

Drawing Block Diagrams in \LaTeX

Xu Chen
University of Washington

2014-12-14

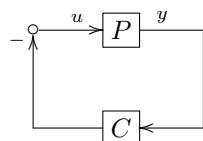
Prerequisites: basic \LaTeX commands and some LyX knowledge.

There are multiple ways to generate block diagrams in \LaTeX . Here are three most popular approaches.

xyplot using \xymatrix

This tool provides the most natural support for LyX.

Big picture: looking at the block diagram



you can see the following composition:

- two framed blocks \boxed{P} and \boxed{C} ,
- four points defining the edges of the rectangular diagram,
- arrows and lines connecting different elements,
- comments like the negative sign, u , and y .

When you draw the diagram on a piece of paper, you might use a ruled notebook for better alignment of the elements. \Xy -plot does the work in exactly the same way. The notebook it uses is called \Xy -matrix.

Basic commands:

- Outlining: \xymatrix generates a matrix where you can fill each entry with items you want (in my case, boxed transfer functions, signals, systems, etc)
- Making connections: \ar[pos] places an arrow to [pos], where pos can take the values of: l(eft), r(ight), d(own), u(p)
 - For instance, $\text{\[\xymatrix{B\ar[r] \& A}\]}$ generates a two by one \Xy -matrix with the entries of B and A . The command \ar[r] generates an arrow, starting from the first entry element B , pointing to the right, and ending on the second entry. Overall, the commands typesets

$$B \longrightarrow A$$

- Arrows can span multiple cells. For instance, \ar[rr] means “draw an arrow that points to the right and ends two entries away on the right”. As an example, $\text{\[\xymatrix{B\ar[rr] \& \& A}\]}$ gives

$$B \longrightarrow\longrightarrow A$$

- Framing a block: $\text{*[F]{contents}}$ puts contents into frames. The basic usage and common examples are summarized next.

– default frame: `\[\xymatrix{*[F]{A}}\]` gives



– Usually the default frame is too tight and must be widened by prefixing with `+` or `++`. Most commonly, I would use a command like `\[\xymatrix{*[++][F]{A}}\]`, which gives



– double-line frame: `\[\xymatrix{*[++][F=]{A}}\]` typesets



– circular frame: `\[\xymatrix{*[+o][F]{A}}\]` gives



Similar to the double-line rectangular frame, we can do `\[\xymatrix{*[++o][F=]{A}}\]`, to get



• Specifying the arrow formats:

– use `@{...}` after `\ar` to let L^AT_EX know the line format of the arrows

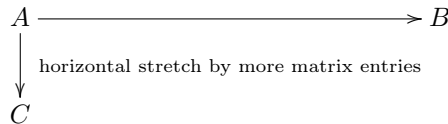
* Dashed arrows can be specified by `\ar@{-->}`. For instance, `\xymatrix{a\ar@{-->}[r]&b}` gives $a \text{---} > b$

* Dotted arrows can be specified by `\ar@{. . >}`. E.g., `\xymatrix{a\ar@{. . >}[r]&b}` gives $a \text{.....} > b$

* Invisible arrows: this is useful for labeling certain graphs. By typing `\ar@{}`, we can specify null format for the arrow. For instance,

```
\xymatrix{
A\ar[rrrrr]\ar[d] & & & & B \\
C\ar@{}[urrrrr] & \text{horizontal stretch by more matrix entries} & & & 
}
```

typesets



– use `@()` after `\ar` to specify the start and the end points of the arrow.

* For instance, `\[\xymatrix{A\ar@(d,u)[r]&B}\]` gives



The command `\ar@(d,u)[r]` reads: “draw an arrow to reach the next matrix entry on the right (say element B), starting downwards and finishing at the upper side of B .”

* We can do more fancy stuff such as:

```
\[\xymatrix{A\ar@(d,d)[r] & B}\],
\[\xymatrix{A\ar@(dr,ul)[r] & B}\],
and \[\xymatrix{A\ar@(d,ul)}\], which give, respectively,
```



and



- Tips for using $\mathbb{X}\mathbb{Y}$ -matrix in LyX's Math editor (press Ctrl-m or Ctrl-Shift-m and then type `\xymatrix`)
 - the braces: `[]` is the same, but `{}` should be entered by `\{`,¹ since by default LyX interprets `{}` as visible braces to display in math mode (e.g. $\{[A + (B^2 + A)]C\} + A$).
 - shortcuts:
 - * Adding a row in $\mathbb{X}\mathbb{Y}$ -matrix: Ctrl-Enter
 - * Adding a column in $\mathbb{X}\mathbb{Y}$ -matrix: Alt-m c i
 - * Horizontal and vertical scaling: How do we change the spacing between different elements in $\mathbb{X}\mathbb{Y}$ -matrix? In the tutorial “Using $\mathbb{X}\mathbb{Y}$ -pic in LyX,” H. Peter Gumm described using the following macros in the preamble of the LyX document (Layout▷Document▷Preamble):

```

\newcommand{\xyR}[1]{%
\xydef@\xymatrixrowsep@{#1}
} % end of \xyR
\newcommand{\xyC}[1]{%
\xydef@\xymatrixcolsep@{#1}
} % end of \xyC

```

Place the cursor inside the $\mathbb{X}\mathbb{Y}$ -matrix, just before the first entry. Then enter `\xyR\{` and/or `\xyC\{`, followed by the desired values of dimensions in the braces. The default spacing is 2pt. As an example, see the difference between

```

\xymatrix{
A\ar[r]\ar[d] & B\\
C\ar@{}[ur]|\{\text{\{default\}}\}
}

```

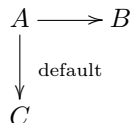
and

```

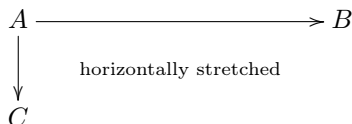
\xyC{9pc}\xyR{2pc}
\xymatrix{
A\ar[r]\ar[d] & B\\
C\ar@{}[ur]|\{\text{\{horizontal stretch\}}\}
}

```

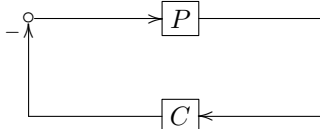
Results:



and



- Placing comments on arrows: In

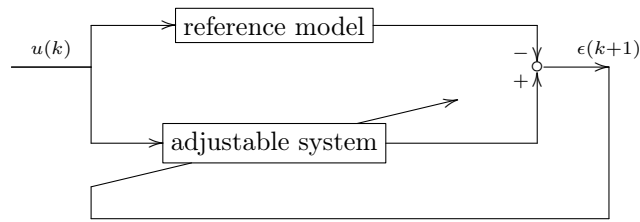


we want to put a negative sign near the end of the arrow on the left side. We can do so using `\ar[u]\sp(0.9){-}`, which reads “draw an arrow that points up, and has a superscript (sp) ‘-’ located at 0.9 of the full length of the arrow.”

It is pretty useful to use `>>` to let L^AT_EX figure out the spacing and automatically locate the arrow comments near the end location. For instance, `A \ar[r]\sp>>+ & B` typesets $A \xrightarrow{+} B$.

¹The closing brace will be automatically supplied by LyX.

- Arrows passing under an element: use `\ar'[ur]'` to go pass the upper right element and then reach the uurr element. For instance



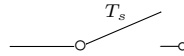
- Defining new arrow types: we can define a new arrow type "`|>`" by `\newdir{|>}{%`
`!/4.5pt/@{|}: (1, -.2)@^{|}: (1, +.2)@_{|}`
 and use it in arrow formats such as `\ar@{=|>}[r]` and `\ar@{-|>}[r]`, to get

$$A \Longrightarrow B$$

$$A \longrightarrow B$$

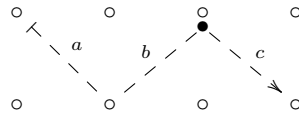
Other commands:

- Drawing a switch: XYPic supports also numerical axis locations. For instance, `\ar@{-}(21,5)*+{\sp{T_s}}` typesets a switch



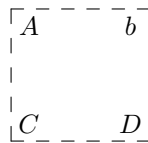
- Arrows passing under: just insert a `'` between the targets on the path

– example: `\xymatrix{{\circ}\ar@{-->'[dr] ^a '[rr]+D*{\bullet}^b [drrr] ^c & {\circ} & \\ {\circ} & {\circ} & \{\circ\}&{\circ\}&{\circ\}&{\circ\}}` typesets

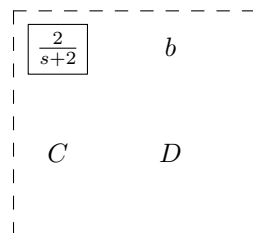


- Framing a group of objects:

`\[\xymatrix{A & b\save"1,1"."2,2"*[F--]\frm{}\restore\\ C & D } \]`



`\[\xymatrix{*+[F]{\frac{2}{s+2}} & b\save"1,1"."2,2"*++++[F--]\frm{}\restore\\ C & D } \]`



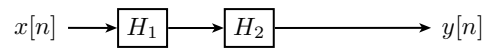
PSTricks and pst-sigsys

pst-sigsys stands for 'additional PSTricks for signal processing'. Below are some brief notes that hopefully illustrate the capabilities of the package. I hope to update more when I get the time.

- Remarks:
 - The lines and arrows generated by the package are more customizable compared to xyplot.
 - The package does not support pdflatex. To obtain the final pdf with correct graphs, one has to compile to dvi first then utilize the command ps2pdf.
- A basic example:

```
\begin{pspicture}(6,2)
\rput(0,1){\rnode{x}{x[n]}}
%\psfblock[framesize =0.75 0.5](2,1){a}{H_1$}
%\psfblock[framesize =1.5 1](4,1){b}{H_2$}
\psblock(1.5,1){a}{H_1$}
\psblock(3,1){b}{H_2$}
\rput(6,1){\rnode{y}{y[n]}}
%-----
\psset{style = Arrow }
\ncline[nodesepA = .15]{x}{a}
\ncline{a}{b}
\ncline[nodesepB = .15]{b}{y}
\end{pspicture}
```

typesets



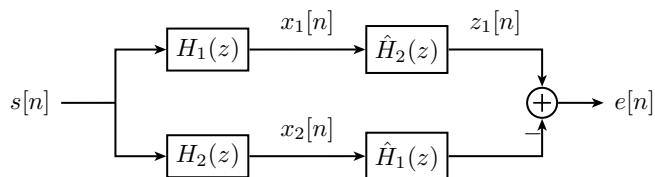
- More examples

```

\begin{figure}[ht]
\centering %
\begin{pspicture}[showgrid=false](0.5,-1.2)(9,1.55)
%--- Define blocks ---
\rput(0.5,0){\rnode{s}{s[n]}}
\dotnode[dotstyle=square*,dotscale=0.001](1.7,0){dot}
\psblock(3,.75){H1}{H_1(z)}
\psblock(3,-.75){H2}{H_2(z)}
\psblock(5.8,.75){B2}{\hat{H}_2(z)}
\psblock(5.8,-.75){B1}{\hat{H}_1(z)}
\pscicleop(7.7,0){ominus}
\rput(9,0){\rnode{e}{e[n]}}
%--- Connect blocks ---
\psset{style=Arrow}
\ncline[nodesepA=.15]{-}{s}{dot}
\ncangle[angleA=90,angleB=180]{dot}{H1}
\ncangle[angleA=-90,angleB=180]{dot}{H2}
\ncline{H1}{B2}
\naput[npos=.5]{x_1[n]}
\ncline{H2}{B1}
\naput[npos=.5]{x_2[n]}
\ncangle[angleB=90]{B2}{ominus}
\naput[npos=.5]{z_1[n]}
\ncangle[angleB=-90]{B1}{ominus}
\naput[npos=.9]{s-e}
\ncline[nodesepB=.15]{ominus}{e}
\end{pspicture}
\end{figure}

```

typesets

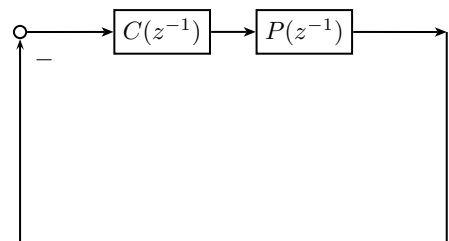


```

\begin{pspicture}(6,5)
\node(0,4){0.1}{s}
\pnode(6,4){e}
\psblock(2,4){c}{C(z^{-1})}
\psblock(4,4){p}{P(z^{-1})}
\pnode(6,1){br}
\pnode(0,1){bl}
\nclist[style = Arrow]{ncline}{s,c,p,e}
\ncline{-}{e}{br}
\ncline{-}{br}{bl}
\ncline{->}{bl}{s}
\nbput[npos=.9]{s-e}
\end{pspicture}

```

typesets



tikz

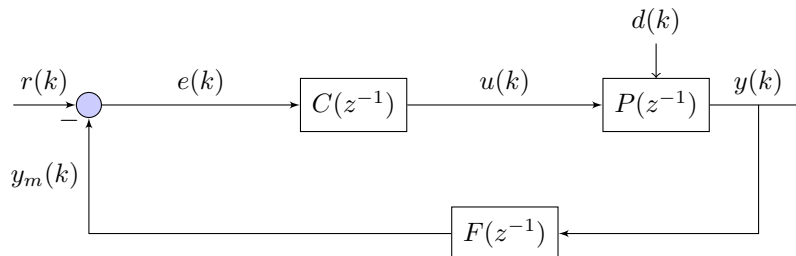
Compared to pst-sigsys, tikz is sometimes more convenient since it supports pdflatex directly. It even supports L^AT_X instant preview, so that we can see the drawings within L^AT_X before compiling them.

To use the package, we will need to add `\usepackage{tikz}` and `\usepackage{pgfplots}` at the beginning of the document. Also, it is very useful to define some basic formats at the beginning of the document:

```
\tikzstyle{block} = [draw, rectangle, minimum height=2em, minimum width=4em]
%fill=blue!20
\tikzstyle{sum} = [draw, fill=blue!20, circle, node distance=1cm]
\tikzstyle{input} = [coordinate] \tikzstyle{output} = [coordinate]
\tikzstyle{pinstyle} = [pin edge={to-,thin,black}]
```

The first example¹ uses relative location for each elements:

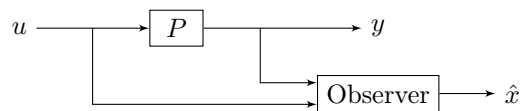
```
\begin{tikzpicture}[auto, node distance=2cm,>=latex']
\node [input, name=input] {};
\node [sum, right of=input] (sum) {};
\node [block, right of=sum, node distance=3.5cm] (controller) {$C(z^{-1})$};
\node [block, right of=controller, pin={[pinstyle]above:$d(k)$}, node distance=4cm] (system) {$P(z^{-1})$};
\draw [->] (controller) -- node[name=u] {$u(k)$} (system);
\node [output, right of=system] (output) {};
\node [block, below of=u] (measurements) {$F(z^{-1})$};
\draw [draw,->] (input) -- node {$r(k)$} (sum);
\draw [->] (sum) -- node {$e(k)$} (controller);
\draw [->] (system) -- node [name=y] {$y(k)$} (output);
\draw [->] (y) |- (measurements);
\draw [->] (measurements) -| node[pos=0.99] {$-$} node [near end] {$y_m(k)$} (sum);
\end{tikzpicture}
```



¹Modified from an example in <http://www.texample.net/tikz/examples/control-system-principles/>

It might be more preferred to construct a matrix and put different nodes there first:

```
\begin{tikzpicture}[auto, node distance=2cm, >=latex']
\matrix[column sep = .75cm, row sep = .375cm]
{
\node (u){$u$}; &
\node [coordinate](d1){}; &
\node [block](plant){$P$}; &
\node [coordinate](d2){}; &
\node (y){$y$}; &
\\
& & & \node [block](obs){Observer}; & \node (xhat){$\hat{x}$};
\\
};
\draw [->] (u) - (plant);
\draw [->] (plant) - (y);
\draw [->] (obs) - (xhat);
\draw [->] (d1) |- (obs.190);
\draw [->] (d2) |- (obs.170);
\end{tikzpicture}
```

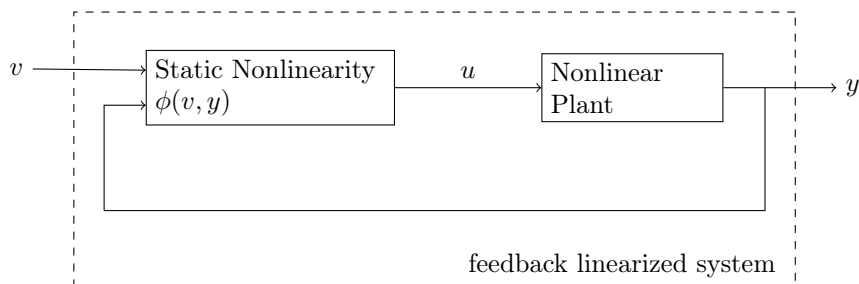


Comments can be very easily added by using relative locations:

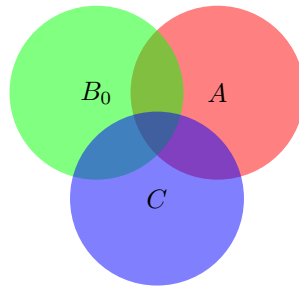
```

\begin{tikzpicture}
[
xscale = 1, % to scale horizontally everything but the text
yscale = 1, % to scale vertically everything but the text
]
% NODES DEFINITION
\matrix [ row sep = .375cm, column sep = .75cm, ]
{
% ----- row 1
\node (nInputv)[yshift = 0.25cm] {$v$}; &
\node (n22) [coordinate] {}; &
\node (nPhi) [block] {\parbox[c]{1.2in}{Static Nonlinearity \\\ $\phi(v,y)$}}; &
\node (nInputu)[above]{$u$}; &
\node (nSystem) [block] {\parbox[c]{.85in}{Nonlinear \\\ Plant}}; &
\node (n26) [coordinate, xshift = -0.2cm] {}; &
\node (nOutput) {$y$}; &
\\
& & & & & & &
\\
& & & & & & &
\\
% ----- row 2
\node (n11) [coordinate] {}; &
\node (n12) [coordinate, xshift = 0.2cm] {}; &
\node (n13) [coordinate] {}; &
\node (n14) [coordinate] {}; &
\node (n15) [coordinate] {}; &
\\
};
% ----- % PATHS
\draw [->] (nInputv) - (nPhi.172);
\draw [->] (nPhi) - (nSystem);
\draw [->] (nSystem) - (nOutput);
\draw [-] (n26) |- (n12);
\draw [->] (n12) |- (nPhi.188); %
% auxiliary nodes
\node [coordinate, xshift = 0.4cm, yshift = 1cm] (nAux1) at (n26) {};
\node [coordinate, xshift = -0.4cm, yshift = -1cm] (nAux2) at (n12) {}; %
\draw [dashed] (nAux1) -| (nAux2) -| (nAux1) node [above, pos = 0.38] {feedback linearized system};
\end{tikzpicture}

```

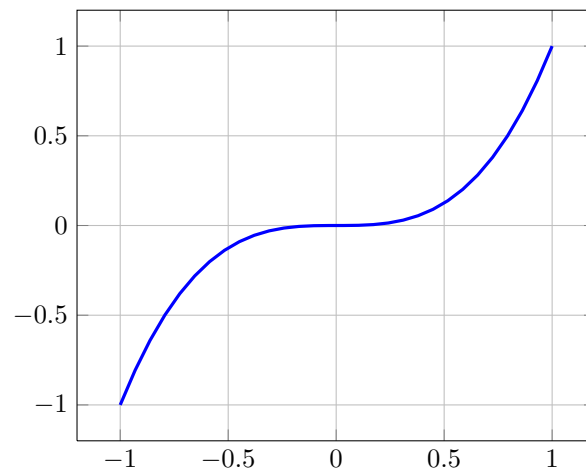


Tikz can also generate simple graphs such as



Several more detailed examples are provided next:

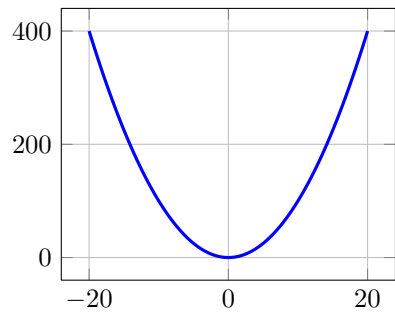
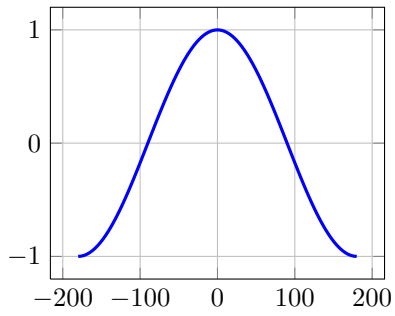
```
\begin{tikzpicture}
\begin{axis}
[grid=major,samples=30,mark=none]
\addplot[blue,very thick,domain=-1:1]
{x^3};
\end{axis}
\end{tikzpicture}
```



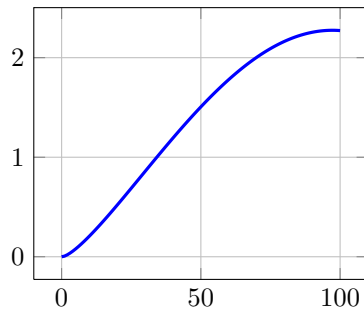
We can actually define a new function to avoid repeated writing the same codes:

```
\newcommand{\plotfun}[3][6cm]{
\begin{tikzpicture}
\begin{axis}
[width=#1,grid=major,samples=100,mark=none]
\addplot[blue,very thick,domain=#3]
{#2};
\end{axis}
\end{tikzpicture}
}
```

Using now $\backslash\text{plotfun}\{\cos(x)\}\{-180:180\}$ and $\backslash\text{plotfun}\{x^2\}\{-20:20\}$, we can get



We can combine different functions. for instance, `\plotfun{(sin(x)*ln(x+1))/2}{0:100}` typesets



We can even use `rnd` to generate random numbers. For instance, `\plotfun{rnd}{-20:20}` typesets

