The `spectralsequences` package is a specialized tool built on top of PGF/TikZ 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 TikZ 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 TikZ 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 TikZ for producing such a wonderful package with such thorough documentation. I would have needed to spend a lot more time reading the TikZ 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 `TeX` stack 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

1.1 Installation ........................................ 3
1.2 Memory Constraints ................................. 3
1.3 A warning about fragile macros .................. 4

2 Package Options and Environments .................. 4

3 The Main Commands ..................................... 4

4 Options for the main commands ....................... 9

4.1 Universal options ................................... 9
4.2 Options for `\class` ................................. 12
4.3 Options for `\d` and `\structline` ................. 17
4.4 Options for `\circleclass` ......................... 19
4.5 Options for Ti\textit{KZ} primitives ................. 20

5 Miscellaneous Commands ............................... 23

5.1 Settings ............................................. 23
5.2 Code reuse commands ............................... 24
5.3 Families ............................................. 27
5.4 Utilities ............................................. 28
5.5 Coordinate parsers and related ................... 30
5.6 The class stack .................................... 32

6 Styles .................................................. 33

6.1 Style-like options .................................. 37

7 Global Options .......................................... 39

7.1 Global coordinate transformations ................. 45
7.2 Plot options and axes style ....................... 45
7.3 Layout .............................................. 49
\section{Introduction}

The \texttt{spectralsequences} package consists of two main environments – the \texttt{\{sseqdata\}} environment, which specifies the data for a named spectral sequence, and the \texttt{\{sseqpage\}} environment, which prints a single page of a spectral sequence. The \texttt{\textbackslash printpage} command is also available as a synonym for a \texttt{\{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]
\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 \texttt{basic} with cohomological Serre grading – that is, the page $r$ differentials go to the right and down $r - 1$. Then we specify four classes and one page 3 differential, and we ask \texttt{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.

\subsection{Installation}

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

\subsection{Memory Constraints}

In a default \texttt{TeX} install, PDF\texttt{LaTeX} has small static memory caps that prevent it from using more than about 60 megabytes of total ram. However, \texttt{spectralsequences} and \texttt{PGF/Ti\texttt{K}Z} use a large amount of memory. For this reason, using PDF\texttt{LaTeX} 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 Lua\texttt{LaTeX}. Lua\texttt{LaTeX} dynamically allocates memory and so is unlikely to run out of it. Using Lua\texttt{LaTeX} 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 Lua\texttt{LaTeX}. This option has the advantage that any modern \texttt{TeX} install comes with a copy of Lua\texttt{LaTeX}, and that Lua\texttt{LaTeX} is the designated successor to PDF\texttt{LaTeX}. It has the disadvantage that there are some incompatibilities between Lua\texttt{LaTeX} and PDF\texttt{LaTeX} so if your document depends on PDF\texttt{LaTeX}-specific features, it might be a pain to switch to Lua\texttt{LaTeX}.

Another option is to increase the static memory caps for PDF\texttt{LaTeX}. See this \texttt{TeX} stack exchange post for instructions on how to do this.
1.3 A warning about fragile macros

All the data in a SPECTRALSEQUENCES environment is stored and used later. As a result, most of the SPECTRALSEQUENCES commands currently cannot tolerate fragile macros. Unfortunately, it is impossible for SPECTRALSEQUENCES to warn you about this situation – if you use a fragile command in a place that it doesn’t belong, the result will be an incomprehensible error message. If you are getting nonsense error messages, this might be why. The solution is to convert fragile macros into robust ones. Common examples of fragile macros include \widehat and \underline. My suggested solution to this is to add the following code to your preamble for each fragile macro (example given for \mathbb):

\let\oldwidehat\widehat
\protected\def\widehat{\oldwidehat}

2 Package Options and Environments

Draft Mode

The drawings that SPECTRALSEQUENCES produces can be quite slow, especially if they are large. Draft mode skips drawing the content of the spectral sequence, but still takes up exactly the same amount of space in the document, so that you can deal with formatting issues. To active draft mode, load the package by saying \usepackage[draft]{spectralsequences}.

\begin{sseqdata}[(options)]
⟨environment contents⟩
\end{sseqdata}

The \{sseqdata\} environment is for storing a spectral sequence to be printed later. This environment is intended for circumstances where you want to print multiple pages of the same spectral sequence. When using the \{sseqdata\} environment, you must use the name option to tell SPECTRALSEQUENCES where to store the spectral sequence so that you can access it later.

\begin{sseqpage}[(options)]
⟨environment contents⟩
\end{sseqpage}

This environment is used for printing a page of existing spectral sequence that was already specified using the \{sseqdata\} environment. The body of the environment adds local changes – classes, differentials, structure lines, and arbitrary TikZ options that are by default only printed on this particular page. The \{sseqpage\} environment can also be used to print a stand-alone page of a spectral sequence – that is, if you only want to print a single page of the spectral sequence, you can skip using the \{sseqdata\} environment.

\printpage[(options)]

This command prints a single page of an existing spectral sequence as-is. This is equivalent to a \{sseqpage\} environment with an empty body.

3 The Main Commands

\class[(options)]((x),(y))

This places a class at (x,y) where x and y are integers. If multiple classes occur at the same position, SPECTRALSEQUENCES will automatically arrange them in a pre-specified pattern. This pattern may be altered using the class pattern option.
The effect of the \texttt{class} command is to print a Ti\textit{k}Z node on a range of pages. Any option that would work for a Ti\textit{k}Z \texttt{node} command will also work in the same way for the \texttt{class}, \texttt{replaceclass}, and \texttt{classoptions} commands.

If a class is the source or the target of a differential on a certain page, then the page of the class is set to that page, and the class is only rendered on pages up to that number:

\begin{sequdata}
\begin{align*}
3 & \quad 0 & 1 & 0 & 1 \\
2 & \quad 0 & 1 & 0 & 1 \\
1 & \quad 0 & 1 & 0 & 1 \\
0 & \quad 0 & 1 & 0 & 1 \\
\end{align*}
\end{sequdata}

\texttt{printpage[ name = class example, page = 2 ]}
\quad 
\texttt{printpage[ name = class example, page = 3 ]}

See the class options section for a list of the sort of options available for classes.

\begin{align*}
\texttt{replaceclass[(\texttt{options})]\{(x),(y),(n)\}} & \\
\texttt{replaceclass[(\texttt{options})]\{\texttt{classname}\}} & \\
\texttt{replacesource[(\texttt{options})]} & \\
\texttt{replacetarget[(\texttt{options})]} & \\
\end{align*}

After a class is the source or target of a differential, it disappears on the next page. However, some differentials are not injective or not surjective. Using the command \texttt{replaceclass} causes a new symbol to appear on the page after a class supported or accepted a differential (or both). If there are multiple classes at the coordinate \( (x, y) \) you may specify which using an integer or a tag \texttt{n}. By default, this command will affect the first class placed in that position. You can also provide the \texttt{name} of a class. The variants \texttt{replacesource} and \texttt{replacetarget} replace the source and target respectively of the most recent differential.

\begin{align*}
\texttt{replaceclass[(\texttt{options})]\{(x),(y),(n)\}} & \\
\texttt{replaceclass[(\texttt{options})]\{\texttt{classname}\}} & \\
\texttt{replacesource[(\texttt{options})]} & \\
\texttt{replacetarget[(\texttt{options})]} & \\
\end{align*}
\begin{sseqdata}[\text{name = replace class example, Adams grading, classes = {draw = none}}]
\class["\mathbb{Z}\)'](0,3)
\class["\mathbb{Z}\)'](1,1)
\class["\mathbb{Z}\)'](1,0)
\d[\cdot 2]\cdot 2(1,1)
\replacetarget["\mathbb{Z}/2"]% \replaceclass["\mathbb{Z}/2"](0,3)
\d[-\to]3(1,0)
\replacesource["2\mathbb{Z}\)"]% \replaceclass["2\mathbb{Z}\)"](1,0)
\end{sseqdata}
\begin{printpage}[\text{name = replace class example, page = 2}]
\node[background] at (0.3,-2.2) \text{The red class is the problem};
\end{printpage}
\begin{printpage}[\text{name = replace class example, page = 3}]
\begin{printpage}[\text{name = replace class example, page = 4}]

Note that this will not restore any structure lines coming into or off of the class, if you want the structure lines to persist, you must call \text{\structline} again (or use the page option).

Another reason to use this is to give a label to one instance of a class that shows up in a loop or a command defined using \text{\NewSseqGroup}:

See the class options section for a list of the sort of options available for classes.

\d[(\text{options})](\text{page})
\d[(\text{options})](\text{page})\,(\text{\langle x\rangle,\langle y\rangle,\langle source n\rangle,\langle target n\rangle})
Calling \texttt{\textbackslash d(page)}\langle (x),(y) \rangle creates a differential starting at \langle (x),(y) \rangle of length determined by the specified page. In order to use the \texttt{\textbackslash d} command like this, you must first specify the degree of the differentials as an option to the \texttt{\textbackslash sseqdata} or \texttt{\textbackslash sseqpage} environment. The degree indicates how far to the right and how far up a page \( r \) differential will go as a function of \( r \). If there is a page \( r \) differential, on page \( r+1 \), the source, target, and any structure lines connected to the source and target of the differential disappear. If no class is specified, the default is to use \texttt{\textbackslash lastclass}.

If there are multiple nodes in the source or target, you may specify which one the differential should go to using an index or tag for (source \texttt{n}) or (target \texttt{n}). It is also possible to provide the name of the source coordinate and an optional target, or to separately provide the source and target coordinate, either as names or as \langle (x),(y),(n) \rangle. Using \texttt{\textbackslash d} with explicit source and target coordinates works even if you did not provide a degree to the spectral sequence. If you did provide a degree, then \texttt{\textsc{spectralsequences}} will check whether the difference between the source and target is appropriate for a differential of a given page, and if not it will throw an error. If this is undesirable, you can use the \texttt{lax degree} option.

If there are multiple nodes in the source or target coordinate, then there is a funny syntax for indicating which one should be the source and target: \texttt{\textbackslash d(page)}\langle (x),(y),\textbackslashtexttt{source n},\texttt{target n} \rangle

Negative indices will count from the most recent class in the coordinate (so the most recent is \(-1\), the second most recent is \(-2\), etc). You can also use a \texttt{tag}, which works better if the situation is complicated.
\texttt{\textbackslash options[⟨options⟩]}\langle page\rangle\langle(source coordinate)\rangle\langle(target coordinate)\rangle

This command adds options to an existing differential, just like \texttt{\textbackslash classoptions} except for differentials. Its syntax is identical to that of \texttt{\textbackslash d}.

\texttt{\textbackslash kill[⟨page⟩][⟨coord⟩]}

This command sets the indicated coordinate to die on the indicated page, but does not establish a target for the differential. This is useful if you want to draw your own differential using \texttt{tikz} (see \texttt{\textbackslash getdtarget}) or if you are not drawing the class on the other side of the differential for clutter reasons. As usual, if no coordinate is provided, the default argument is \texttt{\lastclass}.

\texttt{\textbackslash structline[⟨options⟩]}\langle(source coordinate)\rangle\langle(target coordinate)\rangle

This command creates a structure line from \langle source coordinate \rangle to \langle target coordinate \rangle. The source and target coordinates are either of the form \langle(x,y),(n)\rangle or \langle(class name)\rangle. If there are multiple classes at \langle(x,y)\rangle, then \langle(n)\rangle specifies which of the classes at \langle(x,y)\rangle the structure line starts and ends at – if n is positive, then it counts from the first class in that position, if n is negative, it counts backwards from the most recent. You can also use a \texttt{tag} for n. If the \langle target coordinate \rangle is omitted, then \texttt{\lastclass} and \texttt{\lastclass1} are used, and so \texttt{\structline} with no arguments at all will connect the two most recent classes.

If the source or target of a structure line is hit by a differential, then on subsequent pages, the structure line disappears.

If the source or target has had multiple generations (i.e., they got hit and you used \texttt{\replaceclass}), then the \texttt{\structline} will only appear starting on the first page where the current generation of both the source and target are present. If this is undesirable, you can use the \texttt{page} option to change it.

\texttt{\begin{seqgroup}\tower () \{ \class(0,0) \foreach \y in {1,...,5} { \class(0,\y) \structline } \class(0,2) \structline(0,1,-1) \structline(0,3,-1) \} \begin{sseqdata}\[ name = structline example, \ classes = \{ circle, fill \}, \ Adams grading, no axes, \ yscale = 1.28 \] \class(1,1) \class(1,2) \class(2,3) \class(2,5) \tower[classes = blue](0,0) \tower[struct lines = dashed, orange](1,0) \tower[struct lines = red](2,0) \d2(1,1,2) \end{sseqdata} \printpage[ name = structline example, page = 2 ] \quad \printpage[ name = structline example, page = 3 ] \end{seqgroup}}

\texttt{\textbackslash structlineoptions[⟨options⟩]}\langle(source coordinate)\rangle\langle(target coordinate)\rangle

This command adds options to an existing structure line, just like \texttt{\textbackslash classoptions} except for structure lines. Its syntax is identical to \texttt{\structline}.

\texttt{\textbackslash circleclasses[⟨options⟩]}\langle(source coordinate)\rangle\langle(target coordinate)\rangle

This command is a lot like \texttt{\structline} except that it puts a circle around the classes instead of connecting them with a line. It might take a certain amount of fiddling with options to get \texttt{\circleclasses} to produce good results. There is no \texttt{\circleclassesoptions} command because it doesn’t seem necessary and (more importantly) I didn’t feel like making one. Maybe someday I’ll put one in.
Any code that would work in a \texttt{tikzpicture} environment will also work unchanged in a \texttt{sseqdata} or \texttt{sseqpage} environment, with a few minor differences. This is a very flexible way to add arbitrary background or foreground features to the spectral sequence:

\begin{sseqdata}
\[\text{name} = \text{tikz example}, \text{Adams grading}, \text{math nodes} = \text{false}, \\
\text{tikz primitives} = \{ \text{blue, font = \text{\tiny, \rightarrow}} \}, \text{circle classes} = \text{tikz primitive style}, \\
\text{x range} = \{0\}(2), \text{x axis extend end} = 2\text{em}\]
\begin{itemize}
\item \texttt{\class}(0,0)
\item \texttt{\class}(1,0)
\item \texttt{\class}(0,2)
\item \texttt{\d}(1,0)
\end{itemize}
\end{sseqdata}

\begin{sseqpage}
\[\text{name} = \text{tikz example}\]
\begin{itemize}
\item \texttt{\circleclasses} [ \text{name path} = \text{myellipse}, \text{inner sep} = 3\text{pt}, \text{ellipse ratio} = 1.6 ] (1,0)(0,2)
\item \texttt{\path} [ \text{name intersections} = \{ \text{of = myellipse and myline} \} ] (intersection-1) to (1.3,1.25) node[ \text{right, text width = 1.6cm} ] {Consider this differential};
\end{itemize}
\end{sseqpage}

\begin{sseqpage}
\[\text{name} = \text{tikz example}\]
\begin{itemize}
\item \texttt{\draw} [ \text{xshift} = 1 ] (0,0) to (0.6,0.2) node[ \text{right, text width = 1.1cm} ] {This is the source};
\item \texttt{\draw} [ \text{yshift} = 2 ] (0,0) to (0.6,0.2) node[ \text{right, text width = 1.1cm} ] {This is the target};
\end{itemize}
\end{sseqpage}

\begin{sseqpage}
\[\text{page} = 3, \text{name} = \text{tikz example}\]
\begin{itemize}
\item \texttt{\circleclasses} [ \text{inner sep} = 3\text{pt}, \text{ellipse ratio} = 1.6 ] (1,0)(0,2)
\item \texttt{\node} [ \text{right, font = \text{\tiny}} ] at (1.2,1.22) {Now it’s gone!};
\end{itemize}
\end{sseqpage}

4 Options for the main commands

4.1 Universal options

The following options work with all of the drawing commands in this package, including \texttt{\class}, \texttt{\d}, and \texttt{\structline}, their friends \texttt{\replaceclass}, \texttt{\classoptions}, \texttt{\doptions}, and \texttt{\structlines}, as well as with TikZ primitives.

\texttt{\xshift = \{integer\}}
\texttt{\yshift = \{integer\}}

Shifts by integer values are the only coordinate changes that are allowed to be applied to \texttt{\class}, \texttt{\d}, \texttt{\structline}, their relatives, or to a \texttt{\scope} environment that contains any of these commands. These shift commands help with reusing code. For instance:
This code segment is very useful so SPECTRALSEQUENCES has the command \NewSseqGroup which to make code like this more convenient. The following code produces the same output as above:

\begin{sseqpage}[cohomological Serre grading, yscale = 0.45]
\foreach \x in {0,1} \foreach \y in {0,1} {
  \begin{scope}[xshift = \x, yshift = \y]
    \class(2,0)
    \class(0,1)
    \d2(0,1)
  \end{scope}
}\end{sseqpage}

A word of warning: the behavior of xshift in SPECTRALSEQUENCES is incompatible with the normal behavior of xshift in TikZ. For some reason, saying xshift = 1 in TikZ does not shift the coordinate (0,0) to the coordinate (1,0) – instead it shifts by 1pt. In SPECTRALSEQUENCES, saying xshift = 1 moves the coordinate (0,0) to the coordinate (1,0). This includes TikZ primitives: saying \draw[ xshift = 1 ] (0,0) -- (1,0); inside a \{sseqdata\} or \{sseqpage\} environment is the same as saying \draw(1,0) -- (2,0); despite the fact that this is not the case in the \{tikzpicture\} environment.

Colors
These come from the \LaTeX color package via TikZ, so see the color package documentation for more information.

"\text"(options)
Specify a label for a class, a differential, or a structure line. This uses the TikZ quotes syntax. If the label text includes an equal sign or comma, you need to enclose the entire label in braces, e.g., \class["x = y"](0,0). The options include anything you might pass as an option to a TikZ node, including arbitrary coordinate transforms, colors, opacity options, shapes, fill, draw, etc. The behavior is a little different depending on whether you use it on a class or on a differential or structure line.

For a class, the \text is placed in the position inside the node by default – in effect, the \text becomes the label text of the node (so saying \class[\"label text"](0,0) causes a similar effect to saying \node at (0,0) \{label text\});. There are other position options such as left, above left, etc which cause the label text to be placed in a separate node positioned appropriately. If the placement is above, left, etc, then any option that you may pass to a TikZ node will also work for the label, including general coordinate transformations. If the placement is “inside”, then the only relevant \text\{options\} are those that alter the appearance of text, such as opacity and color.
You can adjust the default behavior of class labels using the labels style option or its relatives class labels, inner class labels or outer class labels. Note that it is also possible to give a label to a \node this way, although the behavior is slightly different. In particular, the label defaults to the above position instead of going in the \node text by default. Also, this won’t respect the various label style options like labels, etc.

\begin{sseqpage}[classes = \{ minimum width = width("a") + 0.5em \}, no axes ]
\class["a"](0,0)
\class["a", red](1,0)
\class["a" black, red](2,0)
\class["b" above](0,1)
\class["b" { below right, yshift = 0.1cm }](1,1)
\class["a" { above right = \{1em\} }](2,1)
\end{sseqpage}

You can adjust the default behavior of class labels using the labels style option or its relatives class labels, inner class labels or outer class labels. Note that it is also possible to give a label to a \node this way, although the behavior is slightly different. In particular, the label defaults to the above position instead of going in the \node text by default. Also, this won’t respect the various label style options like labels, etc.

\begin{sseqpage}[no axes ]
\class(0,0)
\class(2,0)
\node[\textcircle, fill, "a"] at (1,0) ;
\end{sseqpage}

\begin{sseqpage}
\class(0,0)
\class["xy" { above, xshift = -4pt, pin = red }](0,0)
\class(0,0)
\class(0,1)
\structline
\end{sseqpage}

pin = (style)

The pin key makes SPECTRALSEQUENCES draw a line connecting the label to the relevant class, which can provide necessary clarification in dense diagrams. The pin key itself can take options which adjust the way that the line is drawn:

\begin{sseqpage}
\class(0,0)
\class["xy" { above, xshift = \{-4pt\}, pin = red }](0,0)
\class(0,0)
\class(0,1)
\structline
\end{sseqpage}

For either a \structline or a \class the label normally goes on the right side of the edge. The special option \textquote makes it go in the opposite position from the default. I imitated the label handling in the tikzcd package, so if you use tikzcd, this should be familiar.

\begin{sseqpage}[Adams grading, yscale = 0.63 ]
\class(0,0)
\class(0,1)
\class(0,2)
\structline["a" blue](0,0)(0,1)
\class(1,0)
\class(1,1)
\structline["b"](1,0)(1,1)
\draw["\cdot 2" { pos = 0.7, yshift = -5pt } ] 2 (1,0)
\end{sseqpage}

You can use the style options labels, edge labels, differential labels, and struct line labels to adjust the styling of edge labels. For instance, if you would prefer for the labels to default to the left hand side of the edge rather than the right hand side, you could say edge labels = \{auto = left\}. You can also use quotes to label edges drawn with TiKZ primitives:

\begin{sseqpage}[yscale = 0.58, no axes ]
\class(0,0)
\class(0,1)
\structline["a"](0,0)(0,1)
\draw["\cdot 2" { pos = 0.7, yshift = -5pt } ] 2 (1,0)
\end{sseqpage}
description

The description key, stolen from \tikzcd, places the label on top of the edge. In order to make this option work correctly, if the background color is not the default white, you must inform \spectralsequences about this using the key background color = \langle color\rangle. In this document, the background color is called \texttt{graphicbackground}.

\begin{sseqpage}[no axes, background color = \texttt{graphicbackground}]
\foreach \x in \{0,1,2\} \foreach \y in \{0,1\} {
    \class(\x,\y)
}
\structline[\textcolor{red}{a}](0,0)(0,1)
\structline[\textcolor{blue}{a'+b}]](1,0)(1,1)
\structline[\texttt{c} description](2,0)(2,1)
\end{sseqpage}

4.2 Options for \texttt{\class}

Because the main job of the \texttt{\class} command is to print a Ti\textit{k}Z \texttt{node} on the appropriate pages of the spectral sequence, most options that would work for a Ti\textit{k}Z node also work for the commands \texttt{\class}, \texttt{\replaceclass}, and \texttt{\classoptions}. Here are a few that you might care about:

A Ti\textit{k}Z shape

If you give the name of a Ti\textit{k}Z shape, the class node will be of that shape. The standard Ti\textit{k}Z shapes are \texttt{circle} and \texttt{rectangle}. \texttt{\spectralsequences} defines two new shapes:

\texttt{circlen = \langle n\rangle}

This draws \( n \) concentric circles. It’s intended for indicating a \( \mathbb{Z}/p^n \) summand. For large values of \( n \) the result isn’t all that appealing.

\begin{sseqpage}[no axes]
\class[circlen = 2](0,0)
\class[circlen = 2, fill](1,0)
\class[circlen = 3](0,1)
\class[circlen = 4](1,1)
\end{sseqpage}

\texttt{newellipse}

\texttt{ellipse ratio = \langle ratio\rangle}

This shape is used for \texttt{\circleclasses}. It’s a variant on the \texttt{ellipse} shape that gives more control over the ellipse’s aspect ratio.

There are many more Ti\textit{k}Z shapes in the shapes library, which you can load using the command \texttt{\usetikzlibrary{shapes}}. The following are some examples:

\begin{sseqpage}[no axes, \texttt{classes} = \{ \texttt{inner sep = 0.4em}, \texttt{class placement transform = \{ \texttt{scale = 1.8}, \texttt{yscale = 1.63} \} }]
\class(0,0)
\class[isosceles triangle](2,0)
\class[rectangle](1,0)
\class[diamond](0,1)
\class[semicircle](1,1)
\class[regular polygon, regular polygon sides = 5](2,2)
\class[regular polygon, regular polygon sides = 6](2,2)
\class[regular polygon, regular polygon sides = 7](2,2)
\class[regular polygon, regular polygon sides = 8](2,2)
\end{sseqpage}

See the Ti\textit{k}Z manual for more information.
minimum width = ⟨dimension⟩
minimum height = ⟨dimension⟩
minimum size = ⟨dimension⟩
inner sep = ⟨dimension⟩
outer sep = ⟨dimension⟩

These options control the size of a node. This is typically useful to make the size of nodes consistent independent of the size of their label text. For instance:

\begin{sseqdata}[name = minimum width example, no axes, yscale = 0.8]
\class["ab"](0,0)
\class["a"](0,1)
\class(0,2)
\end{sseqdata}
\printpage[ name = minimum width example ]
\printpage[ name = minimum width example, 
change classes = { blue, minimum width = width("ab") + 0.5em } ]

name = ⟨node name⟩

The \texttt{class} command makes a Ti\textit{k}Z node on appropriate pages. You can refer to this node using Ti\textit{k}Z commands by using its coordinates. Using the \texttt{name} option, you can give the node a name, which you can use to refer to the class. Using names creates more readable code. The \texttt{show name} option can be used to display the names of classes. You can modify the names of classes systematically using the options \texttt{class name prefix}, \texttt{class name postfix}, and \texttt{class name handler}.

Named classes are immune to coordinate transformations. For example, in the following code, \texttt{xshift} does not apply to the nodes specified by (id) and (eta) but does apply to the coordinate specified by (1,1):

\begin{sseqpage}[classes = { show name=above }]
\class[name = a](0,2)
\class[name = b](0,3)
\class[name = x](1,0)
\d2(x)(a)
\replacesource[name=2x]
\d3(x)(b)
\end{sseqpage}

show name = ⟨label options⟩

This option is like saying "\texttt{class name}"{⟨label options⟩} if the class has a name, and does nothing if the class has no name. If the class has multiple names, only the most recent is used. This is particularly useful with class styles, . For instance, by saying \texttt{this page classes = { show name = above}} you can display names of all of the sources and targets of differentials on each page.

\begin{sseqdata}[name = show name example, 
this page classes = { show name = { above right, pin } }]
\class[ name = a ](0,2)
\class[ name = b ](0,3)
\class[ name = x ](1,0)
\d2(x)(a)
\replacesource[name=2x]
\d3(x)(b)
\end{sseqdata}
\printpage[ name = show name example, page = 2]
\printpage[ name = show name example, page = 3]

tag = ⟨tag⟩
This key adds a tag to the current class. Tags are used for identifying which of multiple classes in the same position you are referring to. They are useful when you have groups of related classes and want a family of differentials connecting them. For instance:

\begin{sseqpage}[\text{degree} = {-1}{1}, \text{yscale} = 1.1, \text{x range} = {0}{5}, \text{y range} = {0}{5} ]
\tower(3,0)
\hvee[\text{tag} = \text{id} ](0,0)
\hvee[\text{tag} = \text{h21} ](4,2)
\foreach \n in {0,...,5} { \d2(4+\n,2+\n,h21,id) }
\end{sseqpage}

We want each differential to go from the h21 vee to the id vee, independent of which classes are in the same position of the two vees. The easy way to accomplish this is by giving tags to each of the two vees.

\text{\textbf{insert}} = \langle \text{\textit{integer}} \rangle

If there are multiple classes in the same position, this option allows you to insert classes later into earlier positions. This is intended to help you put logically related classes next to each other. If the integer is positive, it inserts the class in the specified position, and if the integer is negative, it counts backwards from the end. Providing 0 is the same as omitting the option entirely. Values larger in absolute value than the total number of classes are truncated. Consider:

\begin{center}
\begin{tabular}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 \\
4 & 4 & 4 & 4 & 4 & 4 & 4 \\
\end{tabular}
\end{center}
offset = \{(x offset), (y offset)\}

By default, a class uses the offset specified by class pattern. Occasionally this is undesirable. In this case, you can specify the offset for a particular class by hand. For example if the sum of two classes is hit by a differential, it looks better for the class replacing them to be centered:

```
\begin{sseqpage}
\class(0,1)
\class(0,2)
\draw(0,1)--(0,2);
\end{sseqpage}
```

```
\begin{sseqdata}
\class(0,0)
\class(0,1)
\class(0,2)
\class(1,0)
\end{sseqdata}
```

```
\begin{sseqpage}
\class(0,0)
\end{sseqpage}
```

```
\begin{sseqpage}
\class(0,0)
\end{sseqpage}
```

```
\begin{sseqpage}
\class(0,0)
\end{sseqpage}
```

```
\begin{sseqpage}
\class(0,0)
\end{sseqpage}
```

```
\begin{sseqpage}
\class(0,0)
\end{sseqpage}
```

tooltip = \{text\}

This key generates a “tooltip” over the given class. That is, if you hover your mouse over it, a little window will popup with the tooltip text. This is particularly useful to give the coordinates or names of classes in large charts where it may be hard to tell from looking at the picture what position the class is in, or there may not be room to supply names to classes.

The tooltip is made using the \pdftooltip command from the pdfcomment package. The pdfcomment package generates two extra auxiliary files, so it is not included by default. In order to use the tooltip option, you have to use the tooltips package option (e.g., load SPECTRALSEQUENCES with
\usepackage[toolips]{spectralsequences}. This cannot handle math, but it will print math expressions into \TeX input form. Not all pdf viewers will display the tooltip correctly. If this concerns you, the command \seqtooltip is used to produce the tooltip, and you can redefine it as any other command that takes \seqtooltip\{text\}\{tooltip text\} and produces a tooltip. For instance, on this stack exchange post, there is code that supposedly produces tooltips that work with Evince. I have not tested whether it works by itself or whether it works with my package, but you could. You could potentially figure out how to get math to work in tooltips too – if you find a satisfactory method, please let me know.

Anyways, here’s an example:

\begin{seqpage}\[classes = \{tooltip = \{(xcoord, ycoord)\}\}\]
\class(0,0)
\class(0,1)
\class(1,0)
\class(1,1)
\end{seqpage}

There’s another example at the beginning of the section on the class stack.

page = \langle page \rangle--\langle page max \rangle

generation = \langle generation \rangle--\langle generation max \rangle

These options only work in \classoptions. The page option gives a range of pages for which the options apply to. If only one page is specified, it is the minimum page and the option applies to all larger pages.

A “generation” of a class is the interval from one call of \class or \replaceclass to the page on which it next supports or is hit by a differential. By default the \classoptions command adds options only to the most recent generation of the class in a \seqdata environment, or on the generation appropriate to the current page in a \seqpage environment. Using the generation option allows you to provide a single generation or range of generations of the class that the options should apply to. The first generation is generation 0, and the most recent generation is generation -1. Larger negative values count backwards.
These commands represent the x and y coordinate of the current class when used in class options. The only use I have for them is in the tooltip option, but maybe there is some other purpose for them.

4.3 Options for \d and \structline

Because the main job of the \d and \structline commands is to print an edge on the appropriate pages of the spectral sequence, most Ti\LaTeX options that you could apply to a Ti\LaTeX “to” operator (as in \draw (x1,y1) to (x2,y2);) can be applied to both \d and \structline. Some such options are as follows:
source anchor = \{anchor\}

target anchor = \{anchor\}

Because you can’t use the normal TikZ mechanism for specifying the source and target anchors, \texttt{SPEC-}
\texttt{TRALSEQUENCES} has these two keys for \texttt{\textbackslash d} and \texttt{\textbackslash structline}:

\begin{verbatim}
\begin{seqpage}[ no axes, yscale = 1.24 ]
\foreach \x in {0,1} \foreach \y in {0,1} {
  \class(\x,\y)
}
\structline(0,0)(0,1)
\structline[ source anchor = north west, target anchor = -30 ](1,0)(1,1)
\end{seqpage}
\end{verbatim}

\textbf{shorten} $>$ = \texttt{(distance)}
\textbf{shorten} $<$ = \texttt{(distance)}

These behave exactly like the corresponding options from TikZ, shortening the end and beginning of
the edge respectively. Note that you can lengthen the edge by shortening by a negative amount.

Dash patterns:

See the TikZ manual for a complete explanation of the dash pattern related options. Some examples:

\begin{verbatim}
\begin{seqpage}[ no axes, yscale = 1.6 ]
\foreach \x in {0,1,2} \foreach \y in {0,1} {
  \class(\x,\y)
}
\structline[ densely dotted](0,0)(0,1)
\structline[ dashed, red, "a"](1,0)(1,1)
\structline[ dash dot, red, "a" black](2,0)(2,1)
\end{seqpage}
\end{verbatim}

\begin{verbatim}
\begin{seqpage}[ no axes, yscale = 1.6 ]
\foreach \x in {0,1,2} \foreach \y in {0,1} {
  \class(\x,\y)
}
\structline[ bend left = 20 ](0,0)(0,1)
\structline[ bend right = 20 ](1,0)(1,1)
\structline[ in = 20, out = north ](2,0)(2,1)
\end{seqpage}
\end{verbatim}

\textbf{page} = \texttt{(page)}\texttt{--}(\texttt{page max})

This key is only for \texttt{\textbackslash structline} and \texttt{\textbackslash structlineoptions}. By default, the \texttt{\textbackslash structline} command only adds a structure line starting on the page where the most recent generation of the source or target
is born:
By specifying a page number, you can adjust which page the \structline starts on:

Similarly, for \structlineoptions you can specify a minimum page on which to apply the options, or a range of pages.

### 4.4 Options for \circleclass

**fit = (coordinates or nodes)**

The \circleclasses command uses the TikZ fitting library. Sometimes it’s desirable to make the resulting node fit extra things, for example a label. It doesn’t necessarily end up looking great though.
rounded rectangle
You can put a shape as an option and it will change the shape of the node drawn by \texttt{\circlecycles}. Any shape will do, but I think that an ellipse or rounded rectangle are the only particularly appealing options.

\texttt{ellipse ratio = (\textit{ratio})}
(initially 1.2)
By default, the shape drawn by \texttt{\circlecycles} is a “newellipse” which is a custom defined shape that respects the option \texttt{ellipse ratio} which roughly controls how long and skinny versus short and fat the ellipse is. If you find that the ellipse is too long, try a larger value of this option, and conversely if it’s too fat try a smaller value. If no value is satisfactory, try out the rounded rectangle shape. (This is stolen from the following stack exchange answer: https://tex.stackexchange.com/a/24621.)

class style
permanent cycle style
transient cycle style
this page class style
differential style
struct line style
See the corresponding entry in the TikZ primitives section.

\texttt{page = (page)--(page max)}
By default, the ellipse will be drawn on the same set of pages that a structure line between the two classes would be drawn on. This specifies a range of pages for the ellipse to be drawn. Note that unlike with structure lines, you can instruct \texttt{\circlecycles} to draw the shape even on pages where one or both of the classes that it is fitting are dead.

4.5 Options for TikZ primitives

background
This key instructs \texttt{SPECTRALSEQUENCES} to put the current TikZ primitive in the background. The way that the spectral sequence is printed is as follows:

- The title, axes, axes ticks, and axes labels are printed (the appropriate steps are skipped when the \texttt{no title}, \texttt{no axes}, \texttt{no ticks}, or \texttt{no labels} keys are used or if no title or axes labels are provided).
- The TikZ background paths are printed.
- The clipping is inserted (unless the \texttt{no clip} key is used).
- All foreground elements (classes, differentials, structure lines, and normal TikZ paths) are printed.

In particular, this means that foreground TikZ paths can be clipped by the standard clipping, but background paths that are outside of the clipping expand the size of the TikZ picture.
Here is an example where Ti\textit{k}Z labels with the \texttt{background} key are used to add labels and a grid. Note that this styling is easier to make using the \texttt{title}, \texttt{x label}, \texttt{y label}, and \texttt{grid} options.

\begin{sseqpage}[ no ticks, yscale = 0.9, math nodes = false ]
\class(0,0)
\class(1,1)
\begin{scope}[background]
\draw(0.1,0.1)--(1.1,1.1);
\end{scope}
\node[background] at (0.5,-1) {not clipped};
\node at (0.5,-0.4) {clipped};
\end{sseqpage}

For this particular use case, it’s probably better to use \texttt{title}, \texttt{x label}, and \texttt{y label}:}

\begin{sseqdata}[ name = tikz background example, cohomological Serre grading, classes = fill ]
\begin{scope}[background]
\node at (\xmax/2,\ymax+1.2) {\textup{Page \page}};
\node[rotate = 90] at (-1.5,\ymax/2) {H^*(F)};
\draw[step = 1cm, gray, very thin] (\xmin-0.5,\ymin-0.5) grid (\xmax+0.4,\ymax+0.5);
\end{scope}
\class(0,0)
\class(3,0)
\class(0,2)
\class(3,2)
\d3(0,2)
\end{sseqdata}

\printpage[name = tikz background example, page = 2]
\printpage[name = tikz background example, page = 3]
But if you need more flexible labeling, you’ll likely want to use TikZ primitives with `background`. See `example_KF3.tex` for an instance where this key is useful.

One useful tip is that you can ensure consistent bounding boxes between different diagrams using `\path[background] (smallest x, smallest y) -- (largest x, largest y);`:

```
\begin{sseqdata}
    \name = tikz background example2, 
    \xrange = {0}{2}, \yrange = {0}{2}, \scale = 0.5
\end{sseqdata}

\printpage[\name = tikz background example2, \title = not aligned]
\path[background] (\xmin,\ymin-4) -- (\xmax,\ymax+2);
\printpage[\name = tikz background example2, \xlabel = Hi, \title = {}]
```

```
\begin{sseqdata}
    \name = boundingboxes, \xrange = {0}{2}, \yrange = {0}{2}, \scale = 0.5
\end{sseqdata}

\printpage[\name = boundingboxes, \title = not aligned]
\path[background] (\xmin,\ymin-4) -- (\xmax,\ymax+2);
\printpage[\name = boundingboxes, \xlabel = Hi, \title = {}]
```

---

`\isalive(⟨coordinate⟩)`

\isalive{⟨coordinate 1⟩}⋯{⟨coordinate n⟩}

This command can only be used with `page constraint`. Saying

```
\page constraint = {\isalive(⟨x⟩,⟨y⟩,[(index)])}
```

will print the TikZ primitive only on pages where the specified class is alive. Saying

```
\page constraint = {\isalive(\meta{coordinate 1}) ⋯ (⟨coordinate n⟩)}
```

is equivalent to

```
\page constraint = {\isalive(⟨coordinate 1⟩) & & \isalive(⟨coordinate n⟩)}
```

Writing

```
\draw[\page constraint = {\isalive(1,0)(2,2)}](1,0)--(2,2);
```

is the same as `\structline(1,0)(2,2)`, except that you can’t later use `\structlineoptions` on it (and it won’t have the `struct lines` style applied).
These classes apply the styling of the corresponding element to your TikZ commands.

```
\begin{sseqpage}[ differials = blue, yscale = 0.65, no axes ]
\class(0,2)
\class(1,0)
% This will be styled as if it were a differential
\draw[differential style] (1,0) -- (0,2);
\end{sseqpage}
```

See \getdtarget for a more natural example.

## 5 Miscellaneous Commands

### 5.1 Settings

`\sseqset{⟨keys⟩}`

The `\sseqset` command is for adjusting the global options for all spectral sequences in the current scope, or for applying options to the rest of the current spectral sequence. For instance, if most of the spectral sequences in the current document are going to be Adams graded, you can say `\sseqset{Adams grading}` and all future spectral sequences in the current scope will have Adams grading (unless you specify a different grading explicitly). As another example, `\sseqset{no axes}` will suppress axes from spectral sequences in the current scope. Note that defaults only apply to new `\sseqdata` environments or to unnamed `\sseqpage` environments; they won’t apply to existing spectral sequences.

You can also use `\sseqset` to create styles to be used in spectral sequences.

```
.global sseq style = ⟨keys⟩
global sseq append style = ⟨keys⟩
sseq style = ⟨keys⟩
sseq append style = ⟨keys⟩
```

These handlers create reusable styles to be used in spectral sequences. If this style is a set of global options, then use the `.global sseq style` handler, whereas if it is supposed to be applied to individual features (classes, differentials, structure lines, circle classes, and tikz primitives) then use the `.sseq style` handler.

```
\begin{sseqpage}[ myseq ]
\class(0,0) \class(0,1) \class(0,2) \class(0,3)
\class(3,1) \class(3,2) \class(3,3)
\structline(0,0)(0,1) \structline(0,1)(0,2)
\structline(0,2)(0,3) \structline(3,1)(3,2) \structline(3,2)(3,3)
\structline[htwostruct](0,0)(3,1) \structline[htwostruct](0,1)(3,2)
\structline[htwostruct](0,2)(3,3)
\end{sseqpage}
```
\SseqErrorToWarning\{error-name\}

Turns the error with the given name into a warning. An error message will start by saying `spectralsequences error: "error-name"`. This is the name you need to put into this command.

\begin{quiet}
\begin{environment contents}
\end{quiet}

This environment quiets error messages that occur inside of it. `SPECTRALSEQUENCES` is pretty good at error recovery, and so most of commands will fail gracefully and do nothing if their preconditions aren’t met. If there are any parsing errors in the body of the `\{quiet\}` environment, prepare to see low level internal error messages. You might also run into bugs in `SPECTRALSEQUENCES` – the error recovery code hasn’t been that carefully tested. If you do get low level error messages, remember to comment out the `\{quiet\}` environment before trying to debug.

This is particularly useful for code reuse commands. Sometimes there is a source of long differentials that only applies to classes that haven’t already supported shorter differentials. Sometimes there should be a structure line if a certain class exists, but it might not exist. In these cases, the `\{quiet\}` environment will help you out. See also `\DrawIfValidDifferential`, which is a variant of `\d` that behaves as if it were inside a `\{quiet\}` environment.

5.2 Code reuse commands
\foreach

This command is from Ti\kZ and works in pretty much the same way in `SPECTRALSEQUENCES`, though with slightly better variants. The `\foreach` command is very flexible and has a lot of variants. The basic usage is `\foreach \x in \{\{xmin\},...,\{xmax\}\} \{\{loop body\}\}` which will execute `\{loop body\}` with `\x` set to each value between `\{xmin\}` and `\{xmax\}` inclusive. If you want a step greater than 1, try `\foreach \x in \{\{xmin\},\{xmin\}+\{xstep\},...,\{xmax\}\} \{\{loop body\}\}.

If you need to do multiple loops with a common body, you can just stack the `\foreach` commands:

```
\begin{sseqpage}[xscale = 0.5, x tick step = 2]
\foreach \x in {0,2,...,6}
\foreach \y in {0,...,3}
\class(\x,\y)
\end{sseqpage}
```

You can also loop through tuples, for instance:

```
\begin{sseqpage}[xscale = 0.5]
\foreach \x/\y/\label in {0/1/a,1/1/b,0/0/c,1/0/d}{
\class["\label" above](\x,\y)
}
\end{sseqpage}
```

See the last example for `normalize monomial` for a better example of this usage.

There are tons of other things you can do with `\foreach`, though I haven’t yet found need for them in combination with `SPECTRALSEQUENCES`. See the Ti\kZ manual for more details.
The one use case that \texttt{foreach} doesn’t cover all that well is if you want the loop to always repeat until the features you are drawing go off the page. This is what \texttt{DoUntilOutOfBounds} and \texttt{DoUntilOutOfBoundsThenNMore} are for. These help ensure that if you change the range of your chart, infinite families will automatically be drawn correctly without the need to adjust a bunch of loop bounds.

The purpose of \texttt{DoUntilOutOfBoundsThenNMore} is for towers that are receiving a differential. If your spectral sequence is Adams graded, and a tower is receiving a $d_r$ differential from another tower, you should use \texttt{DoUntilOutOfBoundsThenNMore}{r}:

You can also nest \texttt{DoUntilOutOfBounds} reasonably:

One important difference between \texttt{foreach} and the \texttt{Do} family of commands is that \texttt{Do} has no effect on the stack. This is in order to ensure that they nest properly.

Note that if you are using these commands and you are planning to draw several pictures of the chart with restricted range, you need to specify a range for the \texttt{sseqdata} that contains all of the ranges of pages that you want to draw. If you then want to set a smaller default range, specify the smaller range the first time you use \texttt{sseqpage} or \texttt{printpage} to draw the spectral sequence, and include the \texttt{keep changes} key.

The \texttt{Do} command is less general than \texttt{foreach}; the purpose is to provide a syntax for stack-based looping that is similar to \texttt{DoUntilOutOfBounds} but with a fixed range. So \texttt{Do(n}{loop body}) repeats \texttt{loop body} n times. The assumption is that the loop body draws something relative to the position of the \texttt{lastclass}.

If you need to know how many iterations one of these three commands has gone through, this is stored in the variable \texttt{iteration}. 25
\NewSseqCommand\langle command\rangle\{(argspec)\}\{(body)\}
\DeclareSseqCommand\langle command\rangle\{(argspec)\}\{(body)\}

The \texttt{xparse} package provides these very powerful commands for defining macros. They are used internally to the \texttt{SPECIALSEQUENCES} package to define \texttt{\class}, \texttt{\d}, etc. To help you create variants of these commands, I will record here the argument specifications for each of them. See the \texttt{xparse} manual for a better explanation and more information.

To make a command like \texttt{\class}, you can use the argument specification \texttt{O{}r()}. The argument type \texttt{O\{\textit{default}\}} stands for a bracket delimited optional argument with default value \texttt{\langle default\rangle}. In this case, we’ve specified the default to be empty. \texttt{r()} stands for a “required” argument delimited by \texttt{(} and \texttt{)}. In the command definition, access the optional argument with \texttt{#1} and the coordinate with \texttt{#2}.

\begin{verbatim}
#1 = {key = value}; #2 = {x,y}
#1 = {}; #2 = {1,2,3}
\end{verbatim}

If you want to separate out the coordinates into different arguments, you can use \texttt{O\{u(u,u)\}}. The argument type \texttt{u} stands for “until” and scans up until the next instance of the given character. So in this case, \texttt{#1} is of argument type \texttt{O} which is an option list, \texttt{#2} corresponds to the \texttt{u} (which is a throw-away argument, then \texttt{#3} corresponds to \texttt{u}, and contains the \texttt{x} coordinate, and \texttt{#4} corresponds to \texttt{u}) and contains the \texttt{y} coordinate. Note however that this will not match balanced parenthetical expressions.

\begin{verbatim}
#1 = {hi}; #3 = {x}; #4 = {y}
#1 = {}; #3 = {1}; #4 = {2}
#1 = {}; #3 = {(1+1)*2}; #4 = {2}
#1 = {}; #3 = {1}; #4 = {(1+1} *2)
\end{verbatim}

You can specify an optional argument delimited by parentheses using \texttt{d()}. Use the commands \texttt{\IfNoValueTF}, \texttt{\IfNoValueT}, and \texttt{\IfNoValueF} to test whether the user provided a value.

\begin{verbatim}
#1 = {hi}; #2 = {x,y}
#1 = {options}; #2 = {no value}
#1 = {}; #2 = {1,2}
#1 = {}; #2 = {no value}
\end{verbatim}

An example where this is actually useful:

\begin{verbatim}
\begin{sseqpage}
3 2 1 0
\begin{tikzpicture}
\draw (0,0) -- (1,2);
\end{tikzpicture}
\end{sseqpage}
\end{verbatim}

The \texttt{\structline} and \texttt{\changeclasses} commands have argument specification \texttt{O{}r()r()}.
#1 = {hi}; #2 = {x,y}; #3 = {x',y'}

#1 = {}; #2 = {x,y,2}; #3 = {x',y',-1}

\DeclareDocumentCommand\demo{ O{} r() r() }{( \#1 = \textcolor{purple}{\{#1\}}; \#2 = \textcolor{purple}{\{#2\}}; \#3 = \textcolor{purple}{\{#3\}} )}

\hbox{\demo[hi](x,y)(x',y')}
\bigskip
\hbox{\demo(x,y,2)(x',y',-1)}

These are similar to \NewSseqCommand and \DeclareSseqCommand except that the commands defined take an optional square bracket delimited options list, followed by an optional parenthesis delimited (x,y) pair and wraps the body of the code in \begin{scope}[xshift = ⟨xcoord⟩, yshift = ⟨ycoord⟩, options]. For instance:

\begin{center}
\begin{tikzpicture}
\begin{scope}[xshift = 0 cm, yshift = 0 cm, options]
\class(0,0)
\foreach \n in {1,...,#1} {
\class(0,\n)
\structline(0,\n-1)(0,\n)
}
\end{scope}
\end{tikzpicture}
\end{center}

Here we define a command which takes an options list, a coordinate, and a single mandatory argument. The options are applied to all the classes and structlines, the coordinate shifts the origin, and the mandatory argument is the length of the tower.

\seqCopyPage\{⟨command⟩\}(⟨spectral sequence name⟩){⟨page⟩}

This defines \langle command \rangle to print all of the classes and structlines on page ⟨page⟩ of the spectral sequence named ⟨spectral sequence name⟩, throwing out differentials and tikz primitives. The resulting command has syntax as if you had used \DeclareSseqGroup with no extra arguments – that is, it takes an optional options list, which are passed to a scope, and an optional coordinate pair.

5.3 Families

SPECTRALSEQUENCES has a concept of “class families” that can be drawn or suppressed as a group, and that can have styling options applied to them as a group.

\seqNewFamily{⟨family name⟩}

This makes a new family with name ⟨family name⟩. The effect of this is to define global options ⟨family name⟩ style which apply options to all classes in the family, draw ⟨family name⟩ which causes the family to be drawn, and no ⟨family name⟩ which suppresses the family (drawing is the default behavior of course). It also defines an option ⟨family name⟩ which puts a feature into the family – so it can be used as an option to any of the normal commands like \class, \d, \structline, etc and to a scope.
5.4 Utilities

\seqparseint\(⟨macro⟩\{⟨integer expression⟩⟩

Stores the result of evaluating an integer expression into \(⟨macro⟩\). An integer expression consists of +, -, *, /, parentheses, and macros that expand to more of the same. The exact rules regarding what is a valid expression are pretty much what you would expect. Note that juxtaposition is a syntax error, not multiplication, so \(2(1+1)\) is invalid, you must say \(2*(1+1)\).

\seqifemptytf\(⟨expression⟩\{⟨true code⟩\}{⟨false code⟩}\)
\seqifemptyt\(⟨expression⟩\{⟨true code⟩\}\)
\seqifemptyf\(⟨expression⟩\{⟨false code⟩\}\)

This tests if an expression is the empty expression. This is mainly useful for giving systematic labels to
things.

\IfExistsTF{(page)}{(coordinate)}{(true code)}{(false code)}
\IfExistsF{(page)}{(coordinate)}{(true code)}{(false code)}
Test whether a class of the given description exists. The description can be any valid coordinate, including a named coordinate. If the coordinate is not valid, this will return false.

\IfAliveTF{(page)}{(coordinate)}{(true code)}{(false code)}
\IfAliveF{(page)}{(coordinate)}{(true code)}{(false code)}
Test whether a class is alive on the given page. If the class doesn’t exist, this will return false.

\IfOutOfBoundsTF{(coordinate)}{(true code)}{(false code)}
\IfOutOfBoundsF{(coordinate)}{(true code)}{(false code)}
\IfInBoundsTF{(coordinate)}{(true code)}{(false code)}
\IfInBoundsF{(coordinate)}{(true code)}{(false code)}
Test whether a class is in bounds or out of bounds. If no such class exists, returns false.

\IfValidDifferentialTF{(page)}{(coordinate)}{(true code)}{(false code)}
\IfValidDifferentialF{(page)}{(coordinate)}{(true code)}{(false code)}
\DrawIfValidDifferentialTF{(options)}{(page)}{(coordinate)}{(true code)}{(false code)}
\DrawIfValidDifferentialF{(options)}{(page)}{(coordinate)}{(true code)}{(false code)}
Test whether a differential is valid. There are many possible reasons for a differential to not be valid – the source or target class could not exist, the target could not be in the correct grading, etc. If any error would be generated if you supplied the same arguments to \d, this tests false. Otherwise, this tests true. The command \DrawIfValidDifferential draws the differential if it is valid, whereas \DrawIfValidDifferentialT draws the differential and executes the true code if it is valid, \DrawIfValidDifferentialF draws the differential or executes false code if it isn’t valid, etc.

\SseqNormalizeMonomial
This command simplifies a monomial by combining like variables and removing bases that are raised to the 0th power, removing exponents of 1, removing 1’s, and replacing the empty monomial with 1. The variables are sorted by first occurrence, use \SseqNormalizeMonomialSetVariables to set a different sort order. It outputs its result into \result. This command is specifically meant to be used as a value for class label handler or class name handler. See the example in the documentation for class label handler for a realistic example. The exponents must be integers or else it will misbehave.

\SseqNormalizeMonomial{x^0y^0} $\result$, \quad \SseqNormalizeMonomial{x^{-3}y^0} \quad \SseqNormalizeMonomial{1x^2} \quad \SseqNormalizeMonomial{1x^2y^2} $\result$

\SseqNormalizeMonomialSetVariables
This command takes an undelimited list sets up \SseqNormalizeMonomial to sort the variables in the input in the specified order. This is useful for consistency, and particularly important for when you use it to name classes. For example:

1, \quad x^2y, \quad x^3 \quad \SseqNormalizeMonomial\quad \SseqNormalizeMonomial\quad \SseqNormalizeMonomial\quad \SseqNormalizeMonomial\quad \SseqNormalizeMonomial\quad \SseqNormalizeMonomial
We always want a first, then $b$, then $x_1$, then $x_2$, then $x_3$.

\SseqNormalizeMonomialSetVariables{abx_1x_2x_3}
\SseqNormalizeMonomial{ba} $\result$, \quad \SseqNormalizeMonomial{bx_1} $\result$, \quad \SseqNormalizeMonomial{abx_2x_3x_1} $\result$

\SseqAHSSNameHandler
This command expects an argument of the form \langle monomial \rangle \langle integer expression \rangle and defines $\result$ to be $m[n]$ where $m$ is the result of applying $\SseqNormalizeMonomial$ to the monomial, and $n$ is the result of evaluation the integer expression plus any $y$shift that is present. This is intended for use with in drawing Atiyah Hirzebruch spectral sequences. See class name handler for an example of the calculation of $KO^*\mathbb{R}P^8$.

5.5 Coordinate parsers and related

\parsecoordinate\langle macro\rangle\langle coordinate\rangle
This command parses the coordinate and puts the triple $(x,y,n)$ into \langle macro\rangle. It also puts the components of the coordinate into macros, for instance if \langle macro\rangle is \coord, then \xcoord will contain the x coordinate, \ycoord will contain the y coordinate, and \ncoord will contain the index. The coordinate can be anything that would be valid to use in a differential or structure line; this is the macro that is used internally to handle coordinates.

\getdtarget\langle macro\rangle\langle page\rangle\langle source coordinate\rangle
Sets \langle macro\rangle equal to the coordinates of the target position of a length \langle page\rangle differential starting at \langle source coordinate\rangle. This helps to make commands that draw fancy differentials.

\parsedifferential\langle page\rangle\langle differential coordinate\rangle\langle differential target\rangle
This has the same weird syntax of $\d$, except that you are required to put braces around the page (if it has multiple digits) and you are required to provide at least one coordinate (you have to say \parsedifferential{2}\langle lastclass\rangle for instance). This command is similar in effect to saying both \parsecoordinate\langle source\rangle\langle source coordinate\rangle and \parsecoordinate\langle target\rangle\langle target coordinate\rangle, but it handles determination of the target coordinate for you.

For instance, consider the following example, suggested by Catherine Ray:

```
\begin{tabular}{|c|c|c|}
\hline
2 & $\odot$ & 2 & $\odot$ & 2 \\
1 & & 1 & & 1 \\
0 & $\odot$ & 0 & $\odot$ & 0 \\
0 & 1 & 0 & 1 & 0 & 1 \\
\hline
\end{tabular}
```
\nameclass{(name)}{(coordinate)}

The \nameclass{} command gives a name to a class. It’s similar to saying \doptions{name = (name)}, but faster. It’s also similar to saying \pushstack{(coordinate)}. Giving temporary names to coordinates that you are going to use repeatedly makes the code easier to read and is faster (though this only matters in very large charts). See \getdtarget for an example.

\tagclass{(tag)}{(coordinate)}

The \tagclass{} command gives a tag to a class. It’s similar to saying \doptions{tag = (name)}, but faster. See example_tmffayss.tex for a use case for this.

\gettag\{macro\}{(coordinate)}

The \gettag{} command finds the most recent tag applied to the coordinate and stores it into \cmdarg{macro}. This is useful for connecting groups of nodes. For example, consider the following code, inspired by example_tmffayss.tex.
5.6 The class stack

The class stack is a linked list of the classes in the order that they are produced that SPECTRALSEQUENCES maintains. I’ve only recently implemented this feature, so it is more liable to change in the future than other things. Whenever you use the \( \text{class} \) function, the class you added is pushed onto the stack. Here’s an example that demonstrates basic usage:

The following commands are used to access the stack:

\begin{itemize}
\item \texttt{\lastx(n)}
\item \texttt{\lasty(n)}
\item \texttt{\lastclass(n)}
\end{itemize}

The commands \texttt{\lastx} and \texttt{\lasty} evaluate to the \( x \) and \( y \) position, respectively, of the \( n \)th class on the stack. If \( n = 0 \) you can leave it off. The command \texttt{\lastclass} evaluates to the coordinate of the most recent class on the stack. This is useful for writing turtle-style code:
You can use \lastx and \lasty in other contexts than in the body of a \texttt{spectralsequences} command, most notably inside \texttt{SseqParseInt} (they also go fine inside \texttt{pgfmathparse} if you need it or one of its siblings). For instance, consider the following tower command:

\begin{sseqpage}
\begin{scope}[yrange = {0}{5}]
\begin{scope}
\class(0,0)
\tower
\class(1,3)
\tower
\class(3,2)
\tower
\end{scope}
\end{scope}
\end{sseqpage}

This adds a class to the top of the stack. The coordinate is specified using the same syntax as a coordinate for \texttt{structline} or \texttt{replaceclass}.

\begin{sseqpage}
\begin{scope}[yrange = {0}{5}]
\begin{scope}
\pushstack(⟨coordinate⟩)
\savestack
\foreach \n in {1,...,\numclasses}{
\class(\lastx,\lasty+1)
\structline
}\restorestack
\tower
\end{scope}
\end{scope}
\end{sseqpage}

This saves and reverts the stack. Saves nest. Most frequently, you will want to use these at the start and end of a command.

6 Styles

The \texttt{spectralsequences} package has a large number of styles which control the appearance of specific components (e.g., classes, differentials, or structure lines) of a spectral sequence. Each style has two corresponding keys: \texttt{classes} and \texttt{change classes}. Saying \texttt{classes = \{(keys)\}} adds the keys to the list of options used
to style every future class, whereas `change classes = {⟨keys⟩}` only makes sense in a `{sseqpage}` environment, and temporarily overwrites the list of options. Note that `change classes` only applies to classes that existed before the current page, and that even with the `keep changes` option, the `change classes` options are local to the current page. Compare:

```
\begin{sseqdata}[ name = style example ]
\class(0,0)\class(1,1)
\end{sseqdata}
\begin{sseqpage}[ name = style example, classes = { fill, blue },
title = change new classes ]
\class(0,1)\class(1,0)
\end{sseqpage}
\quad
\begin{sseqpage}[ name = style example, change classes = { fill, blue },
title = change old classes ]
\class(0,1)\class(1,0)
\end{sseqpage}
```

You can modify these styles outside of a spectral sequence or inside it using `\sseqset`, you can modify them as options to the `{sseqdata}` and `{sseqpage}` environments, or you can modify them as arguments to the `{scope}` environment.

In cases where the same drawing feature is affected by multiple of these styles, the more specific style takes precedence. For instance, for a class that is the source or target of a differential on the current page, the precedence order from lowest to highest goes: `sseq style`, `class style`, `transient cycle style`, `this page cycle style`, and then any options from scopes in the order they appear, and any local options (the options that come right with the class, e.g., `\class[local options](x,y)`). If you don’t want the options to your scopes to override more specific styles, use `sseq`:

```
\begin{sseqpage}[ classes = { blue, fill },
title style = { align = center, text width = 2.4cm },
title = { everything is orange } ]
\begin{scope}[orange]
\class(0,0)\class(0,1)
\structline(0,0)(0,1)
\end{scope}
\end{sseqpage}
\begin{sseqpage}[ classes = { blue, fill },
title style = { align = center, text width = 2.4cm },
title = { only structure line is orange } ]
\begin{scope}[sseq = orange]
\class(0,0)\class(0,1)
\structline(0,0)(0,1)
\end{scope}
\end{sseqpage}
```

Throughout, “class” and “cycle” are synonyms.

```
sseqs = {⟨keys⟩}
change sseqs = {⟨keys⟩}
sseq = {⟨keys⟩}
change sseq = {⟨keys⟩}
```

This passes options to all features in all future spectral sequences in the current scope. Note that for many global options you can set a default directly by saying `\sseqset{key = {⟨value⟩}}` and this is in some cases preferable.
% Applies to both of the following sseqs:
\sseqset{ sseqs = { blue, scale = 0.5 } }
\begin{ssepage}
\foreach \x in {0,1}
\foreach \y in {0,1,2} {
  \class(\x,\y)
}
\end{sseqpage}

\begin{ssepage}
\foreach \x in {0,1,2}
\foreach \y in {0,1,2} {
  \class(\x,\y)
}\d2(1,0)
\d2(2,0)
\end{sseqpage}

\begin{sseqpage}
\foreach \x in {0,2}
\foreach \y in {0,1} {
  \class["\mathbb{Z}"](\x,\y)
}\d2(0,1)
\end{sseqpage}

\begin{sseqpage}
\begin{align*}
\text{classes} &= \{\langle \text{keys} \rangle \} \\
\text{cycles} &= \{\langle \text{keys} \rangle \} \\
\text{change classes} &= \{\langle \text{keys} \rangle \} \\
\text{change cycles} &= \{\langle \text{keys} \rangle \}
\end{align*}
\begin{align*}
\text{permanent classes} &= \{\langle \text{keys} \rangle \} \\
\text{permanent cycles} &= \{\langle \text{keys} \rangle \} \\
\text{change permanent classes} &= \{\langle \text{keys} \rangle \} \\
\text{change permanent cycles} &= \{\langle \text{keys} \rangle \}
\end{align*}
\begin{align*}
\text{transient classes} &= \{\langle \text{keys} \rangle \} \\
\text{transient cycles} &= \{\langle \text{keys} \rangle \} \\
\text{change transient classes} &= \{\langle \text{keys} \rangle \} \\
\text{change transient cycles} &= \{\langle \text{keys} \rangle \}
\end{align*}

These options change the appearance of all permanent cycles (e.g., those classes which never support or are hit by a differential). For instance, we can circle the permanent cycles automatically. In the following example, note that because permanent cycles is more specific than classes, the permanent cycles = \{draw\} command takes precedence over the classes = \{draw = none\} command and the permanent cycle nodes are drawn.

\begin{sseqpage}
\begin{align*}
\text{cohomological Serre grading,} \\
\text{classes} &= \{ \text{draw = none } \}, \\
\text{permanent cycles} &= \{ \text{draw} \}
\end{align*}
\begin{align*}
\text{permanent cycles} &= \{ \text{draw} \}
\end{align*}
\begin{align*}
\text{change permanent cycles} &= \{ \text{draw} \}
\end{align*}
\begin{align*}
\text{transient classes} &= \{ \langle \text{keys} \rangle \} \\
\text{transient cycles} &= \{ \langle \text{keys} \rangle \}
\end{align*}
\begin{align*}
\text{change transient classes} &= \{ \langle \text{keys} \rangle \} \\
\text{change transient cycles} &= \{ \langle \text{keys} \rangle \}
\end{align*}

These options change the appearance of all transient cycles (e.g., those classes which eventually support or are hit by a differential). Again, this takes precedence over the classes option.

35
\textbf{this page classes} = \{\langle\text{keys}\rangle\}
\textbf{this page cycles} = \{\langle\text{keys}\rangle\}
\textbf{change this page classes} = \{\langle\text{keys}\rangle\}
\textbf{change this page cycles} = \{\langle\text{keys}\rangle\}

These options change the appearance of all cycles which support or are hit by a differential on this page. Any class that is hit on the current page is also a transient cycle, and so \textbf{this page classes} takes precedence over \textbf{transient cycles}.

\textbf{edges} = \{\langle\text{keys}\rangle\}
\textbf{differentials} = \{\langle\text{keys}\rangle\}
\textbf{struct lines} = \{\langle\text{keys}\rangle\}
\textbf{change edges} = \{\langle\text{keys}\rangle\}
\textbf{change differentials} = \{\langle\text{keys}\rangle\}
\textbf{change struct lines} = \{\langle\text{keys}\rangle\}

The \textbf{edges} key applies to both differentials and structure lines. The \textbf{differentials} and \textbf{struct lines} keys both take precedence over \textbf{edges}.

\textbf{this page struct lines} = \{\langle\text{keys}\rangle\}
\textbf{change this page struct lines} = \{\langle\text{keys}\rangle\}

This style applies to structure lines whose source or target is hit on the current page. It takes precedence over \textbf{struct lines}.

\textbf{tikz primitives} = \{\langle\text{keys}\rangle\}
\textbf{change tikz primitives} = \{\langle\text{keys}\rangle\}

Applies to all TikZ primitives.
labels = \{\langle keys \rangle \}
change labels = \{\langle keys \rangle \}

This style applies to labels on classes, differentials, and structure lines. All the more specific label styles take precedence over it.

class labels = \{\langle keys \rangle \}
inner class labels = \{\langle keys \rangle \}
outer class labels = \{\langle keys \rangle \}
change class labels = \{\langle keys \rangle \}
change inner class labels = \{\langle keys \rangle \}
change outer class labels = \{\langle keys \rangle \}

Inner class labels specifically applies to class labels that are inside the node, outer class labels specifically applies to ones outside it:

\begin{sseqpage}
\noaxes, classes = \{ inner sep = 1pt \},
\label distance = 2pt,
outer class labels = \{ red \},
inner class labels = \{ blue \ }
\class["a", "b" above](0,0)
\class["a", "c" right](1,0)
\end{sseqpage}

differential labels = \{\langle keys \rangle \}
struct line labels = \{\langle keys \rangle \}
change differential labels = \{\langle keys \rangle \}
change struct line labels = \{\langle keys \rangle \}

6.1 Style-like options

The options are not styles, but can be modified in the same set of places (namely, anywhere):

label distance = \langle dimension \rangle

This sets the default distance from a class to an outer label. There are also variants like above label distance corresponding to above, below, left, right, above left, above right, below left, and below right.

\begin{sseqpage}
\label distance = 0.7em,
right label distance = 0em,
no axes, yscale = 1.25 
\class["a" above](0,0)
\class["b" above right](0,1)
\class["c" right](1,0)
\class["c" {right = 1em}](1,1)
\end{sseqpage}

run off = \langle start tip \rangle--\langle end tip \rangle
run off struct lines = \langle start tip \rangle--\langle end tip \rangle
run off differentials = \langle start tip \rangle--\langle end tip \rangle

(Initially \ldots \ldots)

Change the default behavior of run off edges for either all edges, just structure lines, or just differentials respectively. Local arrowhead options override this.

If an edge runs off the edge of the clipping, SPECTRALSEQUENCES automatically add an arrowhead to indicate that the edge continues. This option controls which arrow head is added if the start or end of an edge runs off the page.
**class label handler = \langle function \rangle**

The value of **class label handler** is a function that is applied to all labels before displaying them. It should put its output into \result. This is intended to help with code reusability. Because these handlers may crash or have annoying side-effects on some input, you may want to toggle the value of this command on and off. To turn this off for the rest of the current spectral sequence you can say `\sseqset{class label handler = {}}`. You can also use the **class label handler** key in a {scope}.

The main function **SPECTRALSEQUENCES** provides for use here is `\SseqNormalizeMonomial{#1}`. This makes it convenient to translate expressions with polynomial labels. You can write your own handlers if your \TeX programming skills are sufficient. Let me know if there are any other functions that you want here, and if you implement them yourself, please send me your implementation. Here is an example of a function that evaluates an arithmetic expression:

Here’s an example using `\SseqNormalizeMonomial`:

Here is another example which demonstrates a useful idiom for drawing Serre spectral sequences. For a more complete example, see `example_KF3n.tex`. Note the use of braces in `\{Sq^1\iota_2\}`. Without braces, `\SseqNormalizeMonomial` will simplify $Sq^1xSq^2x$ into $S^2q^3x^2$, which is obviously undesirable, so the correct way to input this is `\{Sq^1x\}\{Sq^2x\}`. Unfortunately, \foreach strips a pair of braces from its arguments, so you need to put two pairs of braces.
The value of \texttt{class name handler} is a function that is applied to all names before using them. It should put its output into \texttt{result}. The main functions intended for use here are \texttt{\SseqAHSSNameHandler} and \texttt{\SseqNormalizeMonomial}, though you can make your own. This is applied both when you name the class, as in \texttt{\class[name=a](\meta{coordinate})} and when you refer to a class by name, as in \texttt{\structline(a)(b)}. One advantage if you are using \texttt{\SseqNormalizeMonomial} is that you can ensure that $xy$ and $yx$ refer to the same class:

```
\begin{\sseqpage}[class name handler = \SseqNormalizeMonomial, classes = { show name = {right,pin} }] 
\class[name = xy](0,0) \structure[(0,1)](yxy)
\end{\sseqpage}
```

The next page is an example drawing the Atiyah Hirzebruch spectral sequence computing $KO^*\mathbb{RP}^8$.

\begin{verbatim}
\class name prefix = \texttt{prefix} \\
\class name postfix = \texttt{postfix}
\end{verbatim}

These keys add respectively a prefix or a postfix to all names of all classes in the scope. This is applied both when you name the class, as in \texttt{\class[name=a](\meta{coordinate})} and when you refer to a class by name, as in \texttt{\structline(a)(b)}. The prefix and postfix are added to the command before the \texttt{class name handler} is applied, so if you say \texttt{name=a(\texttt{the name})} then the name actually given to the class is the value of \texttt{result} after saying \texttt{\classnamehander\{\texttt{name prefix}\texttt{the name}\texttt{name postfix}\}}

## 7 Global Options

These options can only be set at the beginning of a \texttt{\sseqdata} or \texttt{\sseqpage} environment. When it makes sense, you can also set a default value using \texttt{\sseqset}. Generally, these options either modify the plot style or the logic for the spectral sequence.

\texttt{name = \texttt{seq name}}

This option must be used with the \texttt{\sseqdata} environment where it indicates the name of the spectral sequence, which will be used with the \texttt{\sseqpage} environment or \texttt{\printpage} command to draw the spectral sequence. The name used in a \texttt{\sseqdata} environment must be new unless the environment is used with the \texttt{update existing} key in which case the \texttt{\sseqdata} environment will add to the existing
An AHSS using class name handler = \SseqAHSSNameHandler:

```latex
\NewSseqCommand \KOstar {m}{
\begin{scope}[xshift = -#1, xshift = 8*\b, yshift = #1]
  \class[rectangle, name = b^{-\{b\}[0]}](0,0)
  \class[name = b^{-\{b\} e[0]}](1,0) \structline
  \class[name = b^{-\{b\} e^{-2}[0]}](2,0) \structline
  \class[rectangle, fill = none, name = b^{-\{b\} u[0]}](4,0)
\end{scope}
}

\NewSseqCommand \KOpcell {m}{
  \replacetarget \d1(b^{-\{b\}[#1]})
  \structline\begin{scope}[page = 2, dashed] (b^{-\{b\} e[0]})(b^{-\{b\} e^{-2}[#1+1]})
\end{scope}
}

\NewSseqCommand \KOetacell {m}{
  \structline\begin{scope}[page = 2, dashed] (b^{-\{b\} u[0]})(b^{-\{b\} u[4]})
\end{scope}
\}
\begin{sseqdata}[name = KO^*RP^8, Adams grading, class name handler = \SseqAHSSNameHandler, classes = fill, x range = {-8}{10}, xscale = 0.7, yscale = 0.5, x tick step = 4]
  \foreach \b in {-2,...,2}{\KOstar{\b}}
  \foreach \n in {1,3,...,5}{\KOpcell{\n}}
  \foreach \n in {2,3}{\KOetacell{\n}}
\end{sseqdata}
\printpage[\name = KO^*RP^8, title = AHSS for $KO^*\mathbb{R}P^8$] \vskip6pt
\printpage[\name = KO^*RP^8, page = 6]
```

40
spectral sequence. It is optional when used with \{sseqpage\}, and if included the name given must be the name of an existing spectral sequence.

\[\text{page} = \langle \text{page number} \rangle \text{--} \langle \text{page max} \rangle \text{ (initially 0)}\]

This key is for \{sseqpage\} and \printpage. It specifies which page of the spectral sequence is to be printed. On page \(r\), all \texttt{classes} that are not hit by differentials on pages less than \(r\) will be printed, as well as all \texttt{structlines} whose source and target classes are both printed on page \(r\), and all differentials of length exactly \(r\). The special value \texttt{page} = 0 prints all classes, differentials, and structure lines.

If you use this key with a range, then all differentials in that range of lengths will be drawn. The larger number only changes which differentials are drawn. If you use 0 for the \langle \text{page max} \rangle, then all differentials longer than \langle \text{page number} \rangle will be drawn.

\[\text{degree} = \{\langle x \text{ degree} \rangle\}{\langle y \text{ degree} \rangle}\]

\texttt{cohomological Serre grading}
\texttt{homological Serre grading}
\texttt{Adams grading}

Specifies the degree of differentials. The \langle x \text{ degree} \rangle and \langle y \text{ degree} \rangle should both be mathematical expressions in one variable \#1 that evaluate to integers on any input. They specify the \(x\) and \(y\) displacement of a page \#1 differential. In practice, they will be linear expressions with \#1 coefficient 1, -1, or 0.

The \texttt{degree} option must be given before placing any differentials. It can be specified at the beginning of the \{sseqdata\} environment, at the beginning of the \{sseqpage\} environment if it is being used as a standalone page, or as a default by saying \texttt{sseqset(degree = \{\langle x \text{ degree} \rangle\}{\langle y \text{ degree} \rangle}\}} or \texttt{sseqset(Adams grading)\}} outside of the \{sseqdata\} and \texttt{SPECTRALSEQUENCES\} environments.

You can make a named grading convention by saying \texttt{sseqset(my grading/.sseq grading = \{\langle x \text{ degree} \rangle\}{\langle y \text{ degree} \rangle}\}}. Then later passing \texttt{my grading} to a spectral sequence is equivalent to saying \texttt{degree = \{\langle x \text{ degree} \rangle\}{\langle y \text{ degree} \rangle}\}}. The following grading conventions exist by default:

% equivalent to degree = \{\#1\}(1-\#1):
\begin{sseqpage}\[ cohomological Serre grading \]
\texttt{\class(0,1)}
\texttt{\class(2,0)}
\texttt{\d2(0,1)}
\end{sseqpage}

% equivalent to degree = {-\#1}(\#1-1):
\begin{sseqpage}\[ homological Serre grading \]
\texttt{\class(0,1)}
\texttt{\class(2,0)}
\texttt{\d2(2,0)}
\end{sseqpage}

% equivalent to degree = {-1}{\#1}:
\begin{sseqpage}\[ Adams grading \]
\texttt{\class(0,2)}
\texttt{\class(1,0)}
\texttt{\d2(1,0)}
\end{sseqpage}
strict degree
lax degree

If the degree is strict, then \texttt{BPX} will throw an error if you try to specify a differential that doesn’t have the proper grading. The degree is strict by default.

\begin{verbatim}
\begin{sseqdata} [ name = \texttt{laxdegree}, Adams grading ]
\class(0,2)
\class(1,0)
\d3(1,0)(0,2) \% Error: differential does not respect grading.
\% Target should be in position (0,3) but instead it is (0,2)...
\end{sseqdata}
\end{verbatim}

update existing

This key is only for the \texttt{sseqdata} environment. It specifies that the current \texttt{sseqdata} environment is adding data to an existing spectral sequence. If you don’t pass this key, then giving a \texttt{sseqdata} environment the same \texttt{name} as a different \texttt{sseqdata} environment will cause an error. This is intended to help you avoid accidentally reusing the same name.

\begin{verbatim}
\begin{sseqdata} [ name = \texttt{laxdegree}, Adams grading, lax degree, yscale = 0.6 ]
\class(0,2)
\class(1,0)
\d3(1,0)(0,2) \% No error because degree checking is off
\end{sseqdata}
\end{verbatim}

keep changes = \texttt{boolean}

This option is only for the \texttt{sseqpage} environment, and only works when a \texttt{name} is provided. This option specifies that all of the commands in the current \texttt{sseqpage} environment should be carried forward to future pages of the same named spectral sequence. For example:
Note that the orange classes and differential do not persist because the \texttt{keep changes} option is not set in the first \texttt{sseqpage} environment, but the blue classes and differential do, since the \texttt{keep changes} option is set in the second \texttt{sseqpage} environment.

\textbf{no differentials}

\textbf{draw differentials}

The option \texttt{no differentials} suppresses all of the differentials on the current page, whereas \texttt{draw differentials} causes the page appropriate differentials to be drawn. This is useful for explaining how the computation of a spectral sequence goes, or if you want to display one of the edges of the spectral sequence, like in example\_KF3n\_tex.

\textbf{no struct lines}

\textbf{draw struct lines}

The option \texttt{no struct lines} suppresses all of the structure lines on the current page, whereas the option \texttt{draw struct lines} causes the page appropriate structure lines to be drawn.

\textbf{no orphan edges}

\textbf{draw orphan edges}

An edge is an “orphan” if both its source and target lie off the page. By default these are drawn, but with the option \texttt{no orphan edges} they are not. If the option \texttt{no orphan edges} has been set, \texttt{draw orphan edges} undoes it.

\textbf{class pattern} = \langle \texttt{class pattern name} \rangle

This key specifies the arrangement of multiple classes at the same coordinate. The default value is \texttt{standard}.
You can add new class patterns using \texttt{\textbackslash sseqnewclasspattern}:\[
\texttt{\textbackslash sseqnewclasspattern\{\langle class pattern name\rangle\}\{\langle offsets\rangle\}}
\]

Creates a new class pattern. For example, the \texttt{linear} class pattern is created using the command:\[
\texttt{\textbackslash sseqnewclasspattern\{linear\}\{(0,0); (-0.13,0)(0.13,0); (-0.2,0)(0,0)(0.2,0); (-0.3,0)(-0.1,0)(0.1,0)(0.3,0); (-0.4,0)(-0.2,0)(0,0)(0.2,0)(0.4,0); (-0.5,0)(-0.3,0)(-0.1,0)(0.1,0)(0.3,0)(0.5,0);\}}
\]

For instance the third row indicates that if there are three classes at the position \((x,y)\) they should be printed at \((x-0.2,y)\), \((x,y)\), and \((x+0.2,y)\). You can give as many rows as you like; \texttt{SPECTRALSEQUENCES} will throw an error if there are more classes in any position than the maximum number that your class pattern can handle – for instance, the \texttt{linear} class pattern can handle up to six classes based on this definition.

\textbf{class placement transform} = \{\langle transform keys\rangle\}

\textbf{add class placement transform} = \{\langle transform keys\rangle\}

The option \texttt{class placement transform} allows the user to specify a \texttt{TikZ} coordinate transform to adjust the relative position of multiple nodes in the same \((x,y)\) position. The \texttt{class placement transform} key overrides the previous value of transformations, the \texttt{add class placement transform} just adds the new transformation to the end of the list. This coordinate transform can only involve rotation and scaling, no translation. Specifying a scaling factor helps if the nodes are too large and overlap. In some cases a rotation makes it easier to see which class is the target of a differential.

With multiple large class labels, the best option is to arrange the classes vertically:
7.1 Global coordinate transformations

Of the normal TikZ coordinate transformations, only the following can be applied to a SPECTRALSEQUENCES chart:

- **scale** = \{factor\}
- **xscale** = \{factor\}
- **yscale** = \{factor\}
- **xmirror**
- **ymirror**

Scale the chart by \{factor\}. Under normal circumstances, you can tell TikZ to mirror a chart by saying, for instance, \texttt{xscale = -1}, but SPECTRALSEQUENCES needs to be aware that the chart has been mirrored in order to draw the axes correctly. Thus, if you want to mirror a spectral sequence, use the \texttt{xmirror} and \texttt{ymirror} options as appropriate.

**rotate** = \{angle\}

It probably won’t look great if you pick an angle that isn’t a multiple of 90 degrees.

7.2 Plot options and axes style

- **x range** = \{\langle x \text{ min} \rangle \{x \text{ max}\}\}
- **y range** = \{\langle y \text{ min} \rangle \{y \text{ max}\}\}

These options set the x range (respectively y range) to be a specific interval. By default, if no range is specified then the range is chosen to fit all the classes. If an x range is specified but no y range, then the y range is chosen to fit all the classes that lie inside the specified x range, and vice versa. The values must be integers – if you want to extend the x axis a noninteger amount, try using \texttt{x axis start extend} or \texttt{x axis end extend}.

- **grid** = \{grid type\}
- **grid color** = \{color\}
- **grid step** = \{positive integer\}
- **x grid step** = \{positive integer\}
\begin{sseqdata} \name = grid example, scale = 0.8 \end{sseqdata}

It is possible to make your own grid type by defining the command \texttt{$\textbackslash$seq@grid@yourgridname} to draw a grid.

It is a known problem that Acrobat reader displays grids very dark and thick. This is caused by the enhance thin lines “feature” of Acrobat reader. You can turn it off globally by going into the Acrobat reader preferences window and unchecking the “Enhance thin lines” checkbox in the “Rendering” section. Unfortunately, there is no way to instruct Acrobat Reader to not apply this feature to a particular document. This has apparently been annoying graphic designers for over a decade.

The grid color option changes the color of the grid; the default value is \texttt{gray}. The grid step keys change the grid step.

title = \texttt{\{text\}}
title style = {keys}
x label = {text}
y label = {text}
x label style = {keys}
y label style = {keys}
label style = {keys}

This make chart labels.

\begin{sseqpage}
\title{An example},
yscale = 0.5,
x label = {x axis label},
y label = {y axis label},
label style = {blue, font = \small},
x label style = {yshift = 5pt},
\end{sseqpage}

Note that if you make multiple versions of the same chart and some of the charts have labels and others don’t, they might not align the way you want. An easy solution to this is to use \path[background] (min x, min y) - (max x, max y); where the coordinates are below and to the left, respectively above and to the right, of everything else drawn in any picture. This makes the bounding boxes for all of the pictures the same size, so that they line up even if the exact collection of things drawn changes. See the example at the end of the background key for an illustration of this.

no title
draw title
no x label
draw x label
no y label
draw y label
no labels
draw labels

draw x ticks
draw y ticks
draw ticks

draw ticks

Suppress or unsuppress the title, x label, y label, or both x and y labels, respectively.

no x ticks
draw x ticks
no y ticks
draw y ticks

Suppress axes ticks (the numbers next to the axes). Only matters if axes are drawn. You can make your own ticks using Ti\textit{KZ} inside a \{scope\} environment with the background key. For instance, you might want to label the axes as 0, n, 2n, ... You can achieve this as follows: (you can also use x tick handler).

\begin{sseqpage}
\no x ticks, x range = (0)(3)
\begin{scope}[background]
\node at (0,\ymin - 1) (0);
% \phantom is fragile so we have to throw in an extra \protect
\node at (1,\ymin - 1) (\protect\phantom{2}n);
\foreach n in {2,..., \xmax}(\protect\phantom{\n}){
\node at (\n,\ymin - 1) {\n n};
}
\end{scope}
\end{sseqpage}
x tick step = (positive integer)  (initially 1)
y tick step = (positive integer)  (initially 1)
tick step = (positive integer)  (initially 1)

Sets the interval between labels.

x tick offset = (integer)  (initially 0)
y tick offset = (integer)  (initially 0)
tick offset = (integer)  (initially 0)

Sets the label offset – by default the ticks will always be the set of numbers that are 0 mod \(\langle\text{tick step}\rangle\). Change it so that the ticks are the set of numbers that are \(\langle\text{tick offset}\rangle\mod\langle\text{tick step}\rangle\).

x major tick step = (nonnegative integer)  (initially 0)
y major tick step = (nonnegative integer)  (initially 0)
major tick step = (nonnegative integer)  (initially 0)
x minor tick step = (nonnegative integer)  (initially 0)
y minor tick step = (nonnegative integer)  (initially 0)
minor tick step = (nonnegative integer)  (initially 0)

If these are nonzero, they control the placement of tick marks on the axes. The value 0 prevents tick marks from being drawn. Uses \texttt{x tick offset} as an offset.

x tick style = \{\langle\text{keys}\rangle\}
y tick style = \{\langle\text{keys}\rangle\}
tick style = \{\langle\text{keys}\rangle\}

Change the tick style:

x tick handler = \{\langle\text{function}\rangle\}  (initially \#1)
y tick handler = \{\langle\text{function}\rangle\}  (initially \#1)
tick handler = \{\langle\text{function}\rangle\}  (initially \#1)

The value for \texttt{x tick handler} should be a function that takes in the current \texttt{x} value and outputs the appropriate tick. Correspondingly with \texttt{y tick handler}. The \texttt{tick handler} key sets both.
7.3 Layout

\textbf{x axis type} = \langle type \rangle \hspace{1cm} \text{Initially \textit{border}}
\textbf{y axis type} = \langle type \rangle \hspace{1cm} \text{Initially \textit{border}}
\textbf{axes type} = \langle type \rangle \hspace{1cm} \text{Initially \textit{border}}

\textit{no x axis} \hspace{1cm} \text{no y axis} \hspace{1cm} \text{no axes}

The \langle type \rangle is either \textit{border}, \textit{center}, \textit{frame}, or \textit{none}. \textit{no axes} is a shorthand for \textit{axes type=none}. The \textit{border} type is the default and puts the axes on the bottom and left of the picture. The \textit{center} type by default places the axes to pass through \((0, 0)\); this can be modified using the \textit{x axis origin} and \textit{y axis origin} keys. See example\_KRAHSS.tex and example\_KUHFPSS.tex for examples where this is used. The \textit{frame} type draws a frame around the entire chart:

\begin{sseqpage}
axes type = frame
\end{sseqpage}

\textbf{x axis origin} = \langle x value \rangle \hspace{1cm} \text{Initially 0}
\textbf{y axis origin} = \langle y value \rangle \hspace{1cm} \text{Initially 0}

If you use \textit{axes type} = \textit{center}, these keys change the position of the axes. Otherwise, they are ignored.

\begin{sseqpage}[x range = \{(0)(4), yscale = 1.78, x tick handler = {
  \ifnum\$1 = 0\relax
    0
  \else
    \ifnum\$1 = 1\relax
      \protect\vphantom{2}n
    \else
      #1n
    \fi
  \fi
}
\]}
\end{sseqpage}

\textbf{x axis gap} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{y axis gap} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{axes gap} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}

\textbf{x tick gap} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{y tick gap} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}

\textbf{x axis extend start} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{y axis extend start} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{x axis extend end} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{y axis extend end} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.5cm}
\textbf{x axis tail} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.9cm}
\textbf{y axis tail} = \langle dimension \rangle \hspace{1cm} \text{Initially 0.9cm}
x axis clip padding = \langle \text{dimension} \rangle \quad \text{(initially 0.1cm)}
\n\begin{align*}
\text{y axis clip padding} &= \langle \text{dimension} \rangle \quad \text{(initially 0.1cm)} \\
\text{right clip padding} &= \langle \text{dimension} \rangle \quad \text{(initially 0.1cm)} \\
\text{left clip padding} &= \langle \text{dimension} \rangle \quad \text{(initially 0.1cm)} \\
\text{top clip padding} &= \langle \text{dimension} \rangle \quad \text{(initially 0.1cm)} \\
\text{bottom clip padding} &= \langle \text{dimension} \rangle \quad \text{(initially 0.1cm)}
\end{align*}

\text{custom clip} = \langle \text{clip path} \rangle

Give a custom clipping. The clipping specified must be in the form of a valid Ti\textit{k}Z path, for instance \texttt{\textbackslash clip (0,0) rectangle (10,10);}. This clipping is also applied to any grid and is used to draw ellipses on appropriate differentials or structure lines that go out of bounds and to determine whether a differential or structure line is an “orphan”. It is not applied to any background elements, which is important because these are often used for axes labels and such that should lie outside of the clipping region. Weird things can happen with out of range edges if you provide an oddly shaped path.

\text{clip} = \langle \text{boolean} \rangle \quad \text{(default true) (initially true)}

If this is false the spectral sequence chart won’t be clipped. I’m not really sure why you would want that, but there might be some use case. Setting this to be false is not fully supported, and it’s possible that weird things will happen with some of the edges that go out of range.

\text{rotate labels} = \langle \text{boolean} \rangle \quad \text{(default true) (initially false)}

If you use \texttt{rotate = 90} but also want the labels rotated (so that the whole chart is sideways) use this key.
Axes type center layout

- x tick gap
- y tick gap
- left clip padding
- bottom clip padding
- top clip padding
- y axis extend end
- x axis extend end
- x axis extend start
- y axis extend start
- right clip padding

52