Diagrams, Plots, and Code

STEM Track — Session 5 of 5

Daniel Dia

American University of Beirut

Fall 2026

Today

1. TikZ: coordinates, paths, nodes

Today

1. TikZ: coordinates, paths, nodes
2. How to do a real diagram with different styles and positions

Today

1. TikZ: coordinates, paths, nodes
2. How to do a real diagram with different styles and positions
3. `pgfplots`: functions and data files

Today

1. TikZ: coordinates, paths, nodes
2. How to do a real diagram with different styles and positions
3. `pgfplots`: functions and data files
4. `tikz-cd`

Today

1. TikZ: coordinates, paths, nodes
2. How to do a real diagram with different styles and positions
3. `pgfplots`: functions and data files
4. `tikz-cd`
5. Code: `listings` and pseudocode

SECTION 1

TikZ Basics

A drawing language, not a drawing tool

- You describe the picture: these points, connected so, labelled like this.

A drawing language, not a drawing tool

- You describe the picture: these points, connected so, labelled like this.
- LaTeX renders it in the document's own font and the document's own line weights so it scales properly with the rest of your document.

A drawing language, not a drawing tool

- You describe the picture: these points, connected so, labelled like this.
- LaTeX renders it in the document's own font and the document's own line weights so it scales properly with the rest of your document.
- This is why diagrams in well-made papers look like they were made with the text.

A drawing language, not a drawing tool

- You describe the picture: these points, connected so, labelled like this.
- LaTeX renders it in the document's own font and the document's own line weights so it scales properly with the rest of your document.
- This is why diagrams in well-made papers look like they were made with the text.
- Note that it is very uncommon to write TikZ from memory and most people just use pre-existing examples from `https://texample.net`

First strokes

TYPE ALONG

```
\usepackage{tikz}

\begin{tikzpicture}
    \draw (0,0) -- (2,1);                % line
    \draw (0,0) rectangle (3,2);         % rectangle
    \draw (1,1) circle (0.5);            % circle
    \draw[->, thick] (0,0) -- (2,0);     % arrow
\end{tikzpicture}
```

- Coordinates are `(x,y)`, centimetres by default.

First strokes

TYPE ALONG

```
\usepackage{tikz}

\begin{tikzpicture}
    \draw (0,0) -- (2,1);                % line
    \draw (0,0) rectangle (3,2);         % rectangle
    \draw (1,1) circle (0.5);            % circle
    \draw[->, thick] (0,0) -- (2,0);     % arrow
\end{tikzpicture}
```

- Coordinates are `(x,y)`, centimetres by default.
- Options in square brackets: `->`, `thick`, `dashed`, colours.

First strokes

TYPE ALONG

```
\usepackage{tikz}

\begin{tikzpicture}
    \draw (0,0) -- (2,1);                % line
    \draw (0,0) rectangle (3,2);         % rectangle
    \draw (1,1) circle (0.5);            % circle
    \draw[->, thick] (0,0) -- (2,0);     % arrow
\end{tikzpicture}
```

- Coordinates are `(x,y)`, centimetres by default.
- Options in square brackets: `->`, `thick`, `dashed`, colours.
- **Every path ends with a semicolon.**

Nodes

```
\begin{tikzpicture}
    \node[draw, rounded corners] (a) at (0,0) {Start};
    \node[draw, circle]          (b) at (3,0) {$q_1$};
    \draw[->] (a) -- (b) node[midway, above] {input};
\end{tikzpicture}
```

- A node is a labelled box at a certain position with it's name between parentheses

Nodes

```
\begin{tikzpicture}
    \node[draw, rounded corners] (a) at (0,0) {Start};
    \node[draw, circle]          (b) at (3,0) {$q_1$};
    \draw[->] (a) -- (b) node[midway, above] {input};
\end{tikzpicture}
```

- A node is a labelled box at a certain position with it's name between parentheses
- Once you name them they are connected via their names and not their coordinates.

Nodes

```
\begin{tikzpicture}
    \node[draw, rounded corners] (a) at (0,0) {Start};
    \node[draw, circle]          (b) at (3,0) {$q_1$};
    \draw[->] (a) -- (b) node[midway, above] {input};
\end{tikzpicture}
```

- A node is a labelled box at a certain position with it's name between parentheses
- Once you name them they are connected via their names and not their coordinates.
- TikZ draws edges to the node's border

SECTION 2

A Real Diagram

Relative positioning and styles

```
\usetikzlibrary{positioning, arrows.meta}

\begin{tikzpicture}[
    box/.style = {draw, rounded corners,
                  minimum width=2.2cm, align=center},
    arr/.style = {->, >=Stealth, thick},
    node distance = 1.2cm
]
    \node[box] (in) {Read input};
    \node[box, right=of in]   (proc) {Process};
    \node[box, right=of proc] (out)  {Write output};

    \draw[arr] (in) -- (proc);
    \draw[arr] (proc) -- (out);
\end{tikzpicture}
```

Why this scales

- `.style` definitions are the to TikZ what custom commands are to LaTeX. You can define all the common figures you're going to use early on, change them throughout the whole document by just changing the original definition. Basically, it offers all the same advantages as custom commands.

Why this scales

- `.style` definitions are the to TikZ what custom commands are to LaTeX. You can define all the common figures you're going to use early on, change them throughout the whole document by just changing the original definition. Basically, it offers all the same advantages as custom commands.
- `right=of in`: makes positions relative to each other instead of absolute coordinates to avoid having a meltdown when you want to change things down the line.

Why this scales

- `.style` definitions are the to TikZ what custom commands are to LaTeX. You can define all the common figures you're going to use early on, change them throughout the whole document by just changing the original definition. Basically, it offers all the same advantages as custom commands.
- `right=of in`: makes positions relative to each other instead of absolute coordinates to avoid having a meltdown when you want to change things down the line.
- Anchors let you pick a point on a node's border.

Applicable Across All Fields

- **CS.** Finite Automatons

Applicable Across All Fields

- **CS.** Finite Automatons
- **Physics.** Free-body diagrams

Applicable Across All Fields

- **CS.** Finite Automatons
- **Physics.** Free-body diagrams
- **Engineering.** Circuit schematics with `circuitikz`

Applicable Across All Fields

- **CS.** Finite Automatons
- **Physics.** Free-body diagrams
- **Engineering.** Circuit schematics with `circuitikz`
- **Everyone.** Flowcharts, block diagrams, and timelines.

SECTION 3

pgfplots

Plot a function

TYPE ALONG

```
\usepackage{pgfplots}
\pgfplotsset{compat=1.18}   % preamble, once

\begin{tikzpicture}
\begin{axis}[xlabel={$x$}, ylabel={$f(x)$},
             domain=-2:2, samples=100]
    \addplot[blue, thick] {x^2};
    \addplot[red, dashed] {exp(-x^2)};
    \legend{$x^2$, $e^{-x^2}$}
\end{axis}
\end{tikzpicture}
```

Plot a data file

```
% measurements.csv, uploaded to the project:
%   time,voltage
%   0.0,0.02
%   0.5,1.94

\begin{axis}[xlabel = {Time (\si{\second})},
             ylabel = {Voltage (\si{\volt})},
             grid = major]
    \addplot table[col sep=comma, x=time, y=voltage]
        {measurements.csv};
\end{axis}
```

Why this changes your workflow

- The plot reads the data file itself and uses it to give you the figure.

Why this changes your workflow

- The plot reads the data file itself and uses it to give you the figure.
- Eliminates the need to insert a screenshot of the .csv file or manually doing everything yourself.

Why this changes your workflow

- The plot reads the data file itself and uses it to give you the figure.
- Eliminates the need to insert a screenshot of the .csv file or manually doing everything yourself.
- If you're good at MatLab you can export it as a PDF.

SECTION 4

tikz-cd

Commutative diagrams

```
\usepackage{tikz-cd}

\begin{tikzcd}
    A \arrow[r, "f"] \arrow[d, "g"']
      & B \arrow[d, "h"] \\
    C \arrow[r, "k"']
      & D
\end{tikzcd}
```

- Arrows name a direction and take a label in quotes; `"g"'` flips the label to the other side.

SECTION 5

Code

Real code: `listings`

TYPE ALONG

```
\usepackage{listings}

\lstset{
    language = Python,
    basicstyle = \ttfamily\small,
    numbers = left,
    frame = single,
    showstringspaces = false,
}

% pull a file in directly:
\lstinputlisting[language=C]{solver.c}
```

Three things worth knowing

- `\lstinputlisting` reads your source file itself so there's no risk of copying the code from the wrong file by accident

Three things worth knowing

- `\lstinputlisting` reads your source file itself so there's no risk of copying the code from the wrong file by accident
- `showstringspaces=false`: kills the visible-space markers in strings.

Pseudocode: algorithm2e

```
\usepackage[ruled, vlined]{algorithm2e}

\begin{algorithm}
\caption{Binary search}
\KwIn{sorted array $A$, target $t$}
\KwOut{index of $t$, or $-1$}
$lo \gets 0$; $hi \gets |A| - 1$\;
\While{$lo \leq hi$}{
    $m \gets \lfloor (lo + hi)/2 \rfloor$\;
    \If{$A[m] = t$}{\Return $m$\;}
    \eIf{$A[m] < t$}{$lo \gets m + 1$\;}{$hi \gets m - 1$\;}
}
\Return $-1$\;
\end{algorithm}
```

Code vs pseudocode

- A listing shows an *implementation*. Pseudocode states an *algorithm*.

Code vs pseudocode

- A listing shows an *implementation*. Pseudocode states an *algorithm*.
- Reports and papers in CS and engineering usually want the latter.

Code vs pseudocode

- A listing shows an *implementation*. Pseudocode states an *algorithm*.
- Reports and papers in CS and engineering usually want the latter.
- The algorithm is a float so all the usual referencing rules apply.

SECTION 6

Exercise

Exercise

Three figures from your own work

1. The diagram you brought, rebuilt in TikZ: four or more nodes, `.style` definitions, relative positioning, labelled arrows.
2. A `pgfplots` figure from your `.csv`: axis labels with units, a grid, a caption, a label, a reference from the text.
3. One of: a listing of your function with line numbers, a pseudocode block of the same algorithm, or a commutative diagram from your current course.
4. Reference all three with `\cref`

SECTION 7

Homework

The STEM track problem set

Extend your final project with:

- a theorem–proof or definition–example pair;
- a multi-line derivation under one number;
- two custom operators;
- a TikZ diagram and a `pgfplots` figure from a data file;
- a listing, pseudocode block, or commutative diagram;
- `cleveref` throughout.

Hand in the `.tex` source, data files, and the compiled PDF.

Questions?

https://www.overleaf.com/learn • https://tex.stackexchange.com