SPECTRALSEQUENCES

Hood Chatham
hood@mit.edu

Version 1.2.0
2018/4/13

The SPECTRALSEQUENCES package is a specialized tool built on top of PGF/Ti*k*Z for drawing spectral sequence charts. It provides a powerful, concise syntax for specifying the data of a spectral sequence, and then allows the user to print various pages of a spectral sequence, automatically choosing which subset of the classes, differentials, and structure lines to display on each page. It also handles most of the details of the layout. At the same time, SPECTRALSEQUENCES is extremely flexible. It is closely integrated with Ti*k*Z to ensure that users can take advantage of as much as possible of its expressive power. It is possible to turn off most of the automated layout features and draw replacements using Ti*k*Z commands. SPECTRALSEQUENCES also has a carefully designed error reporting system intended to ensure that it is as clear as possible what is going wrong.

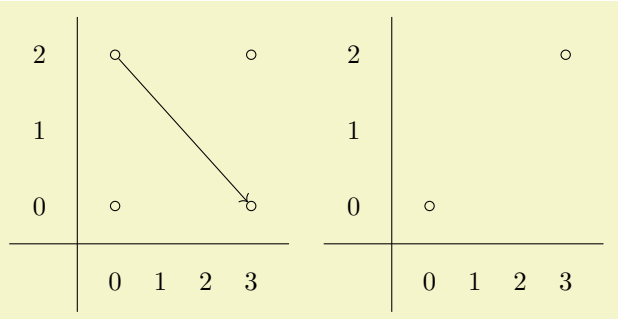
Many thanks to the authors of Ti*k*Z for producing such a wonderful package with such thorough documentation. I would have needed to spend a lot more time reading the Ti*k*Z code if the documentation weren't so excellent. I took ideas or code or both from tikzcd (part of the code for turning quotes into class or edge labels), PGFPLOTS (axes labels), and sseq (the grid types, the stack). I lifted a fair amount of code from TEXstack exchange. Thanks to Eva Belmont for tons of helpful suggestions, bug reports, and productive conversations. Talking to her has helped to clarify many design concepts for the package. Thanks to Eric Peterson for being a very early adopter and reporting many bugs. Also thanks to all my friends, family, and acquaintances listened to me talk about LATEX programming even though they probably found it dreadfully boring.

Contents

1 Introduction

The SPECTRALSEQUENCES package consists of two main environments – the `{sseqdata}` environment, which specifies the data for a named spectral sequence, and the `{sseqpage}` environment, which prints a single page of a spectral sequence. The `\printpage` command is also available as a synonym for a `{sseqpage}` environment with an empty body.

Here is a basic example:




```
\begin{sseqdata}[ name = basic, xscale = 0.6,
                  cohomological Serre grading ]
\class(0,0)
\class(0,2)
\class(3,0)
\class(3,2)
\d3(0,2)
\end{sseqdata}
\printpage[ name = basic, page = 3 ]  \quad
\printpage[ name = basic, page = 4 ]
```

`\begin{sseqdata}[name = basic, cohomological Serre grading]` starts the declaration of the data of a spectral sequence named `basic` with cohomological Serre grading – that is, the page `r` differentials go `r` to the right and down $r-1$. Then we specify four classes and one page 3 differential, and we ask SPECTRALSEQUENCES to print the third and fourth pages of the spectral sequence. Note that on the fourth page, the source and target of the differential have disappeared.

1.1 Installation

In both MiKTEX and TEX Live installation should be automatic – your TEX distribution should automatically install the package the first time you include `\usepackage{spectralsequences}` in a document and compile it. However, in 2016, TEX Live made an incompatible change to their database, so no new packages will run on versions of TEX Live from before 2016. This includes SPECTRALSEQUENCES. If you have an old version of TEX Live, you can either perform a manual install, or, better, you should install an up to date version of TEX Live. If you want to do a manual install, see this TEXstack exchange post for instructions.

1.2 Memory Constraints

In a default TEX install, PDFLATEX has small static memory caps that prevent it from using more than about 60 megabytes of total ram. However, SPECTRALSEQUENCES and PGF/Ti*k*Z use a large amount of memory. For this reason, using PDFLATEX with a default install, you cannot draw more than about 2500 classes across all of your diagrams (fewer if you include differentials, structure lines, and other features). There are a few solutions to this.

The easiest solution is to run LuaLATEX. LuaLATEX dynamically allocates memory and so is unlikely to run out of it. Using LuaLATEX on my computer, I can compile a document that draws two copies of a diagram with 20,000 classes in it (so a total of 40,000 classes). This takes about 50 seconds and 250 megabytes of ram. I expect any real-world use case will compile fine on a modern computer using LuaLATEX. This option has the advantage that any modern TEX install comes with a copy of LuaLATEX, and that LuaLATEX is the designated successor to PDFLATEX. It has the disadvantage that there are some incompatibilities between LuaLATEX and PDFLATEX so if your document depends on PDFLATEX-specific features, it might be a pain to switch to LuaLATEX.

Another option is to increase the static memory caps for PDFLATEX. See this TEXstack exchange post for instructions on how to do this.