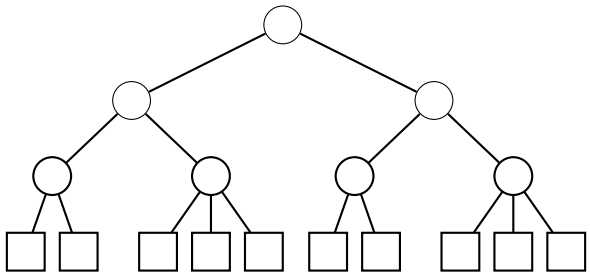
Learning Goals After this lesson, you will be able to:

- Draw a rooted tree using TikZ's `child` syntax.
- Style edges and nodes using TikZ styles.
- Use different node shapes (circles and squares).
- Control spacing with `level distance` and `sibling distance`.
- Read and modify a tree structure by matching indentation to hierarchy.

```
\begin{tikzpicture}[
    edge from parent/.style={draw, thick},
    every node/.style={draw, minimum size=5mm, inner sep=0pt},
    circ/.style={circle},
    sq/.style={rectangle},
    level distance=10mm,
    level 1/.style={sibling distance=40mm},
    level 2/.style={sibling distance=21mm},
    level 3/.style={sibling distance=7mm}
    ]
    \node[circ] {}
    child { node[circ] {}
        child { node[circ] {}
            child { node[sq] {} }
            child { node[sq] {} }
        }
        child { node[circ] {}
            child { node[sq] {} }
            child { node[sq] {} }
            child { node[sq] {} }
        }
    }
    child { node[circ] {}
        child { node[circ] {}
            child { node[sq] {} }
            child { node[sq] {} }
        }
        child { node[circ] {}
            child { node[sq] {} }
            child { node[sq] {} }
            child { node[sq] {} }
        }
    };
\end{tikzpicture}
```

TikZ can draw trees using the pattern:

```
\node {root} child { node {child1} } child { node {child2} };
```

Key keywords

- `\node ...;` creates the root node.
- `child { ... }` creates a child subtree.
- Every `child` contains another `node`, which becomes the child node.
- Indentation is not required, but it helps you see the structure.

3.3.1 Understanding the Style Block (The Part in [...])

At the top:

```
\begin{tikzpicture}[ ... ]
```

Everything inside `[ ... ]` sets default styles and spacing.

```
edge from parent/.style={draw, thick}
```

This means: every edge connecting a parent to a child is drawn as a thick line.

```
every node/.style={draw, minimum size=5mm, inner sep=0pt}
```

This sets defaults for all nodes:

- `draw` draws the node border.
- `minimum size=5mm` makes nodes at least 5mm in width/height.
- `inner sep=0pt` removes extra padding inside nodes.

```
circ/.style={circle}    sq/.style={rectangle}
```

These are **named styles** you can reuse:

- `node[circ]` makes a circular node.
- `node[sq]` makes a rectangular node.

3.3.2 Spacing Control (Preventing Overlap)

TikZ places nodes in levels (root at level 0, its children at level 1, etc.).

```
level distance=10mm
```

This is the distance between one level and the next (vertical gap).

```
level 1/.style={sibling distance=40mm}
level 2/.style={sibling distance=21mm}
level 3/.style={sibling distance=7mm}
```

- **sibling distance** controls how far apart children of the same parent are.
- We give a larger distance near the top (level 1) because the big left and right subtrees must not overlap.
- Lower levels can use smaller distances because their subtrees are narrower.

3.3.3 Reading the Tree Structure

Start with the root:

```
\node[circ] {}
```

This creates one circle (the root).

Then the root has **two children** because it has two `child {...}` blocks:

- First `child {...}` is the left subtree.
- Second `child {...}` is the right subtree.

Inside one subtree Example (left subtree):

```
child { node[circ] {}
    child { node[circ] { ... } }
    child { node[circ] { ... } }
}
```

That means:

- A circular node (level 1),
- with two circular children (level 2),
- each of which has several square leaves (level 3).

3.4 Building a One-Class SVM Figure with TikZ

Learning Goals After this lesson, students should be able to:

- Build a complex TikZ diagram using **structured blocks**.
- Use `\def` parameters to control the picture (window size, origin position, slope, margin).
- Draw **axes** with an **off-center origin**.
- Plot lines defined by an equation (hyperplane and margins).
- Place points using `\foreach` and basic coordinate arithmetic.
- Create annotated elements: arrows, labels, legend, and caption.

```
\begin{tikzpicture}[scale=0.95, line cap=round, line join=round]

   % ----------------------------
   % 0) Canvas box
   % ----------------------------
   \def\xmin{-7} \def\xmax{7}
   \def\ymin{-7} \def\ymax{7}
   \draw[thick] (\xmin,\ymin-2) rectangle (\xmax,\ymax);

   % ----------------------------
   % 1) Off-center origin (like the figure)
   % ----------------------------
   \def\Ox{-2.4}
   \def\Oy{-1.2}
   \coordinate (O) at (\Ox,\Oy);

   \begin{scope}
      \clip (\xmin,\ymin) rectangle (\xmax,\ymax);

      % ----------------------------
      % 2) Axes
      % ----------------------------
      \draw[blue!70, very thick, ->] (\xmin+0.5,\Oy) -- (\xmax-1.5,\Oy)
      node[right] {$\Phi(x_1)$};
      \draw[blue!70, very thick, ->] (\Ox,\ymin+0.5) -- (\Ox,\ymax-1)
      node[above] {$\Phi(x_2)$};

      \fill[blue!80] (O) circle (4pt);
      \node[below left] at (O) {Origin};

      % ----------------------------
      % 3) Hyperplane + two margins (parallel)
      % y - Oy = m(x - Ox) + c
      % ----------------------------
      \def\m{-1.05} % slope
      \def\c{2.0} % shift
      \def\dm{2} % margin spacing

      % solid hyperplane
```

```
    \draw[red, very thick]
    (\xmin,{ \Oy + \m*(\xmin-\Ox) + \c }) --
    (\xmax,{ \Oy + \m*(\xmax-\Ox) + \c })
    node[pos=0.75, rotate=-46, black, above] {Optimal hyperplane};

    % dashed margins
    \draw[red, dashed, thick]
    (\xmin,{ \Oy + \m*(\xmin-\Ox) + \c + \dm }) --
    (\xmax,{ \Oy + \m*(\xmax-\Ox) + \c + \dm });

    \draw[red, dashed, thick]
    (\xmin,{ \Oy + \m*(\xmin-\Ox) + \c - \dm }) --
    (\xmax,{ \Oy + \m*(\xmax-\Ox) + \c - \dm });

    \node[rotate=-46] at (\Ox-3.5,\Oy+6.0) {$\textbf{w}\cdot \textbf{x} + b = 0$};

    % ----------------------------
    % 4) Margin arrow
    % ----------------------------
    \draw[gray, very thick, <->] (\Ox-1,\Oy+5) -- (\Ox-2.9,\Oy+3.1)
    node[gray!50!black, sloped, above, midway, font=\bfseries] {Margin};

    % ----------------------------
    % 5) w arrow
    % ----------------------------
    \draw[blue!70, very thick, ->] (\Ox+.5,\Oy+1.5) -- (\Ox+1,\Oy+2)
    node[sloped, above, midway] {$\textbf{w}$};

    % ----------------------------
    % 6) Normal class points (black cluster)
    % ----------------------------
    \foreach \p in {
        (0.9,4.1),(0.1,3.3),(1.7,3.4),(1.2,2.7),(0.4,2.6),
        (2.2,2.5),(0.5,1.9),(1.6,1.7),(2.5,1.7),(1.1,1.2),
        (2.1,1.1),(0.6,0.9),(1.5,0.5),(2.3,0.5),
        (1.2,1.9),(1.9,2.0)
    }{
        \fill \p circle (4pt);
    }

    % ----------------------------
    % 7) Support vectors (circled points)
    % ----------------------------
    \foreach \p in {(\Ox+2,\Oy+1.9), (\Ox+1,\Oy+2.9), (\Ox+3,\Oy+.9)}{
        \fill \p circle (4pt);
        \draw \p circle (7pt);
    }

    % ----------------------------
    % 8) Samples violating constraint (gray balls) + xi labels
    % ----------------------------
    \shade[ball color=gray!60] (\Ox+1,\Oy+.25) circle (5pt) node[above=2pt] {$\xi_i$};
    \shade[ball color=gray!60] (\Ox+1.75,\Oy-.5) circle (5pt) node[below=2pt] {$\xi_j$};
\end{scope}

% ----------------------------
```

```
    % 9) Legend box
    % ----------------------------
    \begin{scope}[shift={(0,-8)}]
        \draw[blue!60, fill=white, thick, rounded corners=4pt] (-4.5,-.75) rectangle (4.5,1.0)
            ↪ ;

        \fill (-4,0.5) circle (5pt) node[right=4pt] {Normal class};
        \draw (-.5,.5) circle (5pt) node[right=4pt] {Support vector};
        \shade[ball color=gray!60] (-.5,-0.25) circle (5pt) node[right=4pt] {Sample violating
            ↪ constraint};
    \end{scope}

    % ----------------------------
    % 10) Caption (optional)
    % ----------------------------
    \node at (0,-9.5) {Fig.\ 2.\quad One-class Support Vector Machine};

\end{tikzpicture}
```

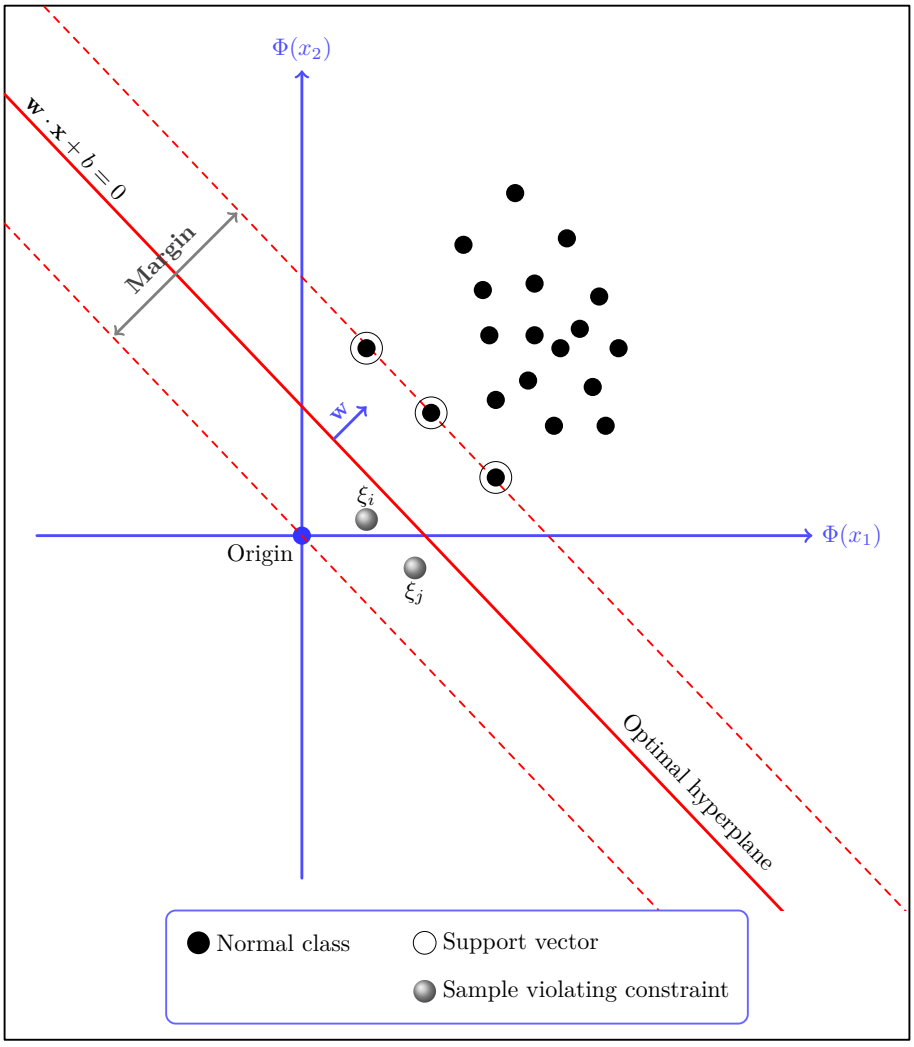


Fig. 2. One-class Support Vector Machine

3.4.1 Block 0: Canvas box (plot window)

- The macros `\xmin,\xmax,\ymin,\ymax` define the drawing window.
- The rectangle frames the figure:

```
\draw[thick] (\xmin,\ymin-2) rectangle (\xmax,\ymax);
```

Teaching tip: Ask students why `\ymin-2` is used (to make extra room for the legend/caption).

3.4.2 Block 1: Off-center origin

```
\def\Ox{-2.4} \def\Oy{-1.2}
\coordinate (O) at (\Ox,\Oy);
```

This is the most important design choice: the axes cross at (`\Ox,\Oy`) rather than $(0, 0)$, matching the reference figure.

3.4.3 Why we use `scope` + `clip`

```
\begin{scope} ... \clip ... \end{scope}
```

Everything inside this scope is clipped to the rectangle:

- prevents lines and points from spilling outside,
- keeps the final graphic clean.

3.4.4 Block 2: Axes with arrowheads and labels

Key idea: use $(\texttt{\textbackslash xmin}, \texttt{\textbackslash Oy})$ and $(\texttt{\textbackslash Ox}, \texttt{\textbackslash ymin})$ to align axes with the off-center origin.

```
(\xmin+0.5,\Oy) -- (\xmax-1.5,\Oy)
```

creates the horizontal axis across $y = \texttt{\textbackslash Oy}$.

```
(\Ox,\ymin+0.5) -- (\Ox,\ymax-1)
```

creates the vertical axis across $x = \texttt{\textbackslash Ox}$.

3.4.5 Block 3: Hyperplane and margins (equation-driven drawing)

The hyperplane is defined by the shifted line equation:

$$y - \texttt{\textbackslash Oy} = m(x - \texttt{\textbackslash Ox}) + c.$$

So if we pick a value of x, TikZ computes y by:

$$y = \texttt{\textbackslash Oy} + m(x - \texttt{\textbackslash Ox}) + c.$$

This is why the endpoints are written like:

```
(\xmin,{ \Oy + \m*(\xmin-\Ox) + \c })
```


Margins. The dashed margins are parallel lines obtained by shifting c:

$$c \to c + \mathrm{dm} \quad \text{and} \quad c \to c - \mathrm{dm}.$$

In code, that is:

`... + \dm` and `... - \dm`

3.4.6 Block 4: Margin arrow

```
\draw[gray, very thick, <->] (A) -- (B) node[...] {Margin};
```

- `<->` puts arrowheads on both ends.
- `sloped` rotates the label to match the arrow direction.
- `midway` puts the label in the middle.

3.4.7 Block 5: The w direction arrow

This arrow is a visual cue for the normal vector direction.

```
(\Ox+.5,\Oy+1.5) -- (\Ox+1,\Oy+2)
```

Notice: these coordinates are written relative to (`\Ox`, `\Oy`) so they move naturally if the origin moves.

3.4.8 Block 6: Plotting the normal class points

The code uses:

```
\foreach \p in { (x1,y1),(x2,y2),... }{ \fill \p circle (4pt); }
```

This is the easiest way to plot many points:

- Put point coordinates in the list.
- TikZ loops through them and plots each dot.

3.4.9 Block 7: Support vectors (circled points)

Support vectors are shown as a dot plus a surrounding circle:

`\fill \p circle (4pt);` and `\draw \p circle (7pt);`

3.4.10 Block 8: Violating samples with shading

The command:

```
\shade[ball color=gray!60] (x,y) circle (5pt);
```

creates a nice "3D ball" effect. Then a node attaches the label ξ_i or ξ_j.

3.4.11 Block 9: Legend box

The legend is drawn in a shifted scope:

```
\begin{scope}[shift={(0,-8)}] ... \end{scope}
```

This moves the legend down without rewriting every coordinate.

Inside the legend, three marker types are drawn using:

- `\fill ... circle` for normal class,
- `\draw ... circle` for support vectors (outline only),
- `\shade ... circle` for violating samples.

