PDIAG – A Package for Drawing Permutation Diagrams in LaTeX

Jason B. Hill

February 15, 2009

Instructions: Please try to compile this code on your machine after saving this file and the `pdiag.sty` file to the same directory. Then, send me the resulting dvi/pdf. Thanks!

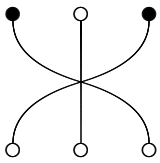
This isn't perfect yet, but it's getting better. Instead of using `xypic` to create permutation diagrams (not possible when live-TeXing), I wrote this small package to draw the diagrams from a code that can be written quickly. It has a few drawbacks right now.

- A significant benefit is that this package allows the user to modify the curves between points greatly. The curves are defined via a sequence of bezier lines, which are part of the standard LaTeX `picture` environment and hence themselves require no additional packages. However, LaTeX is incapable of doing floating point arithmetic to calculate the defining points of these beziers. The `xypic` package gets around this by doing all calculations via Postscript. Instead, I have loaded the `fp` package (standard in most LaTeX installations) to compute floating point operations directly. If your diagrams get large, this will require some processor time.
- If one is drawing a large diagram, there may be many vertices that are not permuted at any given step. No time should be wasted in entering the null mapping information for these vertices. This package is designed to draw the remaining lines that are not explicitly mapped. At present, the structure of the code means that this is limited to approximately 12 vertices. Long story. This problem can be fixed, but it will require some additional coding and I just don't have time right now. For now, 12 vertices will have to be enough.
- I've written code to label maps (rows), but have not yet written the code to label vertices.

Examples:

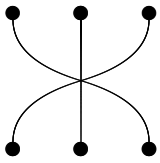
A simple example:

```
\[
\begin{pdiag}{3}{1}
\pdiagmap{1}{3}
\pdiagmap{3}{1}
\pdiagendmap
\end{pdiag}
\]
```



The `pdiag` environment takes 2 arguments: the number of vertices and the number of rows in the diagram. In the above example, there are 3 vertices and 1 row. The `pdiagmap` command takes 2 arguments: domain and image. In the first instance of `pdiagmap` above, the vertex 1 is sent to 3. Notice that only the vertices that are permuted under the map are filled (the rest are empty dots). You can change this by filling all dots in the diagram as follows: Place the command `\pdiagfill` anywhere within the environment.

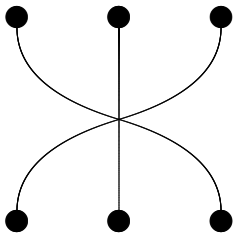
```
\[
\begin{pdiag}{3}{1}
\pdiagfill
\pdiagmap{1}{3}
\pdiagmap{3}{1}
\pdiagendmap
\end{pdiag}
\]
```



One may also scale the diagram with the optional [x] argument added to the `pdiag` environment. The default scale is 1. So, we have the two examples:

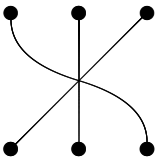
```
\[
\begin{pdiag}[0.5]{3}{1}
\pdiagfill
\pdiagmap{1}{3}
\pdiagmap{3}{1}
\pdiagendmap
\end{pdiag}
\]
```

```
\[
\begin{pdiag}[1.5]{3}{1}
\pdiagfill
\pdiagmap{1}{3}
\pdiagmap{3}{1}
\pdiagendmap
\end{pdiag}
\]
```



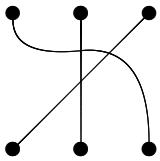
Beziers can be turned into straight lines by using the optional [0] argument inside the `pdiagmap` command as follows:

```
\[
\begin{pdiag}{3}{1}
\pdiagfill
\pdiagmap{1}{3}
\pdiagmap[0]{3}{1}
\pdiagendmap
\end{pdiag}
\]
```

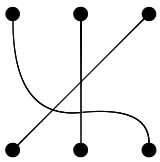


Beziers can be moved with a high degree of customization. To do so, initiate the command `pdiagbump` before the `pdiagmap` command. This command takes two arguments: the first controls the bezier as it enters the domain vertex, the second controls the bezier as it enters the image vertex. The domain and image correspond to the numbers -1 and 1, respectively. To push one side of the bezier towards the domain, set the corresponding argument closer to -1. To fix the situation above with the overlapping intersections of lines, we could use any of the following:

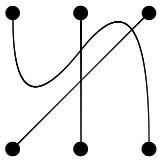
```
\[
\begin{pdiag}{3}{1}
\pdiagfill
\pdiagbump{-0.5}{-0.5}\pdiagmap{1}{3}
\pdiagmap[0]{3}{1}
\pdiagendmap
\end{pdiag}
\]
```



```
\[
\begin{pdiag}{3}{1}
\pdiagfill
\pdiagbump{0.5}{0.5}\pdiagmap{1}{3}
\pdiagmap[0]{3}{1}
\pdiagendmap
\end{pdiag}
\]
```

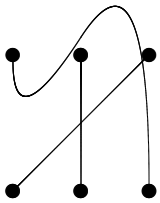


```
\[
\begin{pdiag}{3}{1}
\pdiagfill
\pdiagbump{0.8}{-1.8}\pdiagmap{1}{3}
\pdiagmap[0]{3}{1}
\pdiagendmap
\end{pdiag}
\]
```



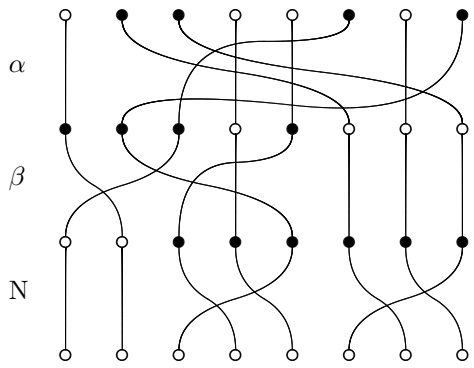
Notice that the values passed to the `pdiagbump` command do not need to stay within the range of -1 to 1. In fact, the bezier can be forced outside of the diagram as shown here:

```
\[
\begin{pdiag}{3}{1}
\pdiagfill
\pdiagbump{0.2}{-3}\pdiagmap{1}{3}
\pdiagmap[0]{3}{1}
\pdiagendmap
\end{pdiag}
\]
```



A larger example with labels for the rows (maps) is as follows. Notice that the `pdiagendmap` command finishes the current row (map), draws the remaining lines in the diagram and then moves to the next row.

```
\[
\begin{pdiag}[1.5]{8}{3}
\pdiagmap{2}{6}
\pdiagmap{3}{8}
\pdiagbump{-0.7}{-0.5}\pdiagmap{6}{3}
\pdiagbump{0.9}{0.4}\pdiagmap{8}{2}
\pdiagname{$\alpha$}\pdiagendmap
\pdiagmap{1}{2}
\pdiagmap{2}{5}
\pdiagbump{-0.6}{-0.3}\pdiagmap{5}{3}
\pdiagmap{3}{1}
\pdiagname{$\beta$}\pdiagendmap
\pdiagmap{4}{5}
\pdiagmap{5}{3}
\pdiagmap{3}{4}
\pdiagmap{8}{6}
\pdiagmap{7}{8}
\pdiagmap{6}{7}
\pdiagname{N}\pdiagendmap
\end{pdiag}
\]
```



This environment can be placed anywhere where an array may be placed.

```
\[
\sum_{i=1}^n\left\{\hspace{-1pc}
\begin{pdiag}{2}{1}
\pdiagendmap
\end{pdiag}\hspace{-1pc}\cdots\hspace{-1pc}
\begin{pdiag}{2}{1}
\pdiagfill
\pdiagmap[0]{1}{2}
\pdiagmap[0]{2}{1}
\end{pdiag}\hspace{-1pc}\cdots\hspace{-1pc}
\begin{pdiag}{2}{1}
\pdiagendmap
\end{pdiag}\hspace{-1pc}
\right\}
\]
```

$$\sum_{i=1}^{n}\left\{\hspace{-1pc}\begin{pdiag}{2}{1}\pdiagendmap\end{pdiag}\hspace{-1pc}\cdots\hspace{-1pc}\begin{pdiag}{2}{1}\pdiagfill\pdiagmap[0]{1}{2}\pdiagmap[0]{2}{1}\end{pdiag}\hspace{-1pc}\cdots\hspace{-1pc}\begin{pdiag}{2}{1}\pdiagendmap\end{pdiag}\hspace{-1pc}\right\}$$