

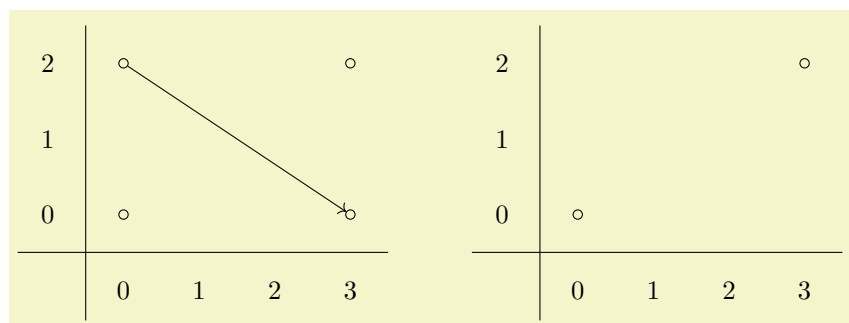
Contents

1	Introduction	1
2	The Environments	2
3	The main commands	2
3.1	TikZ Primitives	6
4	Options for the main commands	8
4.1	Universal options	8
4.2	Options for <code>\class</code>	10
4.3	Options for <code>\d</code> and <code>\structline</code>	12
5	Commands	13
6	Global options	15
6.1	Styles	19
6.2	Global Coordinate Transformations	22
6.3	Layout	22
6.4	Axes Style	23

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 command `\printpage` is also available as a synonym for a `sseqpage` environment 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}[\langle options \rangle]
\langle environment contents \rangle
\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}[\langle options \rangle]
\langle environment contents \rangle
\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[\langle options \rangle]

```

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

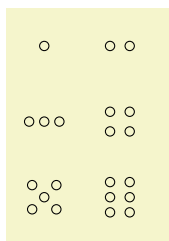
3 The main commands

```

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

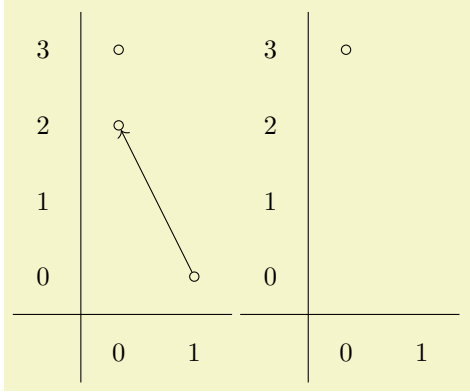
```

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.

	<pre> \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} </pre>
---	--

The effect of the `\class` command is to print a TikZ node on a range of pages. Any option that would work for a TikZ `\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:

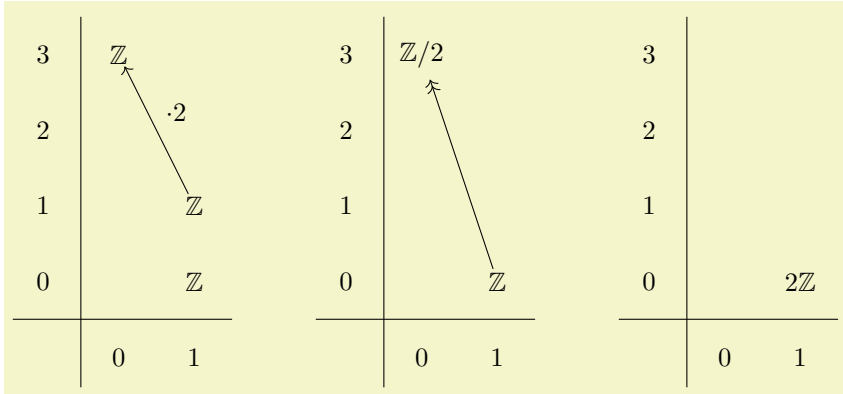


```
\begin{sseqdata}[name=class example,Adams grading]
\class(1,0)
\class(0,2)
\class(0,3)
\class(0,3)
\d2(1,0)
\end{sseqdata}
\printpage[name=class example,page=2]
\printpage[name=class example,page=3]
```

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

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

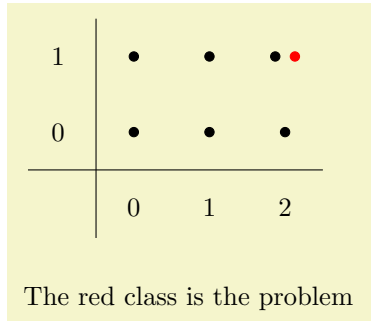
After a class is the source or target of a differential, it disappears on the next page. However, some differentials are not injective or not surjective. Using the command `\replaceclass` causes a new symbol to appear on the page after a class supported or accepted a differential (or both).



```
\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]
```

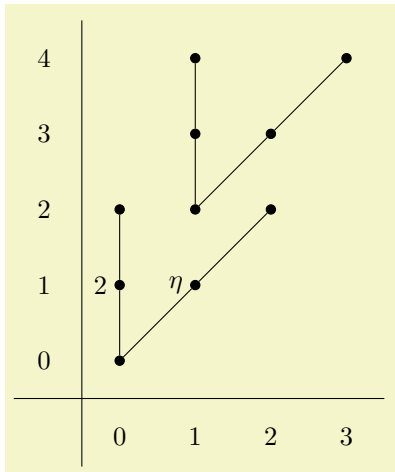
`\classoptions[⟨options⟩](⟨x⟩,⟨y⟩,⟨n⟩)`

This adds options to a class that already exists. If there are multiple classes at the coordinate (x,y) you must specify which using an integer or a tag *n*. 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:



```
\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`:

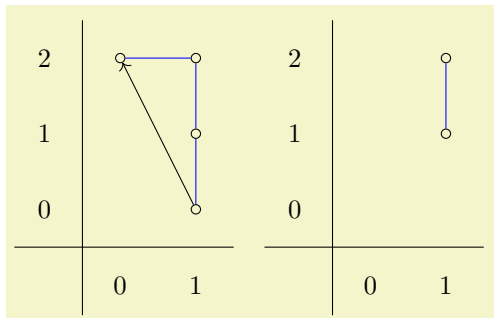


```
\sseqnewgroup\mygroup{
  \class(0,0)
  \class(0,1)
  \class(0,2)
  \class(1,1)
  \class(2,2)
  \structline(0,0)(0,1)
  \structline(0,1)(0,2)
  \structline(0,0)(1,1)
  \structline(1,1)(2,2)
}
\begin{sseqpage}[classes=fill,class labels={left=0.3em},math nodes]
\mygroup(0,0)
\mygroup(1,2)
\classoptions["2"](0,1)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

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

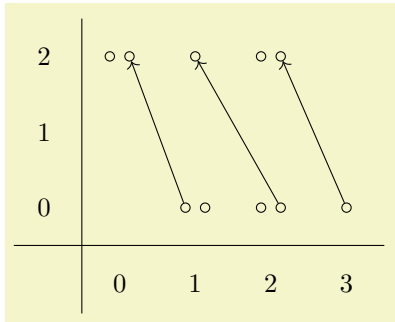
`\d[⟨options⟩](⟨page⟩⟨source coordinate⟩)`

This creates a differential starting at $\langle \text{source coordinate} \rangle$ of length determined by the specified page. In order to use the `\d` command, 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.



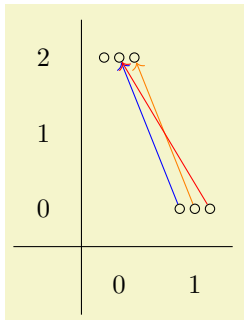
```
\begin{sseqdata}[name=d example,degree={-1}{\#1},
  struct lines=blue]
\class(0,2)
\class(1,2)
\class(1,1)
\class(1,0)
\structline(1,2)(0,2)
\structline(1,2)(1,1)
\structline(1,1)(1,0)
\d2(1,0)
\end{sseqdata}
\printpage[name=d example,page=2]
\hskip0.3cm
\printpage[name=d example,page=3]
```

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`.

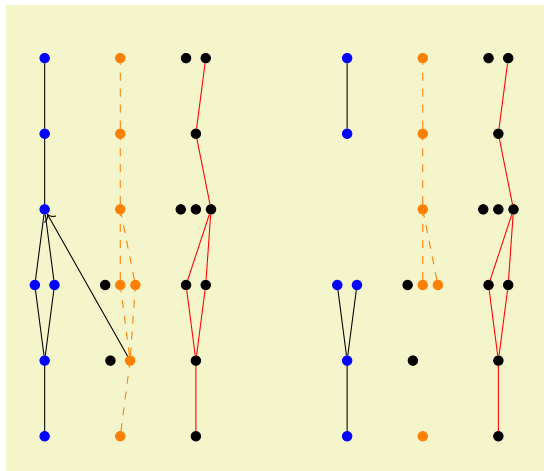
`\doptions[options](page)(source coordinate)`

This command adds options to an existing differential, just like `\classoptions` except for differentials.

`\structline[options](source coordinate)(target coordinate)`

This command creates a structure line from `(source coordinate)` to `(target coordinate)`. The source and target coordinates are of the form $(\langle x \rangle, \langle y \rangle, \langle n \rangle)$. If there are multiple classes at (x,y) , then $\langle n \rangle$ 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.



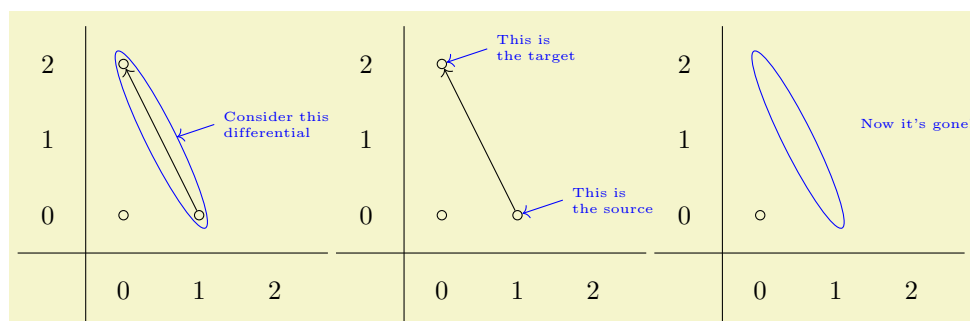
```
\sseqnewgroup*tower{
\class(0,0)
\class(0,2)
\foreach \y in{1,...,5}{
\class(0,\y)
\structline(0,\y-1,-1)(0,\y,-1)
}
\structline(0,1,-1)(0,2,-2)
\structline(0,2,-2)(0,3,-1)
}
\begin{sseqdata}[name=structline example,
classes={circle,fill},
Adams grading, no axes]
\class(1,1)\class(1,2)
\class(2,3)\class(2,3)\class(2,5)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

3.1 TikZ Primitives

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[alias=start](1,0)
\class[alias=end](0,2)
\d2(1,0)
\end{sseqdata}
%
\begin{sseqpage}[name=tikz example]
\begin{scope}[blue,font=\tiny]
\node[name path=myellipse, draw, ellipse, inner sep=4pt, scale=0.7, rotate fit=26.5,
fit=(start) (end)] {};
\path[name path=myline](1.2,1.2)--(0.6,1);
\draw[->,name intersections={of=myellipse and myline}] (1.2,1.2)--(intersection-1);
\node[right,text width=1.6cm] at (1.2,1.2) {Consider this differential};
\end{scope}
\end{sseqpage}
%
\begin{sseqpage}[name=tikz example]
\begin{scope}[<-,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{scope}
\end{sseqpage}
%
\begin{sseqpage}[page=3,name=tikz example]
\node[draw,ellipse,blue,inner sep=4pt,scale=0.7,rotate fit=26.5,fit=(start) (end)] {};
\node[right,blue,font=\tiny] at (1.2,1.2) {Now it's gone!};
\end{sseqpage}
```

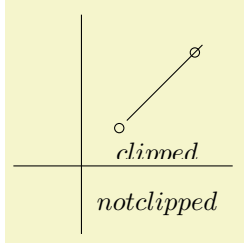
The following two keys are special to TikZ primitives:

background (no value)

This key instructs `sseqpages` to put the current TikZ primitive in the background. The way that the spectral sequence is printed is as follows:

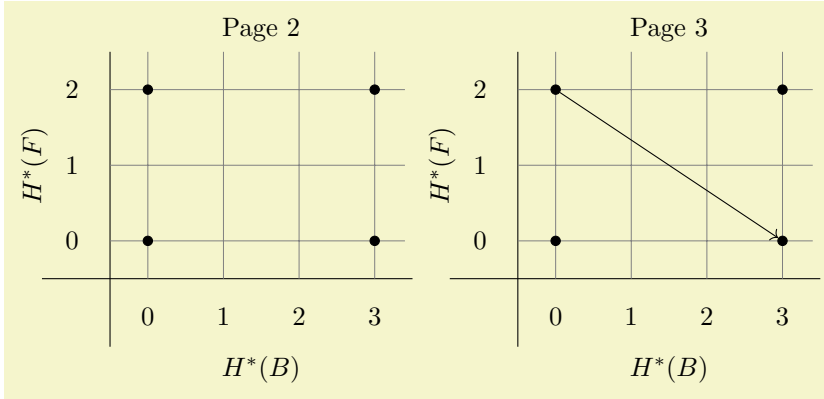
- The axes and axes labels are printed (except when the `no axes` or `no axes labels` keys are used).
- The TikZ background paths are printed.
- The clipping is inserted (unless the `no clip` key is used).
- All foreground elements (classes, differentials, structlines, and normal TikZ paths) are printed.

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:



```
\begin{sseqpage}[no axes labels]
\class(0,0)
\class(1,1)
\begin{scope}[on background layer]
\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 TikZ labels with the `background` key are used to add labels and a grid:



```
\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 TikZ 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 TikZ primitive.

```
\isalive(<coordinate>)
\isalive{(<coordinate 1>)\dots(<coordinate n>)}
```

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

```
page constraint={\isalive(<x>,<y>,<n>)}
```

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

```
page constraint={\isalive(<coordinate 1>) \dots (<coordinate n>)}
```

is equivalent to

```
page constraint={\isalive(<coordinate 1>) && \dots && \isalive(<coordinate n>)}
```

Writing

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

has the same result as `\structline(1,0)(2,2)`, except that you can't later use `\structlineoptions` on it.

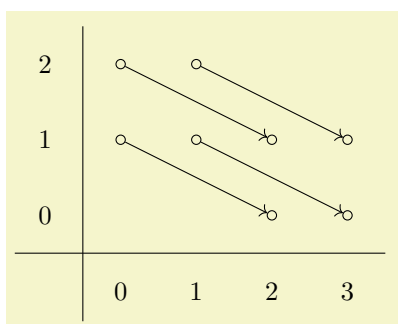
4 Options for the main commands

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 TikZ 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}
```

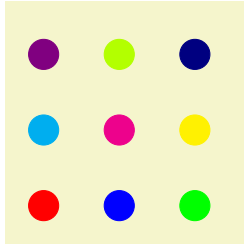
This code segment is very useful so `sseqpages` has the command `\sseqnewgroup` which to make code like this more convenient. The following code produces the same output as above:

```
\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 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 `sseqpages`, 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 L^AT_EX color package, so see the color package documentation for more information.

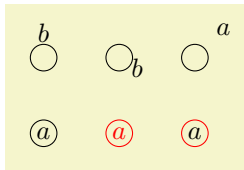


```
\begin{sseqpage}[classes={fill,inner sep=0.4em}, no axes]
\class[red](0,0)
\class[blue](1,0)
\class[green](2,0)
\class[cyan](0,1)
\class[magenta](1,1)
\class[yellow](2,1)
\class[blue!50!red](0,2) % a 50-50 blend of blue and red
\class[green!30!yellow](1,2) % 70% green, 30% yellow
\class[blue!50!black](2,2)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

" $\langle text \rangle$ " $\langle options \rangle$

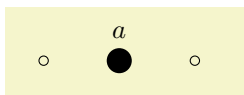
Specify a label for a class, a differential, or a structure line. This uses the TikZ quotes syntax. The options include anything you might pass as an option to a TikZ node, including arbitrary coordinate transforms, colors, opacity options, shapes, fill, draw, etc. The behavior is a little different depending on whether you use it on a class or on a differential or struct line.

For a class, the $\langle text \rangle$ is placed in the position **inside** the node by default – in effect, the $\langle text \rangle$ 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 $\langle options \rangle$ are those that alter the appearance of text, such as opacity and color.



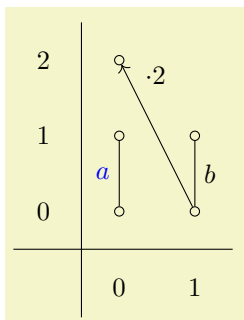
```
\begin{sseqpage}[no axes,classes={minimum width=width("a")+0.5em}]
\class["a"] (0,0)
\class["a",red] (1,0)
\class["a" black,red] (2,0)
\class["b" above] (0,1)
\class["b" {below right,yshift=0.1cm}] (1,1)
\class["a" {above right={1em}}] (2,1)
\end{sseqpage}
```

You can adjust the default behavior of class labels using the **labels**, **class labels**, **inner class labels** or **outer class labels** style options. 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}
```

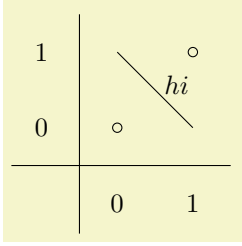
For either a `\structline` or a `\class` the label normally goes on the right side of the edge. The special `'` option makes it go in the opposite position from the default. I copied the code to handle this from the **tikzcd** package, so if you use **tikzcd**, this should be familiar.



```
\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}
```

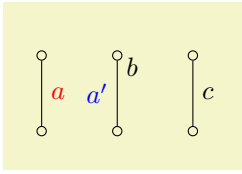
You can use the style options **labels**, **edge labels**, **differential labels**, and **struct line labels** to adjust the styling of edge labels. For instance, if you would prefer for the labels to default to the left

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 TikZ primitives:



```
\begin{sseqpage}
\class(0,0) \class(1,1)
\draw (1,0) to["hi"'] (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 color 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`.



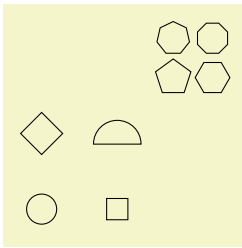
```
\begin{sseqpage}[no axes,background color=graphicbackground]
\foreach\x in {0,1,2} \foreach\y in {0,1}{
  \class(\x,\y)
}
\structline["a" red] (0,0) (0,1)
\structline["a'" blue,"b"'] (1,0) (1,1)
\structline["c" description] (2,0) (2,1)
\end{sseqpage}
```

4.2 Options for `\class`

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

A TikZ shape

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

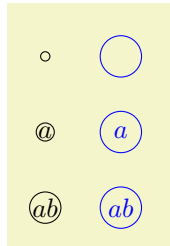


```
\begin{sseqpage}[no axes,classes={inner sep=0.4em},
class placement transform={scale=2}]
\class(0,0)
\class[rectangle] (1,0)
\class[diamond] (0,1)
\class[semicircle] (1,1)
\class[regular polygon, regular polygon sides=5] (2,2)
\class[regular polygon, regular polygon sides=6] (2,2)
\class[regular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

<code>minimum width=<dimension></code>	(no default)
<code>minimum height=<dimension></code>	(no default)
<code>minimum size=<dimension></code>	(no default)
<code>inner sep=<dimension></code>	(no default)
<code>outer sep=<dimension></code>	(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:

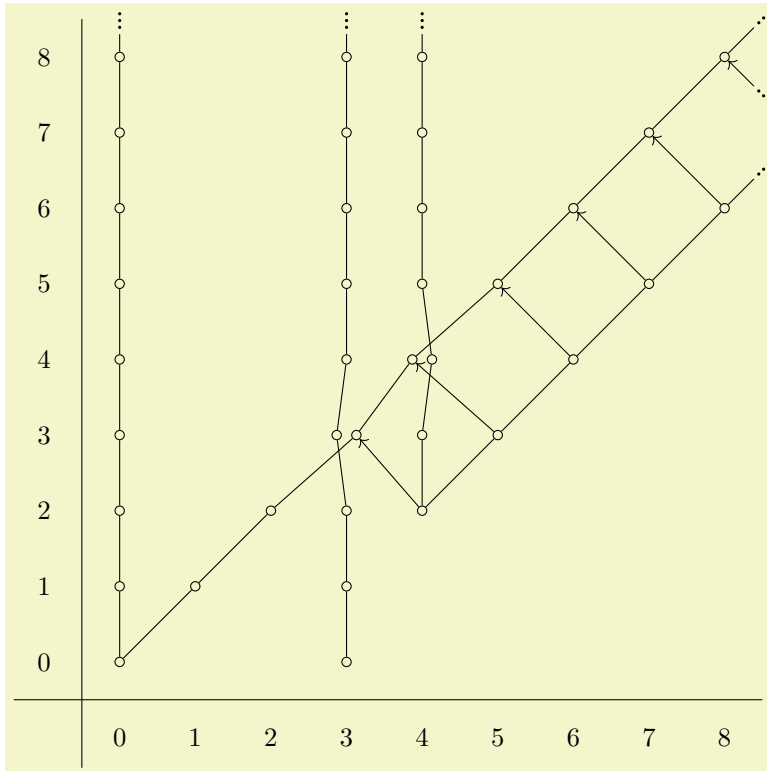


```
\begin{sseqdata}[no axes,name=minimum width example]
\class["ab"](0,0)
\class["a"](0,1)
\class(0,2)
\end{sseqdata}
\printpage[name=minimum width example]
\printpage[name=minimum width example,
change classes={blue,minimum width=width("ab")+0.5em}]
```

tag= $\langle tag \rangle$

(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:



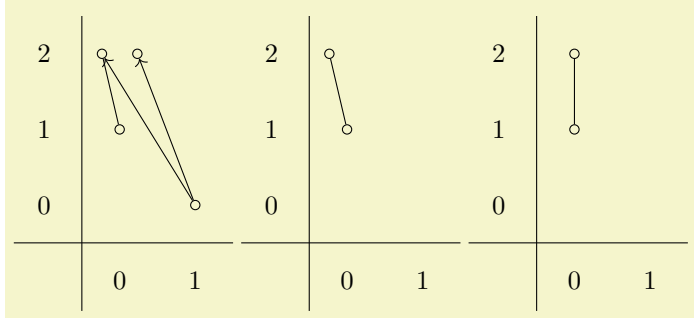
```
\sseqnewgroup*\tower{
\class(0,0)
\foreach\i in {1,...,11}{
\class(0,\i)
\structline(0,\i-1,-1)(0,\i,-1)
}
}
\sseqnewgroup\hvee{
\tower(0,0)
\foreach\i in {1,...,11}{
\class(\i,\i)
\structline(\i-1,\i-1,-1)(\i,\i,-1)
}
}
\begin{sseqpage}[degree=-1{1},x range={0}{8},y range={0}{8}]
\tower(3,0)
\hvee[tag=id](0,0)
\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 "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.

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

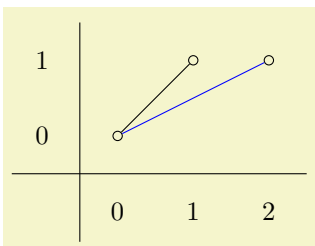
By default, a class uses the offset specified by `class pattern`. Occasionally this is undesirable. In this case, you can specify the offset for a particular class by hand. For example if the sum of two classes is hit by a differential, it looks better for the class replacing them to be centered:



```
\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}
```

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

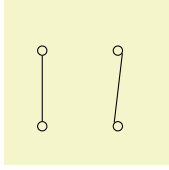
4.3 Options for `\d` and `\structline`

Because the main job of the `\d` and `\structline` commands is to print an edge on the appropriate pages of the spectral sequence, most TikZ options that you could apply to a TikZ “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= $\langle anchor \rangle$ (no default)

`target anchor=<anchor>` (no default)

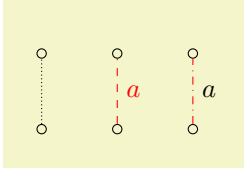
Because you can't use the normal TikZ mechanism for specifying the source and target anchors, `sseqpages` has these two keys for `\d` and `\structline`:



```
\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}
```

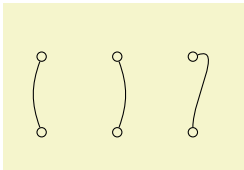
Dash patterns:

See the pgfmanual subsection 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}
```

5 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 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.

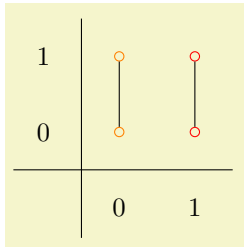
For options that only apply to pages, such as `keep changes`, you should say instead `\sseqset{pages={keep changes}}`.

`\foreach`

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

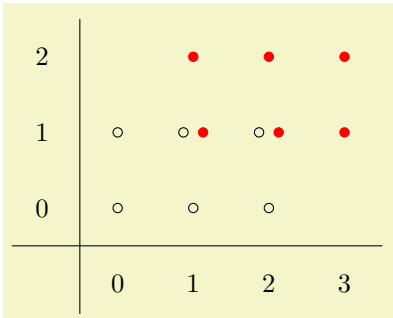
`\sseqnewcmd*⟨macro⟩(⟨x variable macro⟩,⟨y variable macro⟩)[⟨num args⟩]{⟨body⟩}`

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`:



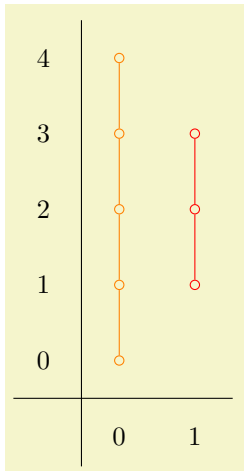
```
\sseqnewcmd\featuregroup{
  \class[\options](\x,\y)
  \class[\options](\x,\y+1)
  \structline(\x,\y)(\x,\y+1)
}
\begin{sseqpage}
\featuregroup[orange](0,0)
\featuregroup[red](1,0)
\end{sseqpage}
```

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.



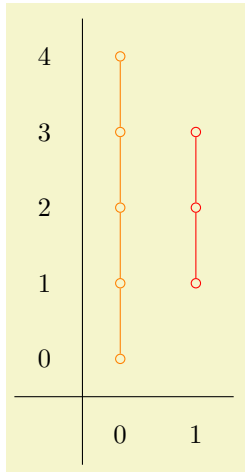
```
\sseqnewcmd*\tower[1]{
  \begin{scope}[\options]
  \class(\x,\y)
  \foreach \n in {1,...,#1}{
    \class(\x,\y+\n)
    \structline(\x,\y+\n-1)(\x,\y+\n)
  }
  \end{scope}
}
\begin{sseqpage}
\tower[orange](0,0){4}
\tower[red](1,1){2}
\end{sseqpage}
```

`\sseqnewgroup*[\langle num args \rangle]{\langle body \rangle}`

This is more or less a shorthand for

```
\sseqnewcmd*(\obscurexname,\obscureyname)[\langle num args \rangle]{
  \begin{scope}[xshift=\obscurexname,yshift=\obscureyname,\options]
  \langle body \rangle
  \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}
```

6 Global options

name= $\langle sseq\ name \rangle$ (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= $\langle page\ number \rangle$ (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 `\classes` that are not hit by differentials on pages less than r will be printed, as well as all `\structlines` 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= $\{\langle x\ degree \rangle\}\{\langle y\ degree \rangle\}$ (no default)

cohomological Serre grading (no value)

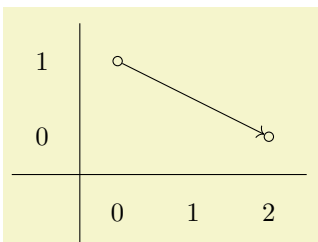
homological Serre grading (no value)

Adams grading (no value)

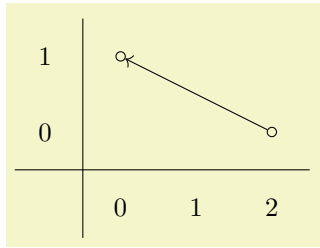
Specifies the degree of differentials. The $\langle x\ degree \rangle$ and $\langle y\ degree \rangle$ should both be mathematical expressions in one variable $\#1$ that evaluate to integers on any input. They specify the x and y displacement of a page $\#1$ differential. In practice, they will usually 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}=\{\langle x\ degree \rangle\}\{\langle y\ degree \rangle\}` or `\sseqset{Adams grading}` outside of the `sseqdata` and `sseqpages` environments.

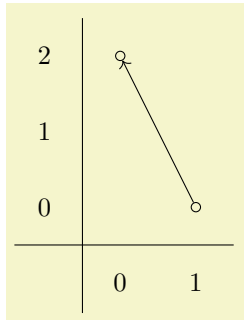
You can make a named grading convention by saying `\sseqset{my grading}/.sseq grading=\{\langle x\ degree \rangle\}\{\langle y\ degree \rangle\}`. Then later passing `my grading` to a spectral sequence is equivalent to saying `degree=\{\langle x\ degree \rangle\}\{\langle y\ degree \rangle\}`. The following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



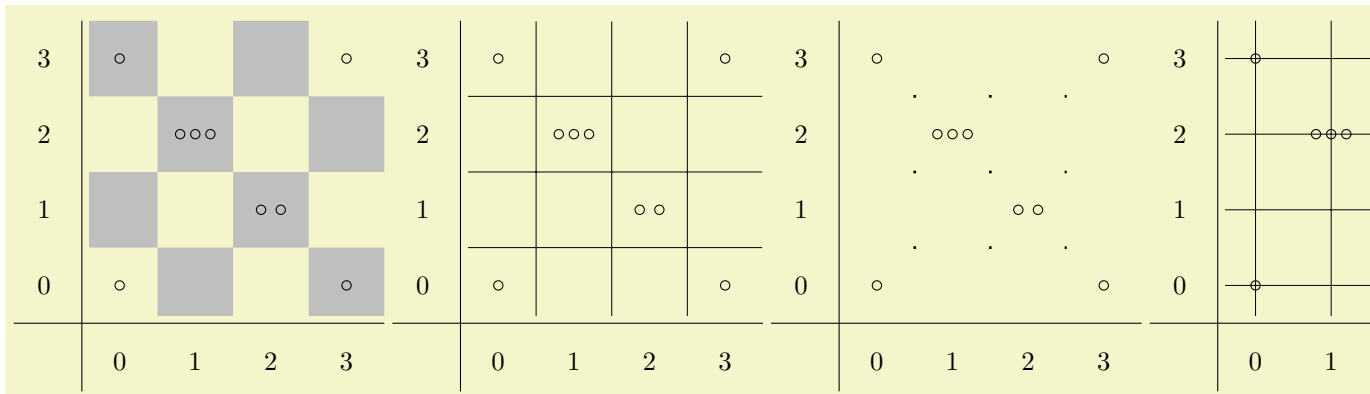
```
\begin{sseqpage}[Adams grading]% equivalent to degree={-1}{#1-1}
\class(0,2)
\class(1,0)
\d2(1,0)
\end{sseqpage}
```

x range= $\langle x \min \rangle \langle x \max \rangle$ (no default)
y range= $\langle y \min \rangle \langle y \max \rangle$ (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= $\{\langle grid \ style \rangle\}$ (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]
\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}
\printpage[name=grid example,grid=chess]
\printpage[name=grid example,grid=crossword]
\printpage[name=grid example,grid=dots]
\printpage[name=grid example,grid=go]
\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.

update existing

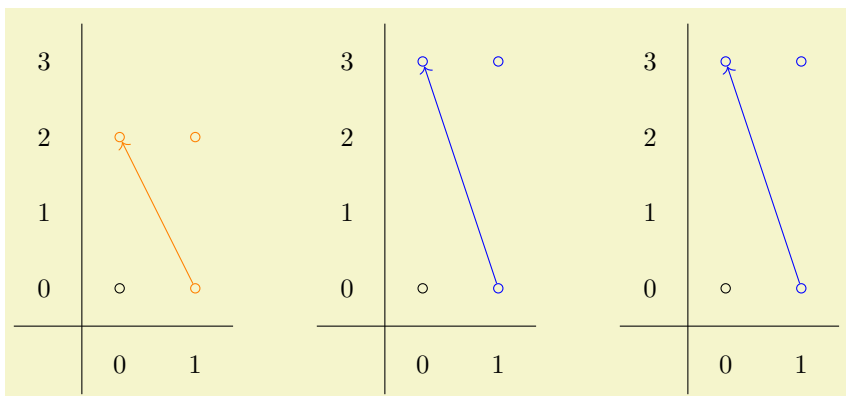
(no value)

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,paths=orange]
\class(0,2)
\class(1,2)
\classoptions[orange](1,0)
\d2(1,0)
\end{sseqpage}
%
\hskip1cm
%
\begin{sseqpage}[name=keep changes example,paths=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.

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 the 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.

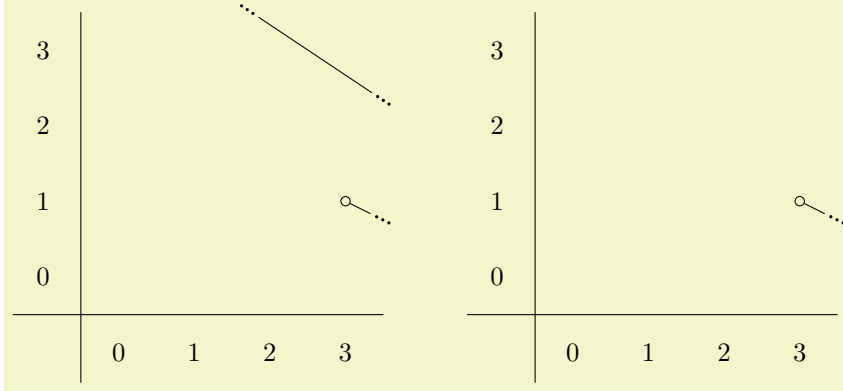
`no orphan edges`

(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.

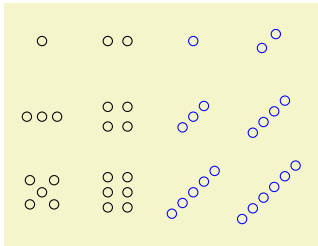


```
\begin{sseqdata}[name=orphan edges example,cohomological Serre grading,
x range={0}{3}, y range={0}{3}]
\class(1,4)\class(4,2)
\d3(1,4)
\class(3,1)\class(5,0)
\d2(3,1)
\end{sseqdata}
\printpage[name=orphan edges example]
\hskip1cm
\printpage[name=orphan edges example,no orphan edges]
```

`class pattern=<class pattern name>`

(no default)

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>}`

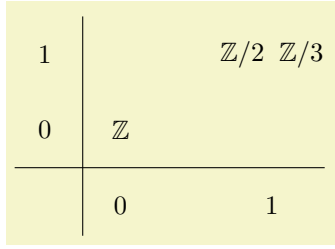
Creates a new class pattern. For example, the `linear` class pattern is created using the command:

```
\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);
}
```

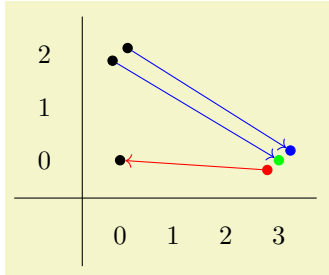
For instance the third row indicates that if there are three classes at the position (x,y) they should be printed at $(x-0.2,y)$, (x,y) , and $(x+0.2,y)$. You can give as many rows as you like; `sseqpages` will throw an error if there are more classes in any position than the maximum number that your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

`class placement transform={\langle transform keys \rangle}` (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=\langle boolean \rangle` (default true)(initially false)

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

6.1 Styles

The `sseqpages` package has a large number of “styles” which control the appearance of specific components (classes, differentials, or structlines) of a spectral sequence. These are named so that each command has a plural variant (e.g., `classes`) and a “style” variant (e.g., `class style`). The difference between these is always that `classes=\marg{keys}` adds the keys to the list of options used to style every class, whereas `class style=\marg{keys}` overwrites the list of options. It’s important to be aware when using the style variants that some of the styles are not empty when `sseqpages` is loaded, so for instance saying `class style={}` will change the appearance of the diagram. Generally, the plural versions are more useful, but in very large diagrams it can be noticeably faster to use the style variants.

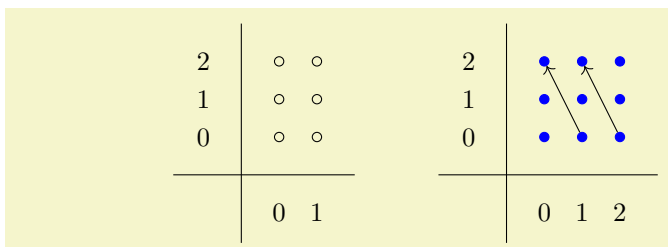
In cases where the same drawing feature is affected by multiple of these styles, the more specific style takes precedence.

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

`sseqs={\langle keys \rangle}` (no default)

`sseq style={\langle keys \rangle}` (no default)

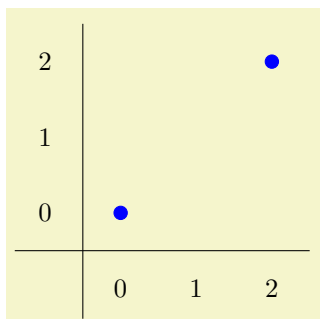
This passes global options to all future spectral sequences in the current scope. It is only useful to use this command with `\sseqset`. This is only really important for TikZ options, because for most options from `sseqpages` you can set a default directly by saying `\sseqset{key}={\langle value \rangle}`.



```
\sseqset{sseqs={scale=0.5}}% Applies to both of the two following sseqs
\begin{sseqpage}
\foreach \x in {0,1} \foreach \y in {0,1,2}{
  \class(\x,\y)
}
\end{sseqpage}
\hskip1cm
\begin{sseqpage}[Adams grading,classes={fill,blue}]
\foreach \x in {0,1,2} \foreach \y in {0,1,2}{
  \class(\x,\y)
}
\draw(1,0)
\draw(2,0)
\end{sseqpage}
```

`classes={⟨keys⟩}` (no default)
`cycles={⟨keys⟩}` (no default)
`class style={⟨keys⟩}` (no default)
`cycle style={⟨keys⟩}` (no default)

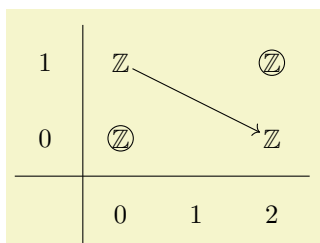
The keys `classes` and `cycles` are synonyms as are `class style` and `cycle style`. These options change the appearance of all classes. The options `classes` and `cycles` append whatever keys you give to the list of class style options, whereas `class style` and `cycle style` overwrite the list of styles.



```
\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)
`permanent class style={⟨keys⟩}` (no default)
`permanent cycle style={⟨keys⟩}` (no default)

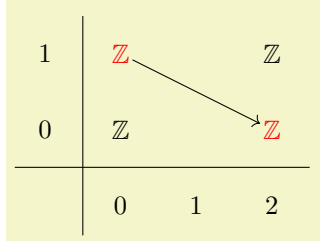
These options change the appearance of all permanent cycles (e.g., those classes which never support or are hit by a differential). For instance, we can circle the permanent cycles automatically. Note that because `permanent cycles` is more specific than `classes`, the `permanent cycles={draw}` command 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)
}
\draw(0,1)
\end{sseqpage}
```

`transient classes={\langle keys \rangle}` (no default)
`transient cycles={\langle keys \rangle}` (no default)
`transient class style={\langle keys \rangle}` (no default)
`transient cycle style={\langle keys \rangle}` (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.



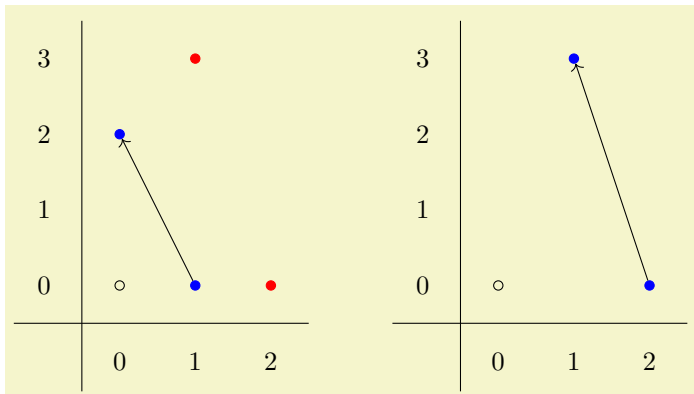
```

\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={\langle keys \rangle}` (no default)
`this page cycles={\langle keys \rangle}` (no default)
`this page class style={\langle keys \rangle}` (no default)
`this page cycle style={\langle keys \rangle}` (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={\langle keys \rangle}` (no default)
`edge style={\langle keys \rangle}` (no default)

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

`differentials={\langle keys \rangle}` (no default)
`differential style={\langle keys \rangle}` (no default)

`struct lines={\langle keys \rangle}` (no default)

`struct line style={⟨keys⟩}` (no default)

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

`this page struct line style={⟨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`.

`labels` (no value)

`label style` (no value)

`class labels` (no value)

`class label style` (no value)

`inner class labels` (no value)

`inner class label style` (no value)

`outer class labels` (no value)

`outer class label style` (no value)

`edge labels` (no value)

`edge label style` (no value)

`differential labels` (no value)

`differential label style` (no value)

`struct line labels` (no value)

`struct line label style` (no value)

6.2 Global Coordinate Transformations

Of the normal TikZ 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 TikZ 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.

6.3 Layout

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

Give a custom clipping. The clipping specified must be in the form of a valid TikZ path, for instance `\clip (0,0) rectangle (10,10);`. See the TikZ 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.

`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.

`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 clip axis padding=<dimension>` (no default, initially 0.1cm)
`y clip axis 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)

6.4 Axes Style

`x axis style=a` (no default, initially border)
`y axis style=a` (no default, initially border)
`axes style=` (no default, initially border)

`x axis origin=` (no default, initially 0)
`y axis origin=` (no default, initially 0)

`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)

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

<code>x label step=</code>	(no default, initially 1)
<code>y label step=</code>	(no default, initially 1)
<code>label step=</code>	(no default, initially 1)
 <code>rotate labels=</code> <i><code>boolean</code></i>	 (default true)(initially false)