Sseqpages

Hood Chatham
hood@mit.edu

version 1.0
2017/5/29

The sseqpages package is a specialized tool built on top of Ti*k*Z for drawing spectral sequences. It provides a powerful, concise syntax for specifying the data of a spectral sequence, and then allows the user to print various pages of 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, sseqpages is extremely flexible. It is closely integrated with Ti*k*Z to ensure that users can take advantage of as much as possible of its expressive power. It is possible to turn off most of the automated layout features and draw replacements using Ti*k*Z commands. sseqpages also has a carefully designed error reporting system intended to ensure that it is as clear as possible what is going wrong.

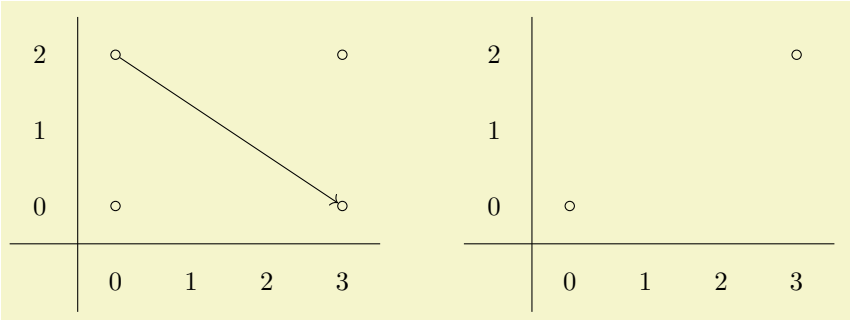
Many thanks to the authors of Ti*k*Z for producing such a wonderful package with such thorough documentation. I would have needed to spend a lot more time reading the Ti*k*Z code if the documentation weren't so excellent. I took ideas or code or both from tikzcd (the code for handling labels), pgfplots (axes labels), and sseq (the grid styles, the stack). I also lifted a fair amount of code from tex stack exchange. Thanks to Eric Peterson for being a very early adopter and reporting many bugs, and to Vishal Arul and Catherine Ray for helpful suggestions.

Contents

1 Introduction

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

Here is a basic example:




```
\begin{sseqdata}[name=ex1,cohomological Serre grading]
\class(0,0)
\class(0,2)
\class(3,0)
\class(3,2)
\d3(0,2)
\end{sseqdata}
\printpage[name=ex1,page=3]\hskip1cm
\printpage[name=ex1,page=4]
```

`\begin{sseqdata}[name=ex1,degree={#1}{1-#1}]` starts the declaration of the data of a spectral sequence named `ex1` whose page `r` differentials go `r` to the right and down `r-1` (this is cohomological Serre grading). Then we specify four classes and one page 3 differential, and we ask sseqpages 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.

2 The Environments

`\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 sseqpages 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 standalone 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.

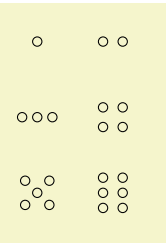
`\printpage[`⟨*options*⟩`]`

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*⟩`)`

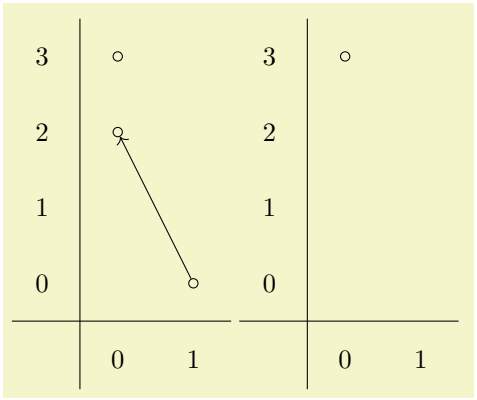
This places a class at `(x,y)` where `x` and `y` are integers. If multiple classes occur at the same position, sseqpages will automatically arrange them in a pre-specified pattern. This pattern may be altered using the `class pattern` option.



```
\begin{sseqpage}[no axes,ymirror]
\class(0,0)
\class(1,0)\class(1,0)
\class(0,1)\class(0,1)\class(0,1)
\class(1,1)\class(1,1)\class(1,1)\class(1,1)
\class(0,2)\class(0,2)\class(0,2)\class(0,2)\class(0,2)
\class(1,2)\class(1,2)\class(1,2)\class(1,2)\class(1,2)\class(1,2)
\end{sseqpage}
```

The effect of the `\class` command is to print a Ti*k*Z node on a range of pages. Any option that would work for a Ti*k*Z `\node` command will also work in the same way for the `\class`, `\replaceclass`, and `\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:

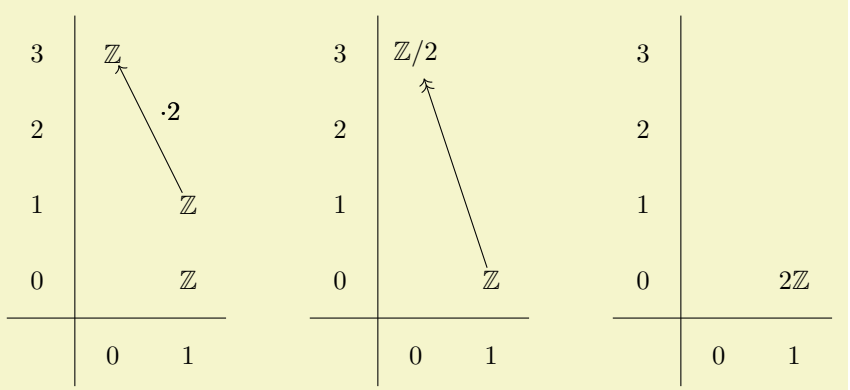



```
\begin{sseqdata}[name=class example,Adams grading]
\class(1,0)
\class(0,2)
\class(0,3)
\d2(1,0)
\end{sseqdata}
\printpage[name=class example,page=2]
\printpage[name=class example,page=3]
```

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

`\replaceclass[`⟨*options*⟩`](`⟨*x*⟩`,`⟨*y*⟩`,`⟨*n*⟩`)`
`\replaceclass[`⟨*options*⟩`](`⟨*classname*⟩`)`

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 `\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` *n*. By default, this command will affect the first class placed in that position. Alternatively, you can provide the `name` of a class.



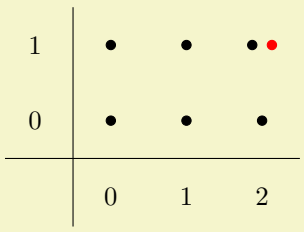

```
\begin{sseqdata}[name=replace class example,Adams grading,classes={draw=none},math nodes]
\class["\mathbb{Z}"](0,3)
\class["\mathbb{Z}"](1,1)
\class["\mathbb{Z}"](1,0)
\d["\cdot 2"]2(1,1)
\replaceclass["\mathbb{Z}/2"](0,3)
\d[->]3(1,0)
\replaceclass["2\mathbb{Z}"](1,0)
\end{sseqdata}
\printpage[name=replace class example, page=2]
\hskip1cm
\printpage[name=replace class example, page=3]
\hskip1cm
\printpage[name=replace class example, page=4]
```

Note that this will not restore any structlines coming into or off of the class, if you want the `\structlines` to persist, you must call `\structline` again (or use the structline `page` option)

`\classoptions`[⟨*options*⟩](⟨*x*⟩,⟨*y*⟩,⟨*n*⟩)
`\classoptions`[⟨*options*⟩](⟨*classname*⟩)

This adds options to a class that already exists. This can be used in a `sseqpage` environment to modify the appearance of a class for just one drawing of the spectral sequence, for instance to highlight it for discussion purposes.

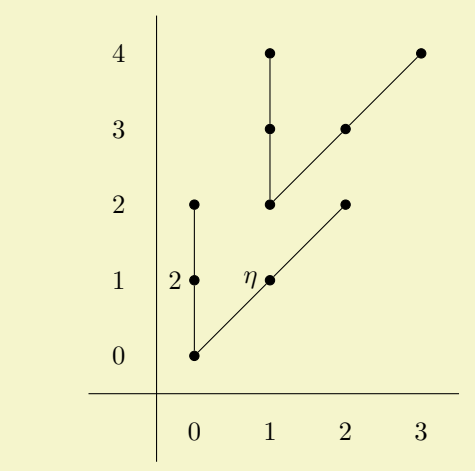
If there are multiple classes at the coordinate `(x,y)` you may specify which using an integer or a `tag` n. By default, this command will affect the first class placed in that position. Alternatively, you can provide the `name` of a class.



The red class is the problem

```
\begin{sseqdata}[name=class options example,classes=fill]
\class(2,1)
\foreach \x in {0,...,2} \foreach \y in {0,1}{
    \class(\x,\y)
}
\end{sseqdata}
\begin{sseqpage}[name=class options example,right clip padding=0.6cm]
\classoptions[red](2,1,2) % Will only show up as red on this page!
\node[background,text width=10em] at (0.3,-2.2)
    {\textup{The red class is the problem}};
\end{sseqpage}
```

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 `\sseqnewgroup`:



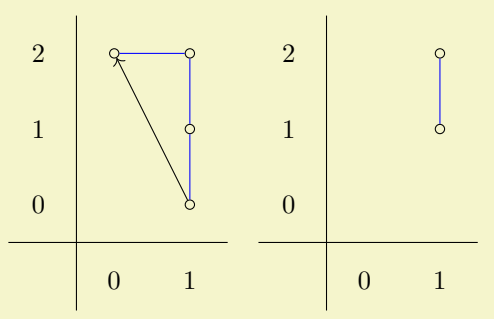

```
\sseqnewgroup\mygroup{
    \class(0,0)
    \class(0,1)
    \class(0,2)
    \class(1,1)
    \class(2,2)
    \structline(0,0)(0,1)
    \structline(0,1)(0,2)
    \structline(0,0)(1,1)
    \structline(1,1)(2,2)
}
\begin{sseqpage}[classes=fill,class labels={left=0.3em},math nodes]
\mygroup(0,0)
\mygroup(1,2)
\classoptions["2"](0,1)
\classoptions["\eta"](1,1)
\end{sseqpage}
```

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

\d[⟨*options*⟩]⟨*page*⟩(⟨*x*⟩,⟨*y*⟩,⟨*sourcen*⟩,⟨*targetn*⟩)
\d[⟨*options*⟩]⟨*page*⟩(⟨*sourcename*⟩,⟨*targetn*⟩)
\d[⟨*options*⟩]⟨*page*⟩(⟨*sourcecoord*⟩)(⟨*targetcoord*⟩)

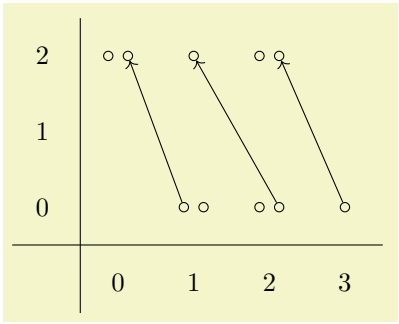
Calling \d⟨*page*⟩(⟨*x*⟩,⟨*y*⟩) creates a differential starting at (⟨*x*⟩,⟨*y*⟩) of length determined by the specified page. In order to use the \d command like this, you must first specify the degree of the differentials as an option to the sseqdata or sseqpage environment. The degree indicates how 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 \structlines connected to the source and target of the differential disappear.

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 ⟨*sourcen*⟩ or ⟨*targetn*⟩. 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 (⟨*x*⟩,⟨*y*⟩,⟨*n*⟩). Using \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 sseqpages 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 lax degree option.



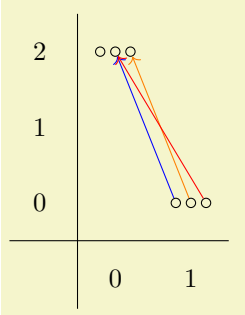

```
\begin{sseqdata}[name=d example,degree={-1}{#1},
                 struct lines=blue]
\class(0,2)
\class(1,2)
\class(1,1)
\class(1,0)
\structline(1,2)(0,2)
\structline(1,2)(1,1)
\structline(1,1)(1,0)
\d2(1,0)
\end{sseqdata}
\printpage[name=d example,page=2]
\hskip0.3cm
\printpage[name=d example,page=3]
```

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: `\d`⟨*page*⟩`(`⟨*x*⟩`,`⟨*y*⟩`,`⟨*source n*⟩`,`⟨*target n*⟩`)`




```
\begin{sseqpage}[Adams grading]
\class(1,0)\class(1,0)
\class(0,2)\class(0,2)
\d2(1,0,1,2)
\class(2,0)\class(2,0)
\class(1,2)
\d2(2,0,2)
\class(3,0)
\class(2,2)\class(2,2)
\d2(3,0,,2)
\end{sseqpage}
```

Negative indices will count from the most recent class in the coordinate (so `-1` is the most recent, `-2` is the second most recent, etc):




```
\begin{sseqpage}[Adams grading]
\class(1,0)
\class(0,2)\class(0,2)
\d[blue]2(1,0,-1,-1)
\class(1,0)
\class(0,2)
\d[orange]2(1,0,-1,-1)
\class(1,0)
\d[red]2(1,0,-1,-2)
\end{sseqpage}
```

You can also use a `tag`.

`\doptions`[⟨*options*⟩]⟨*page*⟩(⟨*x*⟩,⟨*y*⟩,⟨*sourcen*⟩,⟨*targetn*⟩)
`\doptions`[⟨*options*⟩]⟨*page*⟩(⟨*sourcename*⟩,⟨*targetn*⟩)
`\doptions`[⟨*options*⟩]⟨*page*⟩(⟨*sourcecoord*⟩)(⟨*targetcoord*⟩)

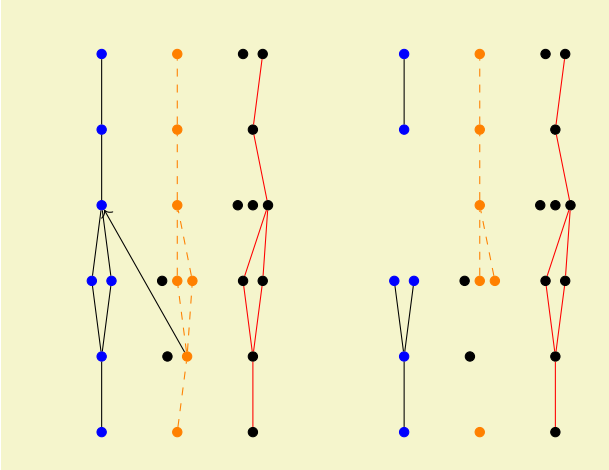
This command adds options to an existing differential, just like `\classoptions` except for differentials. Its syntax is identical to that of `\d`.

`\structline`[⟨*options*⟩]⟨*source coordinate*⟩⟨*target coordinate*⟩

This command creates a structure line from ⟨*source coordinate*⟩ to ⟨*target coordinate*⟩. The source and target coordinates are either of the form (⟨*x*⟩,⟨*y*⟩,⟨*n*⟩) or are a ⟨*classname*⟩. If there are multiple classes at (x,y), then ⟨*n*⟩ specifies which of the classes at (x,y) the structline 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 `tag` for n.

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 `\replaceclass`), then the `\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 `page` option to change it.



```
\sseqnewgroup*\tower{
    \class(0,0)
    \class(0,2)
    \foreach \y in{1,...,5}{
        \class(0,\y)
        \structline(0,\y-1,-1)(0,\y,-1)
    }
    \structline(0,1,-1)(0,2,-2)
    \structline(0,2,-2)(0,3,-1)
}
\begin{sseqdata}[name=structline example,
                 classes={circle,fill},
                 Adams grading, no axes]
\class(1,1)\class(1,2)
\class(2,3)\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]
\hskip1cm
\printpage[name=structline example,page=3]
```

`\structlineoptions`[⟨*options*⟩]⟨*source coordinate*⟩⟨*target coordinate*⟩

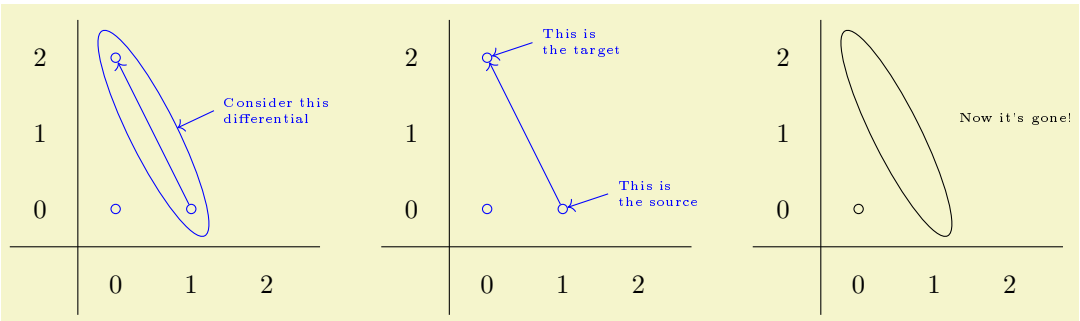
This command adds options to an existing structure line, just like `\classoptions` except for structure lines. Its syntax is identical to `\structline`.

`\circleclasses`[⟨*options*⟩]⟨*source coordinate*⟩⟨*target coordinate*⟩

This command is a lot like `\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 `\circleclasses` to produce good results. There is no `\circleclassesoptions` command because it doesn't seem necessary and (more importantly) I didn't feel like making one. Maybe at some later date I'll put one in.

`\draw`
`\path`
`\node`
`\clip`

Any code that would work in a `tikzpicture` environment will also work unchanged in a `sseqdata` or `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}[name=tikz example,Adams grading,x range={0}{2}, x axis extend end=2em,math nodes=false]
\class(0,0)
\class(1,0)
\class(0,2)
\d2(1,0)
\end{sseqdata}
%
\begin{sseqpage}[name=tikz example,sseq={blue,font=\tiny}]
\circleclasses[name path=myellipse,inner sep=3pt,ellipse ratio=1.6] (1,0)(0,2)
\path[name path=myline](1.2,1.2)--(0.6,1);
\draw[->,name intersections={of=myellipse and myline}] (1.3,1.3)--(intersection-1);
\node[right,text width=1.6cm] at (1.3,1.3) {Consider this differential};
\end{sseqpage}
\qquad
\begin{sseqpage}[name=tikz example,sseq={<-,blue,font=\tiny}]
\draw[xshift=1](0,0) to (0.6,0.2) node[right,text width=1.1cm] {This is the source};
\draw[yshift=2](0,0) to (0.6,0.2) node[right,text width=1.1cm] {This is the target};
\end{sseqpage}
\qquad
\begin{sseqpage}[page=3,name=tikz example]
\circleclasses[name path=myellipse,inner sep=3pt,ellipse ratio=1.6] (1,0)(0,2)
\node[right,font=\tiny] at (1.2,1.2) {Now it's gone!};
\end{sseqpage}
```

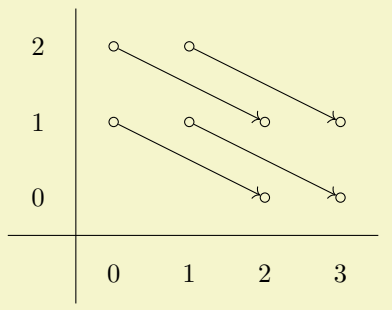
4 Options for the main commands

4.1 Universal options

The following options work with all of the drawing commands in this package, including `\class`, `\d`, and `\structline`, their friends `\replaceclass`, `\classoptions`, `\doptions`, and `\structlines`, as well as with Ti*k*Z primitives.

`xshift`=⟨*integer*⟩ (no default)
`yshift`=⟨*integer*⟩ (no default)

Shifts by integer values are the only coordinate changes that are allowed to be applied to `\class`, `\d`, `\structline`, their relatives, or to a `scope` environment that contains any of these commands. These shift commands help with reusing code. For instance:




```
\begin{sseqpage}[cohomological Serre grading]
\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}
```

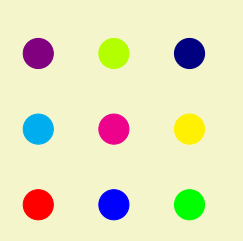
This code segment is very useful so sseqpages has the command `\sseqnewgroup` which to make code like this more convenient. The following code produces the same output as above:

```
\sseqnewgroup\examplegroup{
    \class(2,0)
    \class(0,1)
    \d2(0,1)
}
\begin{sseqpage}
\examplegroup(0,0)
\examplegroup(0,1)
\examplegroup(1,0)
\examplegroup(1,1)
\end{sseqpage}
```

A word of warning: the behavior of `xshift` in sseqpages is incompatible with the normal behavior of `xshift` in Ti*k*Z. For some reason, saying `xshift=1` in Ti*k*Z does not shift the coordinate `(0,0)` to the coordinate `(1,0)` – instead it shifts by 1pt. In sseqpages , saying `xshift=1` moves the coordinate `(0,0)` to the coordinate `(1,0)`. This includes Ti*k*Z 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, so see the `color` package documentation for more information.

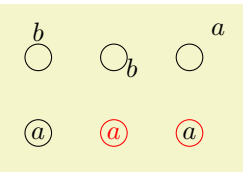


```
\begin{sseqpage}[classes={fill,inner sep=0.4em}, no axes]
\class[red](0,0)
\class[blue](1,0)
\class[green](2,0)
\class[cyan](0,1)
\class[magenta](1,1)
\class[yellow](2,1)
\class[blue!50!red](0,2) % a 50-50 blend of blue and red
\class[green!30!yellow](1,2) % 70% green, 30% yellow
\class[blue!50!black](2,2)
\end{sseqpage}
```

"⟨*text*⟩"⟨*options*⟩

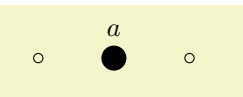
Specify a label for a class, a differential, or a structure line. This uses the Ti*k*Z quotes syntax. The options include anything you might pass as an option to a Ti*k*Z node, including arbitrary coordinate transforms, colors, opacity options, shapes, fill, draw, etc. The behavior is a little different depending on whether you use it on a class or on a differential or struct 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 Ti*k*Z node will also work for the label, including general coordinate transformations. If the placement is "inside", then the only relevant ⟨*options*⟩ are those that alter the appearance of text, such as opacity and color.



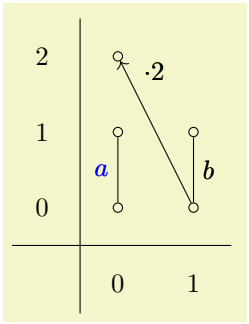

```
\begin{sseqpage}[no axes,classes={minimum width=width("a")+0.5em}]
\class["a"](0,0)
\class["a",red](1,0)
\class["a" black,red](2,0)
\class["b" above](0,1)
\class["b" {below right,yshift=0.1cm}](1,1)
\class["a" {above right={1em}}](2,1)
\end{sseqpage}
```

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.



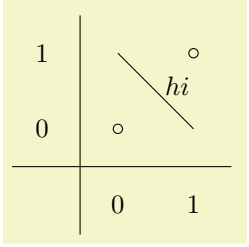

```
\begin{sseqpage}[no axes]
\class(0,0)
\class(2,0)
\node[circle,fill,"a"] at (1,0) {};
\end{sseqpage}
```

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



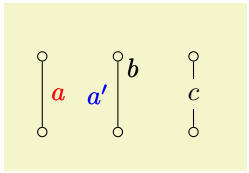

```
\begin{sseqpage}[Adams grading, math nodes]
\class(0,0)
\class(0,1)
\class(0,2)
\structline["a"' blue](0,0)(0,1)
\class(1,0)
\class(1,1)
\structline["b"](1,0)(1,1)
\d["\cdot 2"{pos=0.8}]2(1,0)
\end{sseqpage}
```

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 Ti*k*Z primitives:




```
\begin{sseqpage}
\class(0,0) \class(1,1)
\draw (1,0) to["hi"'{yshift=-0.5em}] (0,1);
\end{sseqpage}
```

The special option "description," stolen from `tikzcd`, places the label on top of the edge. In order to make this option work correctly, if the background coolor is not the default white, you must inform sseqpages about this using the key `background color=`⟨*color*⟩. In this document, the background color is called *graphicbackground*.



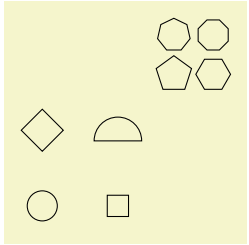

```
\begin{sseqpage}[no axes,background color=graphicbackground]
\foreach\x in {0,1,2} \foreach\y in {0,1}{
    \class(\x,\y)
}
\structline["a" red](0,0)(0,1)
\structline["a'"'blue,"b"{yshift=1em}](1,0)(1,1)
\structline["c" description](2,0)(2,1)
\end{sseqpage}
```

4.2 Options for \class

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

A Ti*k*Z shape

If you give the name of a Ti*k*Z shape, the class node will be of that shape. The standard Ti*k*Z shapes are `circle` and `rectangle`, but there are many more Ti*k*Z shapes in the shapes library, which you can load using the command `\usetikzlibrary{shapes}`. The following are some examples:

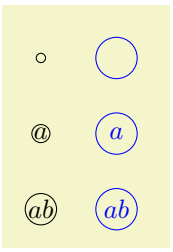


```
\begin{sseqpage}[no axes,classes={inner sep=0.4em},
                 class placement transform={scale=2}]
\class(0,0)
\class[rectangle](1,0)
\class[diamond](0,1)
\class[semicircle](1,1)
\class[regular polygon, regular polygon sides=5](2,2)
\class[regular polygon, regular polygon sides=6](2,2)
\class[regular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

See the PGF manual section on the shape library for more information.

`minimum width`=⟨*dimension*⟩ (no default)
`minimum height`=⟨*dimension*⟩ (no default)
`minimum size`=⟨*dimension*⟩ (no default)
`inner sep`=⟨*dimension*⟩ (no default)
`outer sep`=⟨*dimension*⟩ (no default)

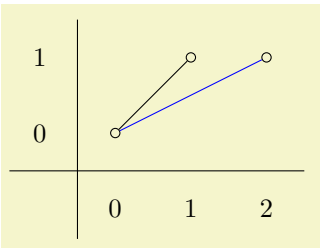
These options control the size of a node. This is typically useful to make the size of nodes consistent independent of the size of their label text. For instance:




```
\begin{sseqdata}[no axes,name=minimum width example]
\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*⟩ (no default)

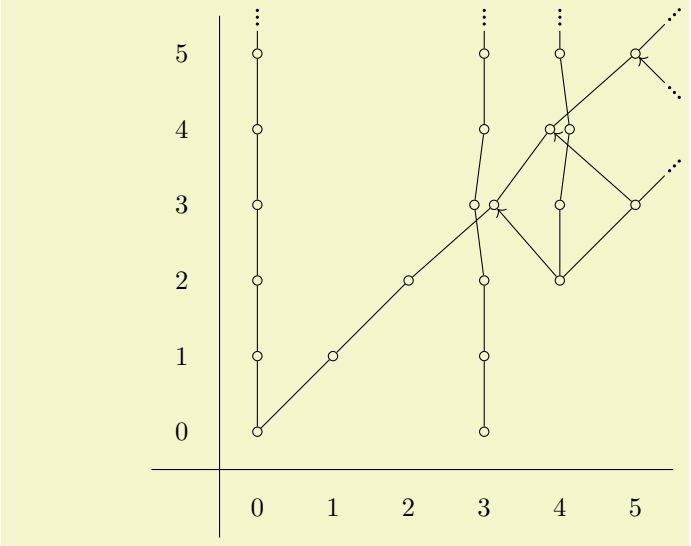
The `\class` command makes a TikZ node on appropriate pages. You can refer to this node using TikZ commands by using coordinates. Using the `name` option, you can give the node a second shorter name. One potential benefit to this is that it is immune to coordinate transformations. For example, in the following code, `xshift` does not apply to the nodes specified by `(id)` and `(eta)` but does apply to the coordinate specified by `(1,1)`:




```
\begin{sseqpage}
\class[name=id](0,0)
\class[name=eta](1,1)
\class(2,1)
\draw[xshift=1] (id) -- (eta);
\draw[xshift=1,blue] (id) -- (1,1);
\end{sseqpage}
```

`tag`=⟨*tag*⟩ (no default)

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:

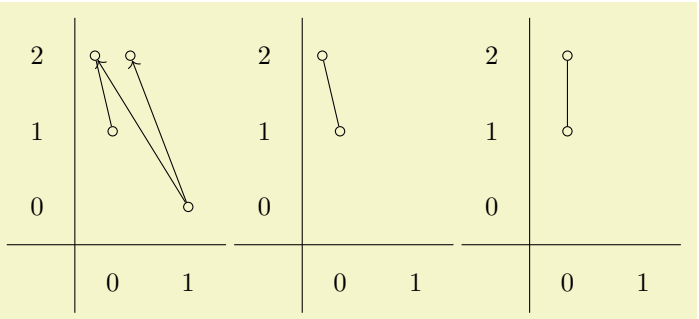


```
\sseqnewgroup*\tower{
    \class(0,0)
    \foreach\i in {1,...,11}{
        \class(0,\i)
        \structline(0,\i-1,-1)(0,\i,-1)
    }
}
\sseqnewgroup\hvee{
    \tower(0,0)
    \foreach\i in {1,...,11}{
        \class(\i,\i)
        \structline(\i-1,\i-1,-1)(\i,\i,-1)
    }
}
\begin{sseqpage}[degree={-1}{1},x range={0}{5},y range={0}{5}]
\tower(3,0)
\hvee[tag=id](0,0)
\hvee[tag=h_{21}](4,2)
\foreach \n in {0,...,5}{
    \d2(4+\n,2+\n,h_{21},id)
}
\end{sseqpage}
```

We want each differential to go from the `h_{21}` 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.

`offset`=$\{(\langle xoffset\rangle,\langle yoffset\rangle)\}$ (no default)

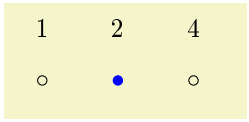
By default, a class uses the offset specified by `class pattern`. Occasionally this is undesirable. In this case, you can specify the offset for a particular class by hand. For example if the sum of two classes is hit by a differential, it looks better for the class replacing them to be centered:




```
\begin{sseqdata}[name=offset example, Adams grading,class placement transform={scale=1.8}]
\class(0,1)
\class(0,2)\class(0,2)
\draw(0,1)--(0,2);
\class(1,0)
\d2(1,0,,1)
\d2(1,0,,2)
\replaceclass(0,2)
\end{sseqdata}
\printpage[name=offset example,page=2]
\printpage[name=offset example,page=3]
\begin{sseqpage}[name=offset example,page=3]
\classoptions[offset={(0,0)}](0,2)
\end{sseqpage}
```

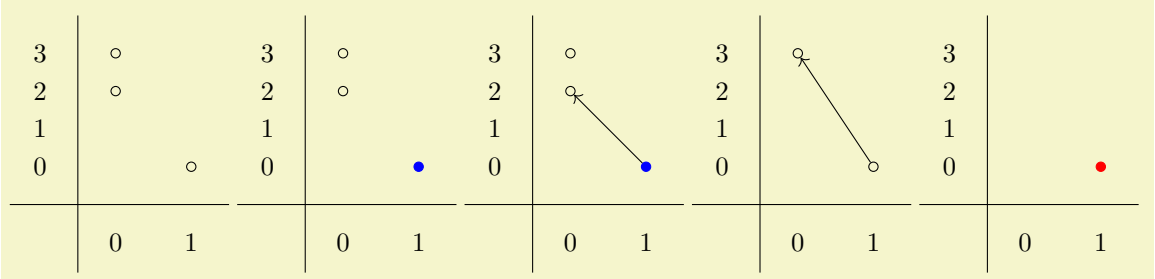
`page`=$\langle page\rangle$--$\langle page\_max\rangle$ (no default)
`generation`=$\langle generation\rangle$--$\langle generation\_max\rangle$ (no default)

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.

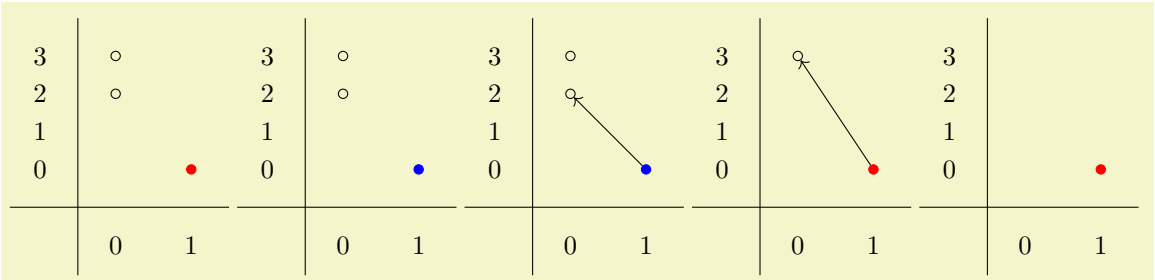



```
\begin{sseqdata}[name=page_example,no axes,
    title=\page,title style={yshift=-0.5cm}]
\class(0,0)
\classoptions[page={2--3},fill,blue](0,0)
\end{sseqdata}

\printpage[name=page_example,page=1]
\printpage[name=page_example,page=2]
\printpage[name=page_example,page=4]
```

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 `sseqdata` environment, or on the generation appropriate to the current page in a `sseqpage` 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.




```
\begin{sseqdata}[name=page_example2, Adams grading,yscale=0.5]
\class(0,2)\class(1,0)
\d2(1,0)
\replaceclass(1,0)
\class(0,3)
\d3(1,0)
\replaceclass(1,0)
\classoptions[fill,red](1,0)% (a) applies to most recent (last) generation.
\end{sseqdata}

\printpage[name=page_example2,page=1] % generation 0 of (1,0), not styled
\begin{sseqpage}[name=page_example2,page=1,keep changes]
\classoptions[fill,blue](1,0) % (b) applies to the generation present on page 1, that is, generation 0.
\end{sseqpage}
\printpage[name=page_example2,page=2] % generation 0 of (1,0), so class is blue from (b)
\printpage[name=page_example2,page=3] % generation 1 of (1,0), class is not styled
\printpage[name=page_example2,page=4] % generation 2 of (1,0), class is red from  (a)
```



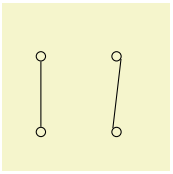

```
\begin{sseqdata}[name=page_example2, Adams grading,update existing]
\classoptions[fill,red,generation=0-- -1](1,0)% (c) applies to all generations, overwrites (b) and (a).
\end{sseqdata}

\printpage[name=page_example2,page=1]% generation 0 of (1,0), so class is red
\begin{sseqpage}[name=page_example2,page=1,keep changes]
\classoptions[fill,blue](1,0) % (d) applies to the generation present on page 1, that is, generation 0.
\end{sseqpage}
\printpage[name=page_example2,page=2] % generation 0 of (1,0), class is blue from (d)
\printpage[name=page_example2,page=3] % generation 1 of (1,0), class is red from (c)
\printpage[name=page_example2,page=4] % generation 2 of (1,0), class is red from (c)
```

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*k*Z options that you could apply to a Ti*k*Z "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*⟩ (no default)
`target anchor`=⟨*anchor*⟩ (no default)

Because you can't use the normal Ti*k*Z mechanism for specifying the source and target anchors, sseqpages has these two keys for `\d` and `\structline`:



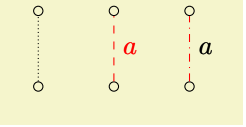
```
\begin{sseqpage}[no axes]
\foreach\x in {0,1} \foreach\y in {0,1}{
    \class(\x,\y)
}
\structline(0,0)(0,1)
\structline[source anchor=north west,target anchor=-30](1,0)(1,1)
\end{sseqpage}
```

`shorten >`=⟨*distance*⟩ (no default)
`shorten <`=⟨*distance*⟩ (no default)

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.

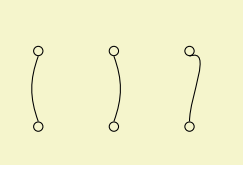
Dash patterns:

See the pgfmanual subsubsection titled "Graphic Parameters: Dash Pattern" for a complete explanation of the dash pattern related options. Some examples:




```
\begin{sseqpage}[no axes]
\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{sseqpage}
```

`bend left`=⟨*angle*⟩ (no default)
`bend right`=⟨*angle*⟩ (no default)
`in`=⟨*anchor*⟩ (no default)
`out`=⟨*anchor*⟩ (no default)



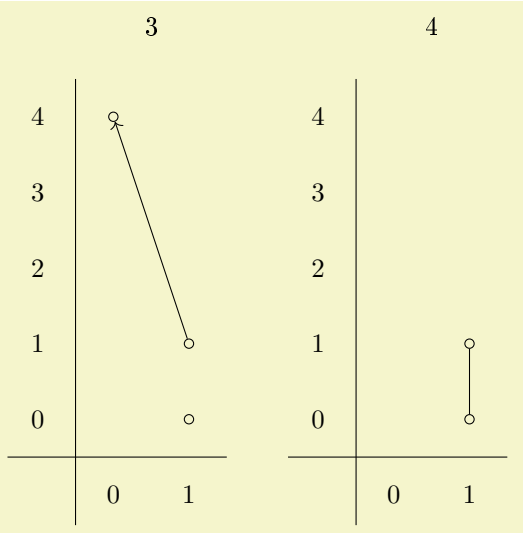
```
\begin{sseqpage}[no axes]
\foreach\x in {0,1,2} \foreach\y in {0,1}{
    \class(\x,\y)
}
\structline[bend left=20](0,0)(0,1)
\structline[bend right=20](1,0)(1,1)
\structline[in=20,out=north](2,0)(2,1)
\end{sseqpage}
```

`invisible` (no value)

This key is only for `\d`. It prevents a differential from being drawn at all. The typical reason you might want this is so that you can draw your own differential using tikz commands. See `\getdtarget` for an example of this.

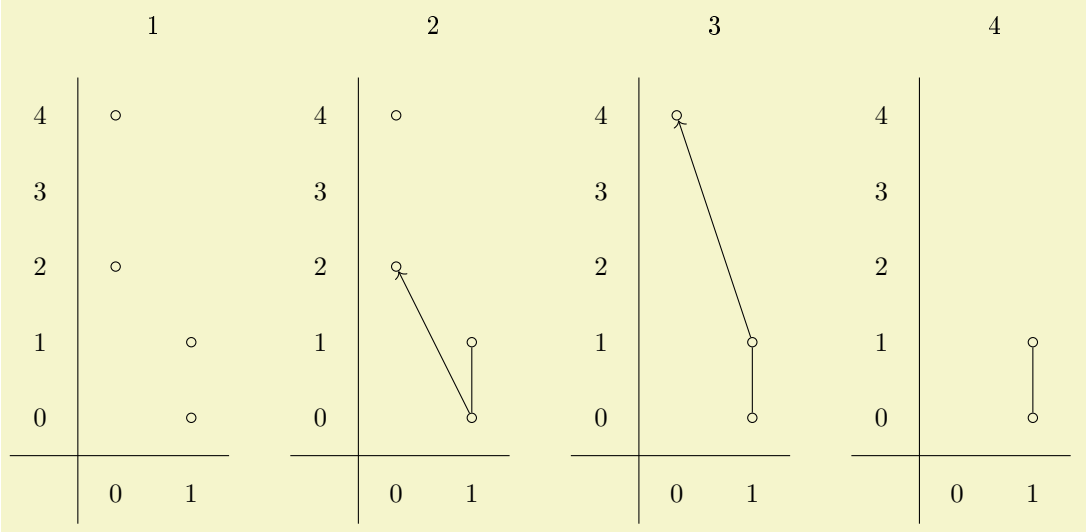
page=⟨*page*⟩--⟨*page_max*⟩ (no default)

This key is only for `\structline` and `\structlineoptions`. By default, the `\structline` command only adds a structline starting on the page where the most recent generation of the source or target is born:




```
\begin{sseqdata}[name=structpage example,title=\page]
\class(0,2)
\class(0,4)
\class(1,0)
\class(1,1)
\d2(1,0)(0,2)
\d3(1,1)(0,4)
\replaceclass(1,0)
\replaceclass(1,1)
\structline(1,0)(1,1)
\end{sseqdata}
\printpage[name=structpage example,page=3]
\qquad
\printpage[name=structpage example,page=4]
```

By specifying a page number, you can adjust which page the `\structline` starts on:



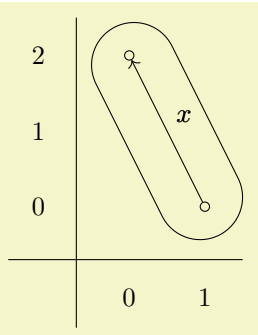

```
\begin{sseqdata}[name=structpage example2,title=\page]
\class(0,2)
\class(0,4)
\class(1,0)
\class(1,1)
\d2(1,0)(0,2)
\d3(1,1)(0,4)
\replaceclass(1,0)
\replaceclass(1,1)
\structline[page=2](1,0)(1,1)
\end{sseqdata}
\printpage[name=structpage example2,page=1]
\qquad
\printpage[name=structpage example2,page=2]
\qquad
\printpage[name=structpage example2,page=3]
\qquad
\printpage[name=structpage example2,page=4]
```

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*⟩ (no default)

The `\circleclasses` command uses the Ti*k*Z fittings 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.




```
\begin{sseqpage}[Adams grading,axes gap=0.7cm]
\class(0,2)
\class(1,0)
% Fit in the label x and also a symmetric invisible label to maintain symmetry
\d["x"{name=x}"x"'{name=x',opacity=0}]2(1,0)
\circleclasses[fit=(x)(x'),rounded rectangle](1,0)(0,2)
\end{sseqpage}
```

`rounded rectangle` (no value)

You can put a shape as an option and it will change the shape of the node drawn by `\circleclasses`. Any shape will do, but I think that an `ellipse` or `rounded rectangle` are the only particularly appealing options.

`ellipse ratio`=⟨*ratio*⟩ (no default, initially `1.2`)

By default, the shape drawn by `\circleclasses` is a "newelipse" which is a custom defined shape that respects the option `elipse 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` (no value)
`permanent cycle style` (no value)
`transient cycle style` (no value)
`this page class style` (no value)
`differential style` (no value)
`struct line style` (no value)

See the corresponding entry in the Ti*k*Z primitives section.

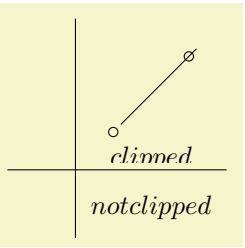
4.5 Options for Ti*k*Z primitives

`background` (no value)

This key instructs sseqpages to put the current Ti*k*Z primitive in the background. The way that the spectral sequence is printed is as follows:

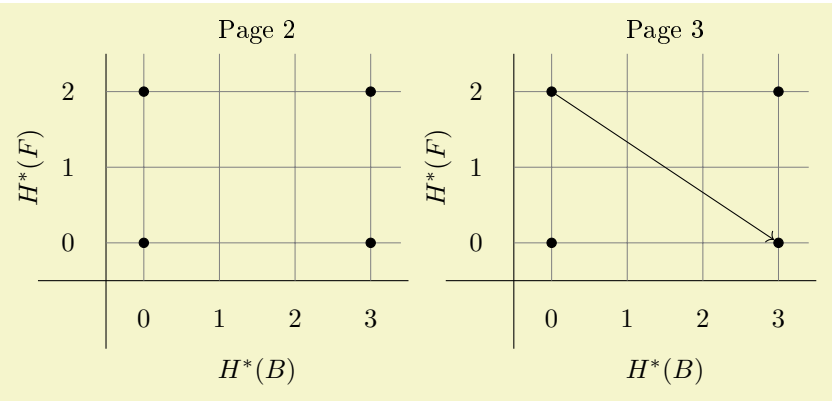
- The title, axes, axes ticks, and axes labels are printed (the appropriate steps are skipped when the `no title`, `no axes`, `no ticks`, or `no labels` keys are used or if no title or axes labels are provided).
- The Ti*k*Z background paths are printed.
- The clipping is inserted (unless the `no clip` key is used).
- All foreground elements (classes, differentials, structlines, and normal Ti*k*Z paths) are printed.

In particular, this means that foreground Ti*k*Z paths can be clipped by the standard clipping, but background paths that are outside of the clipping expand the size of the Tikz picture.




```
\begin{sseqpage}[no ticks]
\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,-0.35) {clipped};
\end{sseqpage}
```

Here is an example where Ti*k*Z labels with the `background` key are used to add labels and a grid. Note that the following example can be made more easily using the `title`, `x label`, `y label`, and `grid` options.




```
\begin{sseqdata}[name=tikz background example, cohomological Serre grading, math nodes,
                 classes=fill]
\begin{scope}[background]
\node at (\xmax/2,\ymax+0.8) {\textup{Page \page{}}};
\node at (\xmax/2,-1.7) {H^*(B)};
\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]
```

`page constraint`=⟨*predicate*⟩ (no default)
`page constraint or`=⟨*predicate*⟩ (no default)

This places a constraint on the pages in which the Ti*k*Z primitive is printed. This predicate should look something like `(\page<=4)&&(\page>=3)`. The predicate is anded together with any previous predicates, so that you can use this as an option for a `scope` and again for the individual Ti*k*Z primitive.

`\isalive`(⟨*coordinate*⟩)
`\isalive`{(⟨*coordinate 1*⟩)⋯(⟨*coordinate n*⟩)}

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

`page contraint={\isalive`(⟨*x*⟩,⟨*y*⟩,⟨*n*⟩)`}`

will print the Ti*k*Z primitive only on pages where the specified class is alive. Saying

`page contraint={\isalive`(⟨*coordinate 1*⟩) ⋯ (⟨*coordinate n*⟩)`}`

is equivalent to

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

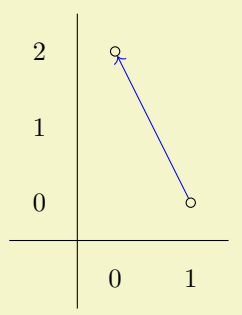
Writing

```
\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).

`class style` (no value)
`permanent cycle style` (no value)
`transient cycle style` (no value)
`this page class style` (no value)
`differential style` (no value)
`struct line style` (no value)

These classes apply the styling of the corresponding element to your tikz commands.




```
\begin{sseqpage}[differentials=blue]
\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 another example.

5 Miscellaneous commands

`\sseqset{`⟨*keys*⟩`}`

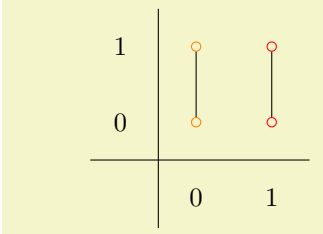
`\sseqset` is for adjusting the global options for all spectral sequences in the current scope. For instance, if most of the spectral sequences in the current document are going to be Adams graded, you can say `\sseqset{Adams grading}` and all future spectal sequences in the current scope will have Adams grading (unless you specify a different grading explicitly). As another example, `\sseqset{no axes}` will suppress axes from spectral sequences in the current scope.

`\foreach`

This command is from Ti*k*Z and works in pretty much the same way in sseqpages . The `\foreach` command is very flexible and has a lot of variants. See the Ti*k*Z manual for more details.

`\sseqnewcmd*`⟨*macro*⟩`(`⟨*x variable macro*⟩`,`⟨*y variable macro*⟩`)[`⟨*num args*⟩`]{`⟨*body*⟩`}`

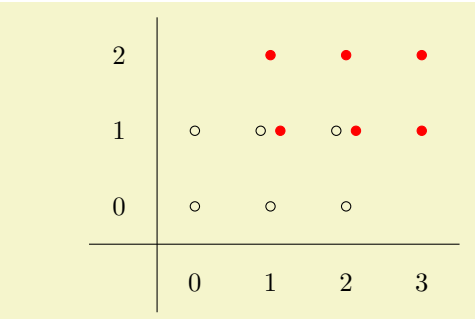
This command makes a new command with syntax similar to `\class`. By default it takes three arguments: `\mycommand[`⟨*options*⟩`](`⟨*x*⟩`,`⟨*y*⟩`)`. To access the ⟨*options*⟩ argument in the body of the command, use the macro `\options`. Likewise to access ⟨*x*⟩, use `\x`, and to access ⟨*y*⟩ use `\y`:




```
\sseqnewcmd\featuregroup{
    \class[\options](\x,\y)
    \class[\options](\x,\y+1)
    \structline(\x,\y)(\x,\y+1)
}
\begin{sseqpage}
\featuregroup[orange](0,0)
\featuregroup[red](1,0)
\end{sseqpage}
```

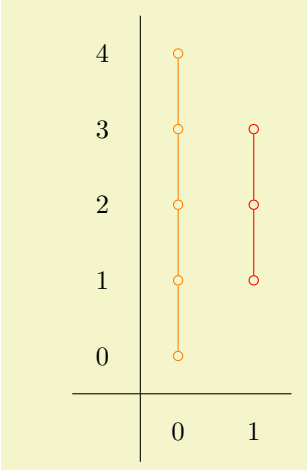
The unstarred version will throw an error if the command to be defined already exists. The starred variant will quietly overwrite the existing command. Note that the command is always defined globally.

Sometimes it's inconvenient to use `\x` and `\y` as arguments, for instance if you want to use code that already uses them as iterators. In this case, you can specify other macros to use for the arguments:




```
\sseqnewcmd\test(\a,\b){
    \foreach \x in {0,...,2} \foreach \y in {0,...,1}{
        \class[\options](\a+\x,\b+\y)
    }
}
\begin{sseqpage}
\test(0,0)
\test[red,fill](1,1)
\end{sseqpage}
```

The command you create with `\sseqnewcmd` can take up to six arguments in addition to `\options`, `\x`, and `\y`. These extra arguments are mandatory and are delimited by braces.



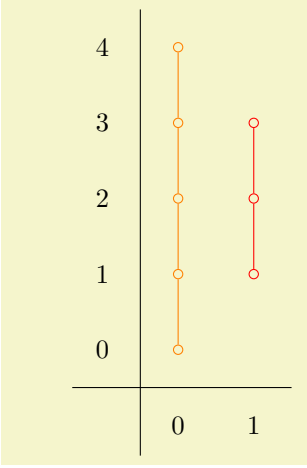

```
\sseqnewcmd*\tower[1]{
    \begin{scope}[\options]
    \class(\x,\y)
    \foreach \n in {1,...,#1}{
        \class(\x,\y+\n)
        \structline(\x,\y+\n-1)(\x,\y+\n)
    }
    \end{scope}
}
\begin{sseqpage}
\tower[orange](0,0){4}
\tower[red](1,1){2}
\end{sseqpage}
```

`\sseqnewgroup*[`⟨*num args*⟩`]{`⟨*body*⟩`}`

This is more or less a shorthand for

```
\sseqnewcmd*(\obscurexname,\obscureyname)[⟨num args⟩]{
    \begin{scope}[xshift=|obscurexname,yshift=|obscureyname,\options]
    ⟨body⟩
    \end{scope}
}
```

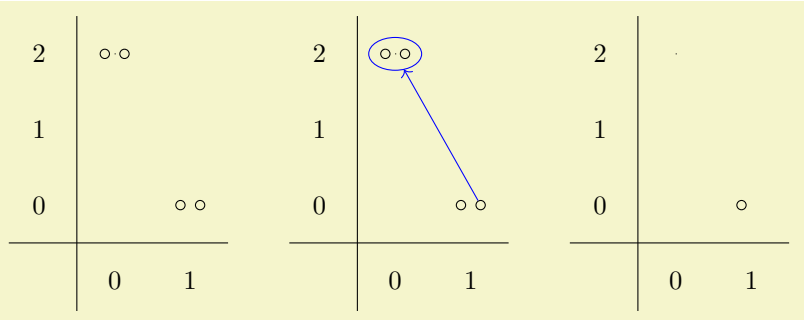
So that calling `\mygroup(x,y)` prints the whole group shifted to start at `(x,y)` instead of `(0,0)`. For instance:




```
\sseqnewgroup*\tower[1]{
    \class(0,0)
    \foreach \n in {1,...,#1}{
        \class(0,\n)
        \structline(0,\n-1)(0,\n)
    }
    \end{scope}
}
\begin{sseqpage}
\tower[orange](0,0){4}
\tower[red](1,1){2}
\end{sseqpage}
```

\getdtarget{⟨*macro*⟩}{⟨*page*⟩}⟨*source coordinate*⟩

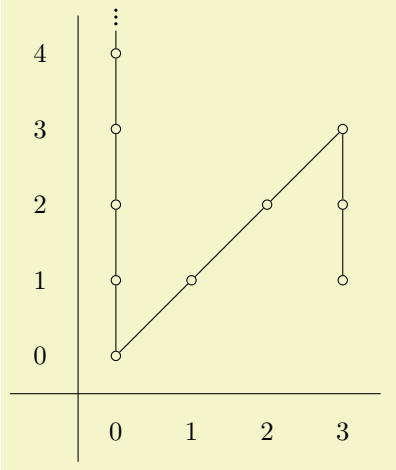
Sets ⟨*macro*⟩ equal to the coordinates of the target position of a length ⟨*page*⟩ differential starting at ⟨*source coordinate*⟩. This helps to make commands that draw fancy differentials. For instance, consider the following example, suggested by Catherine Ray:




```
% 0{}u(u) is the arg-spec for \d, 0{}u(u)mm looks like \d with two extra mandatory arguments
\NewDocumentCommand\twods{0{}u(u)mm}{
    \getdtarget\target#2{#3}              % Store the target position in \target
    \circleclasses[differential style,#1,name path=circ,page=#2--#2]
          (\target,#4)(\target,#5)        % Circle the classes, use differential style
    \d[invisible]#2(#3)(\target,#4)       % don't draw anything, but record source and targets as hit.
    \d[invisible]#2(#3)(\target,#5)
    \draw(\target,#4) -- (\target,#5)
               coordinate[midway](midpt);% put a coordinate in the center of the two classes
    \path[name path=lin] (#3) -- (midpt);% save path from start to midpoint
    % draw line in "differential style" from start to intersection point of circ and lin
    \draw[differential style,#1,name intersections={of=circ and lin},page constraint={|page==#2}]
                (#3) -- (intersection-1);
}
\begin{sseqdata}[name=cathex,Adams grading,differentials={blue}]
\class(0,2)
\class(0,2)
\class(1,0)\class(1,0)

\twods2(1,0,-1){1}{2}
\end{sseqdata}
\printpage[name=cathex,page=1]
\qquad
\printpage[name=cathex,page=2]
\qquad
\printpage[name=cathex,page=3]
```

5.1 The Class Stack

The class stack is a list of recently used classes that sseqpages maintains. I've only recently implemented this feature, so it is more liable to change in the future than other things. You activate the stack by saying `stack depth=`⟨*positive integer*⟩, which also indicates how many classes to store. The time cost scales linearly with the size, so it might be wise to pick a small number. Whenever you use the `\class` function, the class you added is pushed onto the stack. Here's an example that demonstrates basic usage:




```
\def\tower#1{
    \savestack
    \foreach \n in {1,...,#1}{
        \class(\lastx,\lasty+1)
        \structline(lastclass1)(lastclass)
    }
    \restorestack
}
\def\eta{
    \class(\lastx+1,\lasty+1)
    \structline(lastclass1)(lastclass)
}
\def\divtwo{
    \class(\lastx,\lasty-1)
    \structline(lastclass1)(lastclass)
}
\begin{sseqpage}[stack depth=2,y range={0}{4}]
\class(0,0)
\tower{5}\eta\eta\eta\divtwo\divtwo
\end{sseqpage}
```

The following commands are used to access the stack:

`lastclass`⟨n⟩

This is an automatically defined class name which refers to the nth most recent class if $n <$ ⟨*stack depth*⟩. If $n = 0$ you can leave it off.

`\lastx`⟨n⟩
`\lasty`⟨n⟩

This refers to the x and y position, respectively, of the nth class on the stack. If $n = 0$ you can leave it off.

`\pushstack`(⟨*coordinate*⟩)

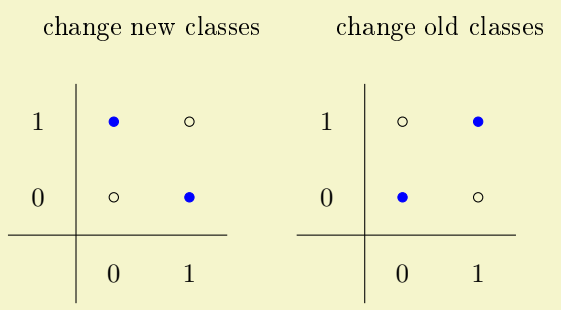
This adds the coordinate to the top of the stack. The coordinate is specified using the same syntax as a coordinate for `\structline` or `\replaceclass`.

`\savestack`
`\restorestack`

This saves and reverts the stack. Saves nest. Most frequently, you will want to use these at the start and end of a command that shouldn't modify the stack.

6 Styles

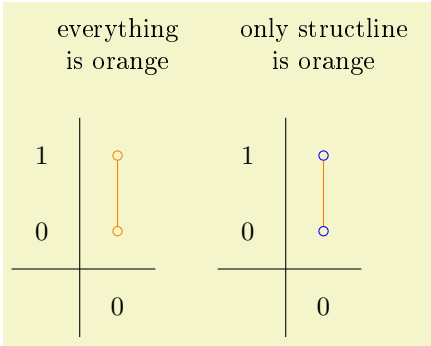
The sseqpages package has a large number of styles which control the appearance of specific components (e.g., classes, differentials, or structlines) of a spectral sequence. Each style has two corresponding keys: `classes` and `change classes`. Saying`classes=`*`\marg{keys}`* adds the keys to the list of options used to style every future class, whereas `change classes=`*`\marg{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=stylesex]
\class(0,0)\class(1,1)
\end{sseqdata}
\begin{sseqpage}[name=stylesex,classes={fill,blue},
    title={change new classes}
]
\class(0,1)\class(1,0)
\end{sseqpage}
\quad
\begin{sseqpage}[name=stylesex,
    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`:

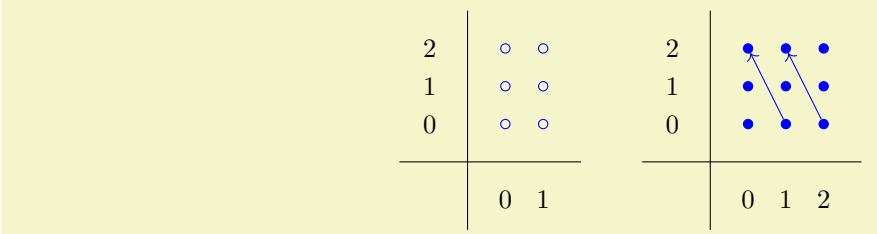



```
\begin{sseqpage}[classes=blue,
    title style={align=center,text width=2.4cm},
    title={everything is orange}]
\begin{scope}[orange]
\class(0,0)\class(0,1)
\structline(0,0)(0,1)
\end{scope}
\end{sseqpage}

\begin{sseqpage}[classes=blue,
    title style={align=center,text width=2.4cm},
    title={only structline 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.

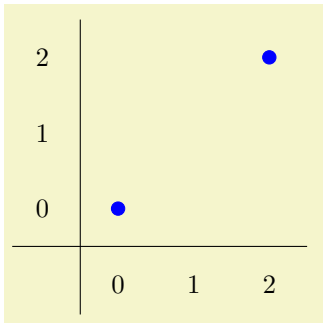
`sseqs={`⟨*keys*⟩`}` (no default)
`change sseqs={`⟨*keys*⟩`}` (no default)
`sseq=`⟨*keys*⟩ (no default)
`sseqs=`⟨*keys*⟩ (no default)

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{sseq={blue,scale=0.5}}
\begin{sseqpage}
\foreach \x in {0,1}
\foreach \y in {0,1,2}{
    \class(\x,\y)
}
\end{sseqpage}
\qquad
\begin{sseqpage}[
    Adams grading,
    classes=fill
]
\foreach \x in {0,1,2}
\foreach \y in {0,1,2}{
    \class(\x,\y)
}
\d2(1,0)
\d2(2,0)
\end{sseqpage}
```

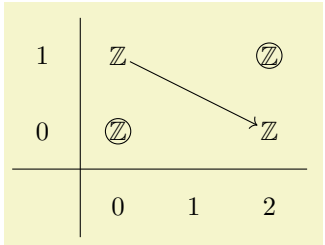
`classes={`⟨*keys*⟩`}` (no default)
`cycles={`⟨*keys*⟩`}` (no default)




```
\begin{sseqpage}[classes={blue,fill,minimum width=0.5em}]
\class(0,0)
\class(2,2)
\end{sseqpage}
```

`permanent classes={`⟨*keys*⟩`}` (no default)
`permanent cycles={`⟨*keys*⟩`}` (no default)
`change permanent classes={`⟨*keys*⟩`}` (no default)
`change permanent cycles={`⟨*keys*⟩`}` (no default)

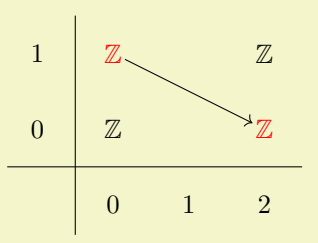
These options change the appearance of all permanent cycles (e.g., those classes which never support or are hit by a differential). For instance, we can circle the permanent cycles automatically. Note that because `permanent cycles` is more specific than `classes`, the `permanent cycles={draw}` command wins out over the `class={draw=none}` command to insure that the permanent cycle nodes are drawn.




```
\begin{sseqpage}[cohomological Serre grading, math nodes,
                 classes={draw=none},permanent cycles={draw}]
\foreach \x in {0,2} \foreach \y in {0,1}{
    \class["\mathbb{Z}"](\x,\y)
}
\d2(0,1)
\end{sseqpage}
```

`transient classes={`⟨*keys*⟩`}` (no default)
`transient cycles={`⟨*keys*⟩`}` (no default)
`change transient classes={`⟨*keys*⟩`}` (no default)
`change transient cycles={`⟨*keys*⟩`}` (no default)

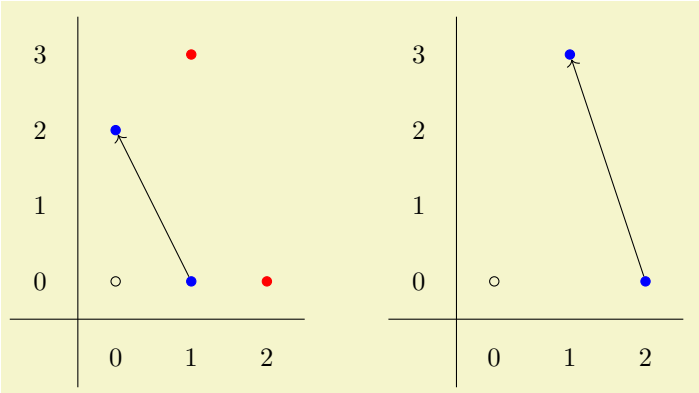
These options change the appearance of all transient cycles (e.g., those classes which eventually support or are hit by a differential). Again, this takes precedence over the `classes` option.




```
\begin{sseqpage}[cohomological Serre grading, math nodes,
                 classes={draw=none}, transient cycles=red]
\foreach \x in {0,2} \foreach \y in {0,1}{
    \class["\mathbb{Z}"](\x,\y)
}
\d2(0,1)
\end{sseqpage}
```

`this page classes`={⟨*keys*⟩} (no default)
`this page cycles`={⟨*keys*⟩} (no default)
`change this page classes`={⟨*keys*⟩} (no default)
`change this page cycles`={⟨*keys*⟩} (no default)

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 `this page classes` takes precedence over `transient cycles`




```
\begin{sseqdata}[name=this page cycles example,Adams grading,
                 transient cycles={red,fill},this page cycles={blue}]
\class(0,0)
\class(0,2)\class(1,0)
\class(1,3)\class(2,0)
\d2(1,0)\d3(2,0)
\end{sseqdata}
\printpage[name=this page cycles example,page=2]
\hskip1cm
\printpage[name=this page cycles example,page=3]
```

`edges`={⟨*keys*⟩} (no default)
`differentials`={⟨*keys*⟩} (no default)
`struct lines`={⟨*keys*⟩} (no default)
`change edges`={⟨*keys*⟩} (no default)
`change differentials`={⟨*keys*⟩} (no default)
`change struct lines`={⟨*keys*⟩} (no default)

The `edges` key applies to both differentials and structure lines. The `differentials` and `struct lines` keys both take precedence over `edges`.

`this page struct lines`={⟨*keys*⟩} (no default)
`change this page struct lines`={⟨*keys*⟩} (no default)

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

`tikz primitives`={⟨*keys*⟩} (no default)

`change tikz primitives`={⟨*keys*⟩} (no default)

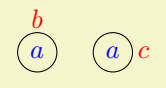
Applies to all tikz primitives.

`labels`={⟨*keys*⟩} (no default)
`change labels`={⟨*keys*⟩} (no default)

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

`class labels`={⟨*keys*⟩} (no default)
`inner class labels`={⟨*keys*⟩} (no default)
`outer class labels`={⟨*keys*⟩} (no default)
`change class labels`={⟨*keys*⟩} (no default)
`change inner class labels`={⟨*keys*⟩} (no default)
`change outer class labels`={⟨*keys*⟩} (no default)

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



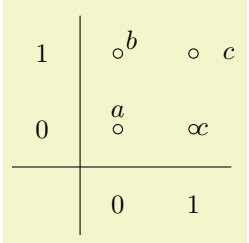

```
\begin{sseqpage}[no axes, classes={inner sep=0.1cm},
    outer class labels={red},
    inner class labels={blue}]
\class["a", "b" above](0,0)
\class["a", "c" right](1,0)
\end{sseqpage}
```

`edge labels`={⟨*keys*⟩} (no default)
`differential labels`={⟨*keys*⟩} (no default)
`struct line labels`={⟨*keys*⟩} (no default)
`change edge labels`={⟨*keys*⟩} (no default)
`change differential labels`={⟨*keys*⟩} (no default)
`change struct line labels`={⟨*keys*⟩} (no default)

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

`label distance`=⟨*dimension*⟩ (no default)

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`.

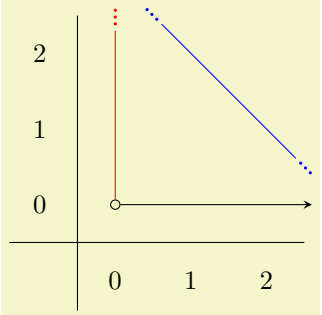



```
\begin{sseqpage}[label distance=0.3em, right label distance=0em]
\class["a" above](0,0)
\class["b" above right](0,1)
\class["c" right](1,0)
\class["c" {right=1em}](1,1)
\end{sseqpage}
```

`run off`=⟨*start tip*⟩-⟨*end tip*⟩ (no default)
`run off struct lines`=⟨*start tip*⟩-⟨*end tip*⟩ (no default, initially `...-...`)
`run off differentials`=⟨*start tip*⟩-⟨*end tip*⟩ (no default, initially `...-...`)

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

If an edge runs off the edge of the clipping, sseqpages 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.




```
\begin{sseqpage}[x range={0}{2},y range={0}{2},
                 draw orphan edges,run off=>-stealth]
\class(0,0)
\class(3,0)\class(0,3)
\structline(0,0)(3,0)
\structline[red](0,0)(0,3)
\structline[blue](3,0)(0,3)
\end{sseqpage}
```

7 Global options

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

`name`=⟨*sseq name*⟩ (no default)

This option must be used with the `sseqdata` environment where it indicates the name of the spectral sequence, which will be used with the `sseqpage` environment or `\printpage` command to draw the spectral sequence. The name used in a `sseqdata` environment must be new unless the environment is used with the `update existing` key in which case the `sseqdata` environment will add to the existing spectral sequence. It is optional when used with `sseqpage`, and if included the name given must be the name of an existing spectral sequence.

`page`=⟨*page number*⟩ (no default, initially `0`)

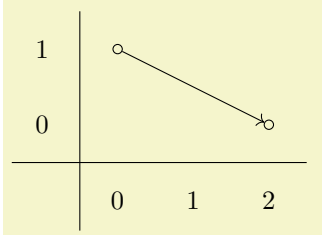
This key is for `sseqpage` and `\printpage`. It specifies which page of the spectral sequence is to be printed. On page r, all `\class`es that are not hit by differentials on pages less than r will be printed, as well as all `\structline`s whose source and target classes are both printed on page r, and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

`degree`={⟨*x degree*⟩}{⟨*y degree*⟩} (no default)
`cohomological Serre grading` (no value)
`homological Serre grading` (no value)
`Adams grading` (no value)

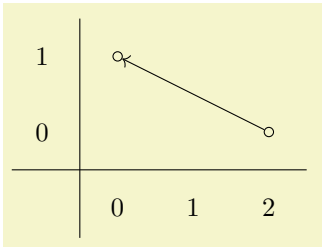
Specifies the degree of differentials. The ⟨*x degree*⟩ and ⟨*y degree*⟩ should both be mathematical expressions in one variable `#1` that evaluate to integers on any input. They specify the x and y displacement of a page `#1` differential. In practice, they will usually be linear expressions with `#1` coefficient 1, -1, or 0.

The `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 `\sseqset{degree={`⟨*x degree*⟩`}{`⟨*y degree*⟩`}}` or `\sseqset{Adams grading}` outside of the `sseqdata` and sseqpages environments.

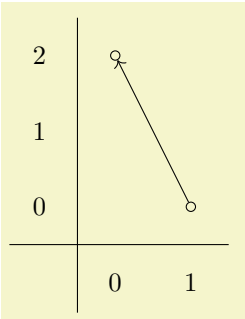
You can make a named grading convention by saying `\sseqset{my grading/.sseq grading={`⟨*x degree*⟩`}{`⟨*y degree*⟩`}}`. Then later passing `my grading` to a spectral sequence is equivalent to saying `degree={`⟨*x degree*⟩`}{`⟨*y degree*⟩`}`. The following grading conventions exist by default:




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




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

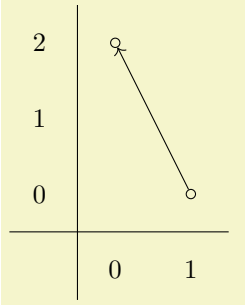



```
\begin{sseqpage}[Adams grading]% equivalent to degree={-1}{#1-1}
\class(0,2)
\class(1,0)
\d2(1,0)
\end{sseqpage}
```

`strict degree` (no value)
`lax degree` (no value)

If the degree is strict, then latex 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{sseqdata}[name=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}
```



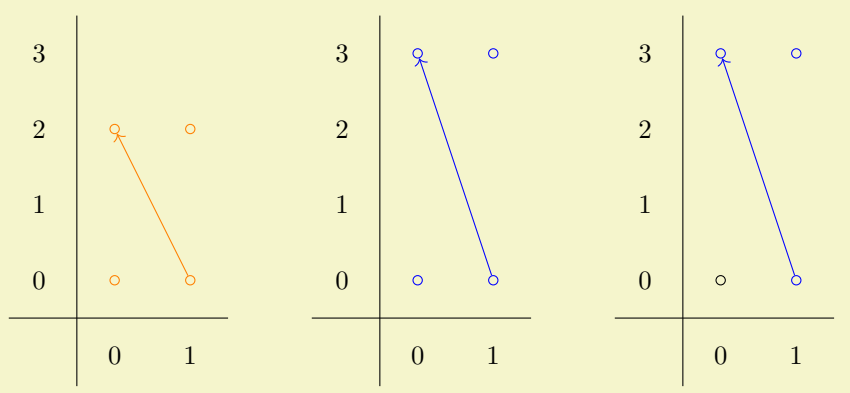

```
\begin{sseqdata}[name=laxdegree,Adams grading, lax degree]
\class(0,2)
\class(1,0)
\d3(1,0)(0,2) % No error because degree checking is off
\end{sseqdata}
\printpage[name=laxdegree,page=3]
```

`update existing` (no value)

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

`keep changes=`⟨*boolean*⟩ (default `true`) (initially `false`)

This option is only for the `sseqpage` environment, and only works when a `name` is provided. This option specifies that all of the commands in the current `sseqpage` environment should be carried forward to future pages of the same named spectral sequence. For example:




```
\begin{sseqdata}[name=keep changes example,Adams grading,y range={0}{3}]
\class(0,0)
\class(1,0)
\end{sseqdata}

\begin{sseqpage}[name=keep changes example,sseq=orange]
\class(0,2)
\class(1,2)
\classoptions[orange](1,0)
\d2(1,0)
\end{sseqpage}
%
\hskip1cm
%
\begin{sseqpage}[name=keep changes example,sseq=blue,keep changes]
\class(0,3)
\class(1,3)
\classoptions[blue](1,0)
\d3(1,0)
\end{sseqpage}
%
\hskip1cm
%
\printpage[name=keep changes example,page=3]
```

Note that the orange classes and differential do not persist because the `keep changes` option is not set in the first `sseqpage` environment, but the blue classes and differential do, since the `keep changes` option is set in the second `sseqpage` environment.

`stack depth`=⟨*nonnegative integer*⟩ (no default, initially 0)

Indicate how deep the class stack should be. By default, this is set to 0, which turns the class stack off. That is because the class stack can be time expensive in certain circumstances, and so it is only worth the time if you are using it. Right now, the class stack is internally an array, so every time an element is pushed, all of the elements of the stack have to be copied. (I might in the future turn the class stack into a linked list, which would make the stack cheap to maintain and expensive to index).

`no differentials` (no value)
`draw differentials` (no value)

The option `no differentials` suppresses all of the differentials on the current page, whereas `draw differentials` causes the page appropriate differentials to be drawn. This is useful for explaining how the computation of a spectral sequence goes.

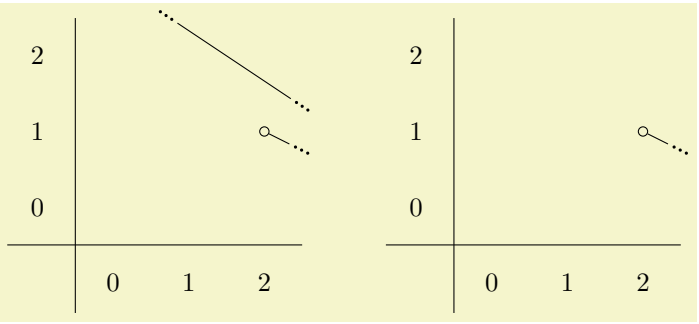
`no struct lines` (no value)
`draw struct lines` (no value)

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

`no orphan edges` (no value)

`draw orphan edges`=⟨*boolean*⟩ (default `true`)(initially `true`)

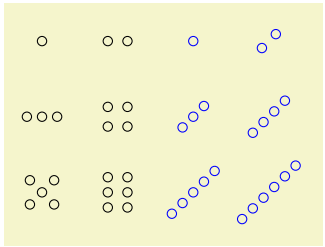
An edge is an "orphan" if both its source and target lie off the page. By default these are drawn, but with the option `no orphan edges` they are not. If the option `no orphan edges` has been set, `draw orphan edges` undoes it.




```
\begin{sseqdata}[
    name=orphan edges example,
    cohomological Serre grading,
    x range={0}{2}, y range={0}{2}]
\class(0,3)\class(3,1)
\d3(0,3)
\class(2,1)\class(4,0)
\d2(2,1)
\end{sseqdata}
\printpage[name=orphan edges example]
\hskip1cm
\printpage[name=orphan edges example,
    no orphan edges]
```

`class pattern`=⟨*class pattern name*⟩ (no default, initially `standard`)

This key specifies the arrangement of multiple classes at the same coordinate. The default value is `standard`.



```
\begin{sseqdata}[name=class pattern example,no axes,ymirror]
\class(0,0)
\class(1,0)\class(1,0)
\class(0,1)\class(0,1)\class(0,1)
\class(1,1)\class(1,1)\class(1,1)\class(1,1)
\class(0,2)\class(0,2)\class(0,2)\class(0,2)\class(0,2)
\class(1,2)\class(1,2)\class(1,2)\class(1,2)\class(1,2)\class(1,2)
\end{sseqdata}

\printpage[name=class pattern example, class pattern=standard]
\printpage[name=class pattern example, change classes=blue,
           class pattern=linear, class placement transform={rotate=45}]
```

You can add new class patterns using `\sseqnewclasspattern`:

`\sseqnewclasspattern`{⟨*class pattern name*⟩}{⟨*offsets*⟩}

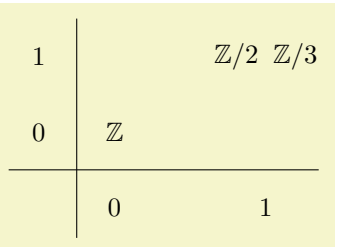
Creates a new class pattern. For example, the `linear` class pattern is created using the command:

```
\newsseqclasspattern{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);
}
```

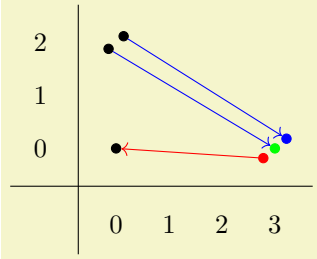
For instance the third row indicates that if there are three classes at the position `(x,y)` they should be printed at `(x-0.2,y)`, `(x,y)`, and `(x+0.2,y)`. You can give as many rows as you like; sseqpages will throw an error if there are more classes in any position than the maximum number that your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

`class placement transform`={⟨*transform keys*⟩} (no default)

The option `class placement transform` allows the user to specify a Tikz coordinate transform to adjust the relative position of multiple nodes in the same (x,y) position. 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.




```
\begin{sseqpage}[classes={draw=none},class placement transform={xscale=3},
                 math nodes, xscale=2, x axis extend end=0.7cm]
\class["\mathbb{Z}"](0,0)
\class["\mathbb{Z}/2"](1,1)
\class["\mathbb{Z}/3"](1,1)
\end{sseqpage}
```




```
\begin{sseqpage}[classes=fill,class placement transform={rotate=40},
                 cohomological Serre grading,differentials=blue,scale=0.7]
\class(0,0)
\class(0,2)\class(0,2)
\class[red](3,0)\class[green](3,0)\class[blue](3,0)

\d3(0,2,1,2)
\d3(0,2,-1,-1)
\draw[->,red](3,0,1)--(0,0);
\end{sseqpage}
```

`math nodes=`⟨*boolean*⟩ (default `true`)(initially `true`)

This key instructs sseqpages to put all labels in math mode automatically.

7.1 Global Coordinate Transformations

Of the normal Ti*k*Z coordinate transformations, only the following can be applied to a sseq diagram:

`scale=`⟨*factor*⟩ (no default)
`xscale=`⟨*factor*⟩ (no default)
`yscale=`⟨*factor*⟩ (no default)
`xmirror` (no value)
`ymirror` (no value)

Scale the diagram by ⟨*factor*⟩. Under normal circumstances, you can tell Ti*k*Z to mirror a diagram by saying, for instance, `xscale=-1`, but sseqpages needs to be aware that the diagram has been mirrored in order to draw the axes correctly. Thus, if you want to mirror a spectral sequence, use the `xmirror` and `ymirror` options as appropriate.

`rotate=`⟨*angle*⟩ (no default)

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

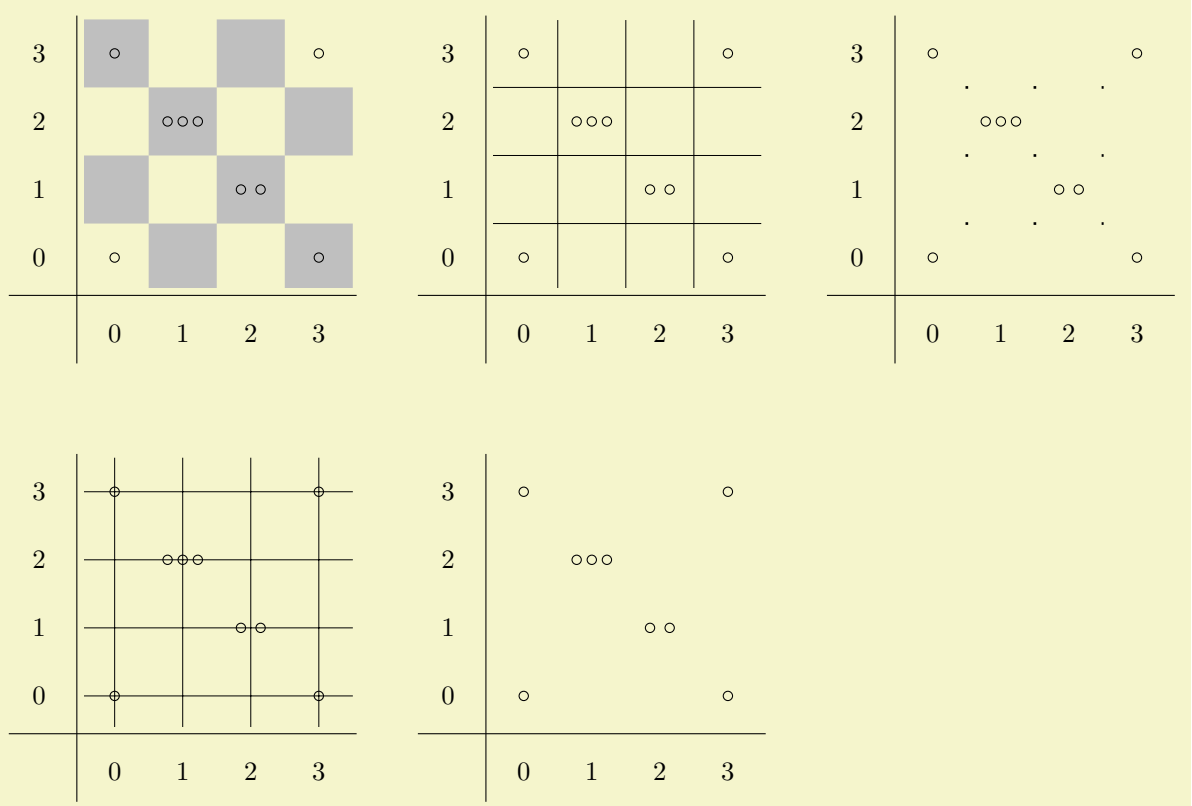
7.2 Plot Options and Axes Style

`x range={`⟨*x min*⟩`}{`⟨*x max*⟩`}` (no default)
`y range={`⟨*y min*⟩`}{`⟨*y max*⟩`}` (no default)

These options force the x and 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.

`grid={`⟨*grid style*⟩`}` (no default)

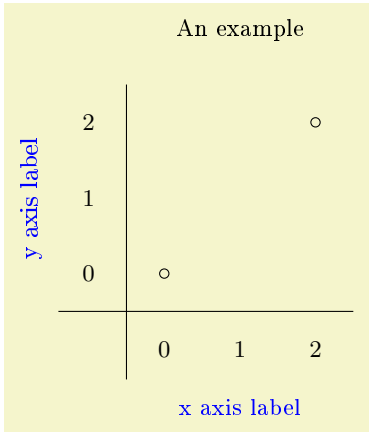
Makes sseqpages draw a grid. The grid styles and a significant part of the code that produces them were stolen from `sseq.sty`.



```
\begin{sseqdata}[name=grid example,scale=0.9]
\class(0,0)
\class(3,0)
\class(2,1)\class(2,1)
\class(1,2)\class(1,2)\class(1,2)
\class(0,3)
\class(3,3)
\end{sseqdata}
\vbox{
\hbox{
\printpage[name=grid example,grid=chess]
\qquad
\printpage[name=grid example,grid=crossword]
\qquad
\printpage[name=grid example,grid=dots]
}
\vskip30pt
\hbox{
\printpage[name=grid example,grid=go]
\qquad
\printpage[name=grid example,grid=none]
}
}
```

It is possible to make your own grid style by defining the command `\sseq@grid@yourgridname` to draw a grid.

`title`=⟨*text*⟩	(no default)
`title style`=⟨*options*⟩	(no default)
`x label`=⟨*text*⟩	(no default)
`y label`=⟨*text*⟩	(no default)
`x label style`=⟨*options*⟩	(no default)
`y label style`=⟨*options*⟩	(no default)
`label style`=⟨*options*⟩	(no default)




```
\begin{sseqpage}[title={An example},
    x label={x axis label}, y label={y axis label},
    label style={blue,font=\small},
    x label style={yshift=10pt},y label style={xshift=10pt}]
\class(0,0)
\class(2,2)
\end{sseqpage}
```

`no title` (no value)
`draw title` (no value)
`no x label` (no value)
`no y label` (no value)
`draw x label` (no value)
`draw y label` (no value)

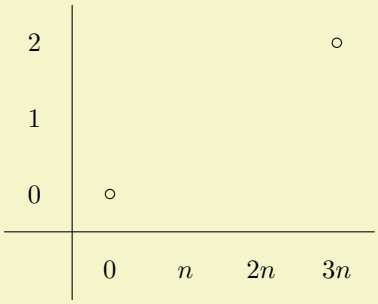
Suppress or unsuppress the title, x label, or y label respectively.

`no x axis` (no value)
`no y axis` (no value)
`no axes` (no value)
`draw x axis` (no value)
`draw y axis` (no value)
`draw axes` (no value)

Suppress the axis. Also suppresses axes ticks. If there is a title or axes labels they will still be drawn. You can draw your own axes using tikz inside a `scope` environment with the `background` key.

`no x ticks` (no value)
`no y ticks` (no value)
`no ticks` (no value)
`draw x ticks` (no value)
`draw y ticks` (no value)
`draw ticks` (no value)

Suppress axes ticks (the numbers next to the axes). Only matters if axes are drawn. You can make your own ticks using tikz 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 function`).




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

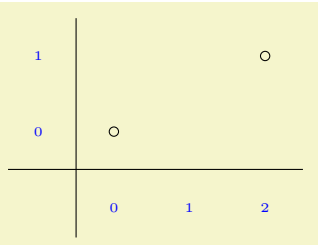
`x tick step`=⟨*positive integer*⟩ (no default, initially 1)
`y tick step`=⟨*positive integer*⟩ (no default, initially 1)

`tick step`=⟨*positive integer*⟩ (no default, initially `1`)

Sets the interval between labels.

`x tick style`={⟨*keys*⟩} (no default)
`y tick style`={⟨*keys*⟩} (no default)
`tick style`={⟨*keys*⟩} (no default)

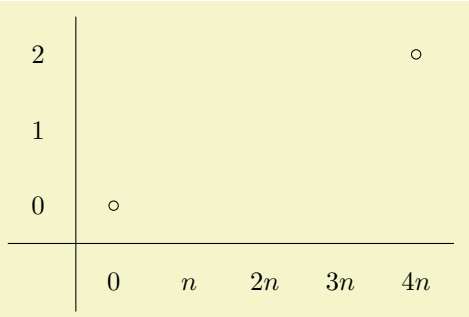
Change the tick style:




```
\begin{sseqpage}[tick style={blue,font=\tiny}]
\class(0,0)\class(2,1)
\end{sseqpage}
```

`x tick function`=⟨*function*⟩ (no default, initially `#1`)
`y tick function`=⟨*function*⟩ (no default, initially `#1`)
`tick function`=⟨*function*⟩ (no default, initially `#1`)

A function that takes in the current x value and outputs the appropriate tick.




```
\begin{sseqpage}[x range={0}{4},
    x tick function={
        \ifnum#1=0\relax
            0
        \else
            \ifnum#1=1\relax
                \vphantom{2}n
            \else
                #1n
            \fi
        \fi
    }
]
\class(0,0)
\class(4,2)
\end{sseqpage}
```

7.3 Layout

`x axis gap`=⟨*dimension*⟩ (no default, initially `0.5cm`)
`y axis gap`=⟨*dimension*⟩ (no default, initially `0.5cm`)
`axes gap`=⟨*dimension*⟩ (no default, initially `0.5cm`)

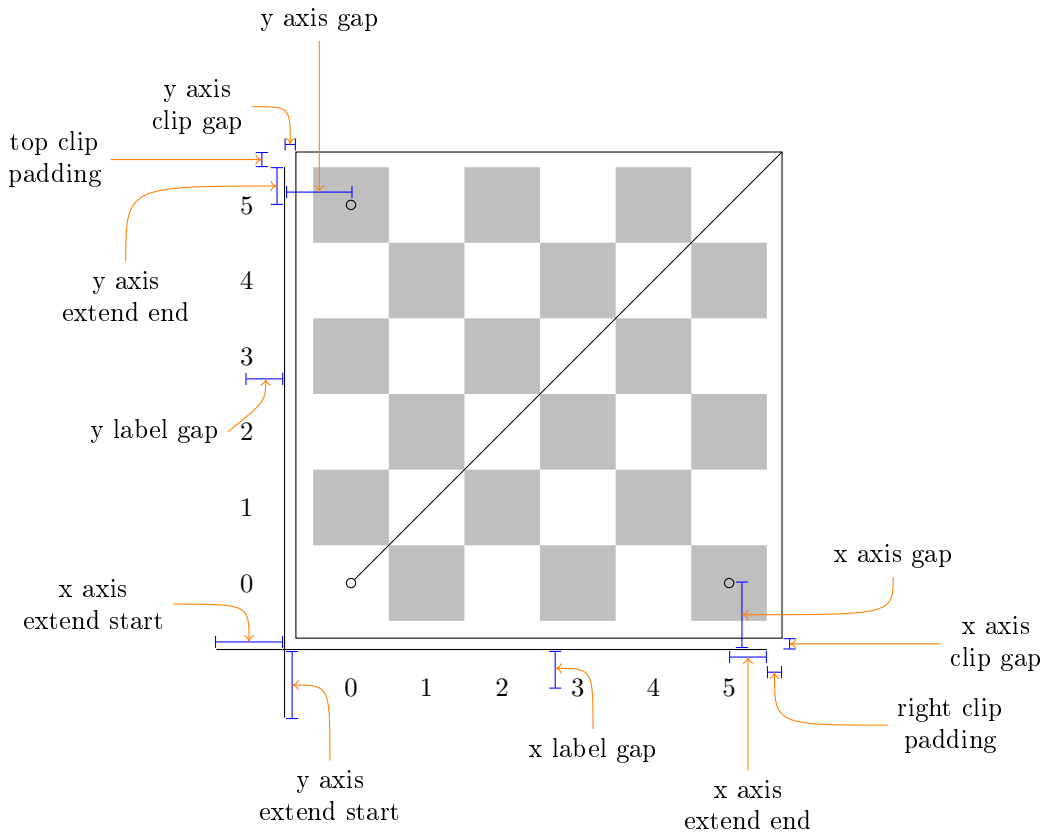
`x label gap`=⟨*dimension*⟩ (no default, initially `0.5cm`)
`y label gap`=⟨*dimension*⟩ (no default, initially `0.5cm`)

`x axis start extend`=⟨*dimension*⟩ (no default, initially `0.5cm`)
`y axis start extend`=⟨*dimension*⟩ (no default, initially `0.5cm`)
`x axis end extend`=⟨*dimension*⟩ (no default, initially `0.9cm`)
`y axis end extend`=⟨*dimension*⟩ (no default, initially `0.9cm`)

`x axis clip padding`=⟨*dimension*⟩ (no default, initially `0.1cm`)
`y axis clip padding`=⟨*dimension*⟩ (no default, initially `0.1cm`)

`right clip padding`=⟨*dimension*⟩ (no default, initially `0.1cm`)

`left clip padding`=⟨*dimension*⟩ (no default, initially `0.4cm`)
`top clip padding`=⟨*dimension*⟩ (no default, initially `0.1cm`)
`bottom clip padding`=⟨*dimension*⟩ (no default, initially `0.4cm`)



`custom clip`=⟨*clip path*⟩ (no default)

Give a custom clipping. The clipping specified must be in the form of a valid Ti*k*Z path, for instance `\clip (0,0) rectangle (10,10);`. See the Ti*k*Z manual for more details on clippings. This clipping is also applied to any grid and is used to draw ellipses on appropriate differentials or struct lines that go out of bounds and to determine whether a differential or struct 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.

Note that if you are not careful about how you use this, really weird things can happen.

`clip`=⟨*boolean*⟩ (default `true`)(`initially true`)

If this is false the spectral sequence 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.

`rotate labels`=⟨*boolean*⟩ (default `true`)(`initially false`)

If you use `rotate=`*90* but also want the labels rotated (so that the whole diagram is sideways) use this key.