Typesetting spectral sequences in LaTeX with sseq.sty

Tilman Bauer*

April 9, 2009

1 Introduction

The present package, `sseq`, facilitates the typesetting of mathematical objects called *spectral sequence charts* (henceforth simply called “chart”). From a typographical point of view, a chart is a two-dimensional grid with integer coordinates; at every position (x,y), there may be any number of symbols (usually dots, little circles or boxes, digits etc.), possibly decorated with labels, and between any two such symbols may or may not be a connection—e. g., a line, an arrow, or some curved line.

The `sseq` package is built on top of the pgf package by Till Tantau. It shares the disadvantage of using up lots of TeX's memory; therefore, if your system provides a `biglatex` or `hugelatex` command, it is wise to use it when using `sseq`. Still, problems may arise with very large charts (i.e., more than 100 × 100). Previous versions of `sseq` (pre-2.0) were based on the graphics package xy-Pic; the current version produces higher quality output and allows for more customization, at the cost of requiring a fairly recent TeXdistribution (or, at least, the packages `pgf` and `xkeyval` should be installed and the former should be no older than from 2006).

This package automates the following functions:

- Automatic drawing of the grid and the axis labels;
- Clipping. Anything outside the displayed portion of the chart is clipped away. This has the advantage that a large chart, which does not fit on a page, can be cut into smaller pieces which contain exactly the same `sseq` code, but different clipping regions.
- Arranging. Multiple symbols dropped at the same integer coordinates will be automatically arranged so that they usually do not overlap. The algorithm for doing this is rather primitive, but still powerful enough for most applications

---

*tilman@alum.mit.edu

- Simplified “turtle graphics” syntax. Every primitive element of a chart is typeset with a macro defined by `sseq`.
- Control structures (loops, if/then, etc.) are allowed inside the `sseq` environment.

2 Global options

The `sseq` package is loaded with
`\usepackage[`⟨*options...*⟩`]{sseq}`.
The following options are defined:

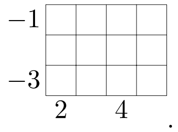
- `nops` is a deprecated option of a previous version. It is ignored.
- `nocolor` will tell `sseq` not to generate any color information. This includes gray shades. Many `dvi` drivers do not support color, but most of them tolerate (i.e. ignore) the color directives.
- `debug` will type out the pgf code generated by every `sseq` environment and wait for the user to confirm.

3 Basic usage

`sseq` A spectral sequence is typeset by the code
`\begin{sseq}[`⟨*options...*⟩`]{`⟨*x-range*⟩`}{`⟨*y-range*⟩`}`
⟨*sseq commands...*⟩
`\end{sseq}`

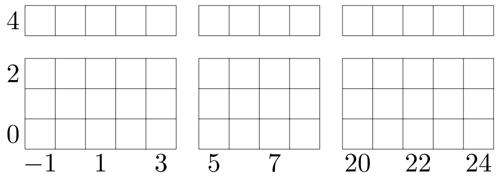
In the simplest case, a range is of the form ⟨*min*⟩...⟨*max*⟩, where ⟨*min*⟩ and ⟨*max*⟩ are two integers with ⟨*min*⟩ $\leq$ ⟨*max*⟩.

Thus `\begin{sseq}{2...5}{-3...-1}\end{sseq}` will typeset



It is also possible for ranges to be a comma-separated list of ranges of the above form, e.g. `0...3,8...10`. In this case, the chart is broken into several pieces. Here is an example:

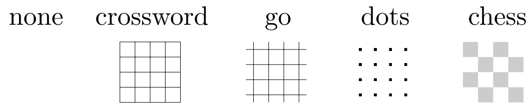
```
\begin{sseq}{-1...3,5...8,20...24}
            {0...2,4...4}
\end{sseq}
```



The minimum value of one block always has be at least two greater than the maximum of the previous block; `0...2,3...5` is illegal.

The following options are defined for the `sseq` environment. Options in bold face are the default setting.

`grid=` ⟨*none,**crossword**,go,dots,chess*⟩ Select an option of drawing the background grid.



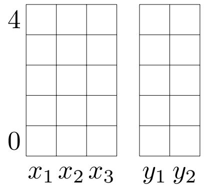
`entrysize=`⟨***size***⟩ Specify the size of each square of the grid. The default is 4mm.

`labels=`⟨***labels***⟩,`xlabels=`⟨***labels***⟩,`ylabels=`⟨***labels***⟩ Specify how labels on the x- and y-axis are drawn, respectively. The `labels` option sets both `xlabels` and `ylabels` to the same. Possible values are `none`, `numbers`, or an explicit list of semicolon-separated labels `{`$\langle x_1\rangle$`;`$\langle x_2\rangle$`;...}` which will be typeset in math mode. If the range consists of more than one block, the labels for the separate blocks are separated by a comma; see the example below. The default is `numbers`.

`labelstep=`⟨***step***⟩,`xlabelstep=`⟨***step***⟩,`ylabelstep=`⟨***step***⟩ Frequently it is not desirable that every label is printed, but only every second or third label. This can be done by setting this option to a positive integer. The default is 2.

The following example illustrates how labels can be customized:

```
\begin{sseq}[xlabelstep=1,ylabelstep=4,
             xlabels={x_1;x_2;x_3,y_1;y_2;y_3}]
            {1...3,10...11}{0...4}
\end{sseq}
```



`gapsize=`⟨***size***⟩,`xgapsize=`⟨***size***⟩,`ygapsize=`⟨***size***⟩ This sets the size of the gap between two blocks in a range (i.e. in the above example, the distance between the x_3 column and the y_1 column). The default is 3mm.

`gridstroke=`⟨***thickness***⟩ For the grid types `crossword` and `go`, this sets the line width. The default is .1pt. A good option is to set it to the resolution of your output device.

`leak=`⟨***size***⟩,`xleak=`⟨***size***⟩,`yleak=`⟨***size***⟩ When a line is drawn from within the visible range of the chart to a point outside, or vice versa, this line will protrude beyond the boundaries of the grid. These values define how far; the default is one third of `leak`.

`arrows=`⟨***arrow type***⟩ Sets the default arrow type to use in the spectral sequence. Here are some default arrow types:

⟶ `to`
→ `stealth`
⟶ `latex`

Other arrow types can be defined by the user or loaded from a library; see the pgf package documentation for details.

You may specify which algorithm you want to use to arrange multiple objects in a grid square. The following are possible:

- `\def\sseqpacking{\sspacksmart}`: the default, arrange dots roughly as on a dice
- `\def\sseqpacking{\sspackhorizontal}`: put all objects in a horizontal line, vertically centered.
- `\def\sseqpacking{\sspackvertical}`: put all objects in a vertical line, horizontally centered.
- `\def\sseqpacking{\sspackdiagonal}`: put all objects in a line going from top left to bottom right.

It is recommended to use the packing commands before the `\begin{sseq}` command; though it is possible to change the packing strategy within a single chart, the user has to take care to make sure that in every coordinate consistent packing strategies are used; the result will be unexpected otherwise.

4 sseq commands

Inside an `sseq` environment there is defined a virtual cursor, which starts out at position $(0,0)$ (even if that position is not within the visible region!). Most drawing commands are relative to the current cursor position; this facilitates reusage of `sseq` code when a certain pattern has to be repeated, as is often the case in mathematical spectral sequences.

`\ssmoveto` To move the cursor to the absolute position (x,y), use `\ssmoveto{x}{y}`.

`\ssmove` To move the cursor relative to the current position by (x,y), use `\ssmove{x}{y}`.

`\ssdrop` The command `\ssdrop[`⟨*optios*⟩`]{`⟨*mathcode*⟩`}` will display *mathcode* at the current cursor position. The argument is always interpreted in math mode. The following options can be given:

`circled` A circle is drawn around the object. If the object is a digit 1,…,9, a dingbat (eg. ①) is substituted.

`boxed` A box is drawn around the object.

`color=`⟨***color***⟩ The object is drawn in the specified color. Any LaTeX color can be used, e.g. predefined colors such as `black`, `blue`, `PineGreen`, etc., or user defined rgb colors.

`name=`⟨***name***⟩ This is equivalent to issuing `\ssname{`⟨*name*⟩`}` after `\ssdrop`, see below.

If the argument is `\bullet` or `\circle`, the object is replaced by a better-spaced graphics primitive.

`\ssname` `\ssname{`⟨*name*⟩`}` gives the object most recently dropped the name ⟨*name*⟩. If the previous command is one of the drop commands, then it refers to that object; if it is not, then if there is one and only one object at the current cursor position, it refers to that object; if that is also not the case, an error message is generated.

`\ssgoto` After an object has been given a ⟨*name*⟩ with `\ssname`, the cursor can be moved back to that object at any time by issuing `\ssgoto{`⟨*name*⟩`}`. This becomes necessary when there is more than one object in one position.

`\ssprefix` Often the mechanism provided by `\ssname`/`\ssgoto` is not flexible enough to deal with repeated code. In that case, `\ssprefix{`⟨*prefix*⟩`}` defines a prefix for all names that follow; i.e. a `\ssname{`⟨*name*⟩`}` after such a command will really define a name ⟨*prefix*⟩⟨*name*⟩. However, since `\ssgoto` also observes the prefix, `\ssgoto{`⟨*name*⟩`}` will still work. `\ssprefix` commands can be iterated; the prefices are then concatenated (most recent right).

`\ssresetprefix` This command resets the prefix defined by (a sequence of) `\ssprefix` to the empty prefix.

`\ssabsgoto` This is a version of `\ssgoto` that ignores the current prefix.

`\ssdroplabel` This command decorates the previously typeset object with a label. It is used in the form `\ssdroplabel[`⟨*options...*⟩`]{`⟨*label*⟩`}`.

The ⟨*label*⟩ will then be typeset next to the most recently dropped object (for a definition for what that is, exactly, consult the description of `\ssname`). If you specify one of `U`,`LU`,`RU`,`L`,`R`,`LD`,`RD`,`D` as an option, the label is positioned relative to the object it labels (default: `U`=up). As in `\ssdrop`, an option `color=`⟨*color*⟩ will typeset the label in the LaTeX color ⟨*color*⟩.

`\ssdropextension` This command has no arguments and is rather specialized. It refers to a previously dropped object (see `\ssname`), draws a circle around it, and considers that circle a new object. Thus it produces a new circle object that is attached to the original object, and not subject to the algorithm that tries to make objects non-overlapping.

`\ssstroke` There are two ways of typesetting connections between objects. One of them is the command `\ssstroke`, which requires that the cursor recently moved from one object to another. It takes no non-optional arguments and typesets a line between the two objects. Example: Suppose there are two objects, which have been given the names `a` and `b` by `\ssname`. Drawing a line between them is achieved by the command `\ssgoto{a} \ssgoto{b} \ssstroke`.

The following options can be given in the form `\ssstroke[options]`:

`color=`**color** the connection is drawn in the given LaTeX color `curve=`**value** the connection is curved to the left by an amount proportional to the value given.

`dashed`[=**dashing type**] the connection is drawn in a dashed style. The optional *dashing type* is an expression of the form `{{a}{b}...}`, where each of a, b, ... is a length (like 2pt or 3mm). The line then consists of a stroke of length `a`, followed by a gap of length `b`, followed by a stroke of length ... etc.

`dotted`[=**dashing type**] the connection is drawn in a dotted style. If the optional dashing type is given, this option behaves exactly like `dashed`.

`arrowfrom`[=**arrow style**] An arrow is drawn at the beginning of the line. The global arrow style can be overridden by specifying an arrow style.

`arrowto`[=**arrow style**] An arrow is drawn at the end of the line.

`void` This option says that the target of the connection is not another object; instead, the connection should just point into the correct direction.

`\ssarrowhead`
`\ssinversearrowhead`
Instead of specifying `arrowfrom` or `arrowto` in `ssstroke`, you can issue these commands afterwards to typeset an arrow head onto the beginning resp. end of the connection most recently typeset. There is one optional parameter `[`⟨*arrow style*⟩`]` which selects the arrow tips of style ⟨*arrow style*⟩, cf. the documentation of `\ssstroke`.

`\ssline` A second way of producing connections is `\ssline[`⟨*options...*⟩`]{`⟨*x*⟩`}{`⟨*y*⟩`}`. This command draws a connection from the most recent object (cf. `\ssname`) to an object at relative position (⟨*x*⟩,⟨*y*⟩), and it moves that cursor to that new position. If `\ssline` is followed by a drop command, then the line is attached to this newly dropped object (note the slightly out-of-order execution!), no matter how many other objects there are at the target position. However, if it is not followed by a drop command, then there has to be one and only one object at the target position, otherwise an error message is generated. The options are exactly the same as for `\ssstroke`.

`\ssarrow` `\ssarrow[`⟨*options...*⟩`]{`⟨*x*⟩`}{`⟨*y*⟩`}` is an abbreviation for
`\ssline[`⟨*options...*⟩`,arrowto]{`⟨*x*⟩`}{`⟨*y*⟩`}`

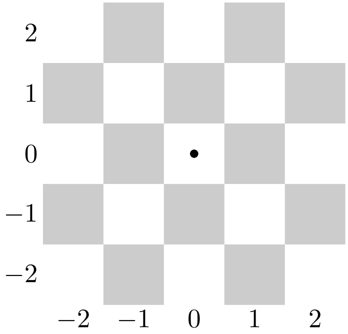
`\ssbullstring` `\ssbullstring{x}{y}{n}` is a shortcut for `\ssdrop{\bullet}` followed by $n-1$ copies of `\ssline{x}{y} \ssdrop{\bullet}`.

`\ssinfbullstring` `\ssinfbullstring{x}{y}{n}` is a shortcut for `\ssdrop{\bullet}` followed by $n-1$ copies of `\ssline{x}{y} \ssdrop{\bullet}`, followed by `\ssarrow[void]{x}{y}`. The cursor finishes on the last bullet.

5 Examples

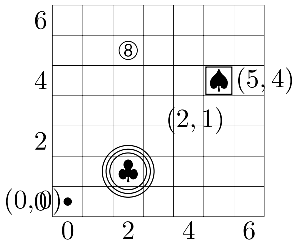
Example 1 *The following code generates a* 5×5 *grid with labels between* -2 *and* 2*. The size of every square is* $(.8cm)^2$*, and labels are written on every square. The grid is chess-style. A bullet is drawn at coordinate (0,0).*

```
\begin{sseq}[grid=chess,labelstep=1,
  entrysize=.8cm]{-2...2}{-2...2}
\ssdropbull
\end{sseq}
```



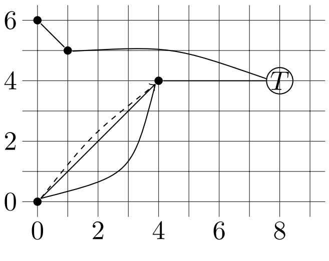
Example 2 *This example demonstrates how to move the cursor and drop objects and labels. Note how the last bullet, which is dropped at position (8,4), is outside the grid and thus clipped. The grid style is the default (*`crossword`*).*

```
\begin{sseq}{0...6}{0...6}
\ssdropbull
\ssdroplabel{(0,0)}
\ssmove 2 1
\ssdrop{\clubsuit}
\ssdropextension
\ssdropextension
\ssdropextension
\ssdroplabel[RU]{(2,1)}
\ssmove 0 4
\ssdropcircled{8}
\ssmoveto 5 4
\ssdropboxed{\spadesuit}
\ssdroplabel[R]{(5,4)}
\ssmove 3 0
\ssdropbull
\end{sseq}
```



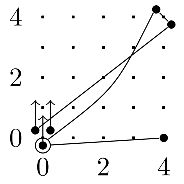
Example 3 *This example illustrates the different ways of drawing connections.*

```
\begin{sseq}[grid=go]{0...9}{0...6}
\ssdropbull
\ssmove 4 4
\ssdropbull \ssstroke
\sscurve{-.5}
\ssdashedcurve{.1} \ssarrowhead
\ssmove 4 0 \ssdropcircled{T}
\ssstroke
\ssmoveto 0 6
\ssdropbull
\ssline {1} {-1}
\sscurvedline 7 {-1} {.2}
\ssdropbull
\end{sseq}
```



Example 4 *This sample code shows how to use names for objects dropped in spectral sequence; this is particularly useful when more than one item is dropped at one position. It also demonstrates void arrows, which do not need a target.*

```
\begin{sseq}[grid=dots]{0...4}{0...4}
\ssdropbull \ssname{a} \ssvoidarrow 0 1
\ssdropbull \ssname{b} \ssvoidarrow 0 1
\ssdropbull \ssname{c} \ssvoidarrow 0 1
\ssdropextension \ssname{d}
\ssmove 4 4
\ssdropbull \ssname{e}
\ssdropbull \ssname{f}
\ssgoto a \ssgoto f \ssstroke
\ssgoto e \ssstroke
\ssgoto d \sscurve{.2}
\ssline 4 0 \ssdropbull
\end{sseq}
```



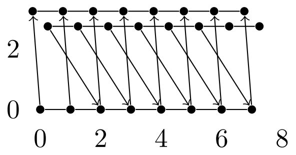
Example 5 *This final example shows how to take advantage of loops and prefices.*

```
\newcount\cnti
\def\drawstring#1#2{
 \ifnum#2=1
  \ssdropbull \ssname{#1}
 \else
  \ssdropbull \ssname{#1}
  \ssline 1 0
  \ssprefix{i}
  \cnti=#2
  \advance \cnti by -1
  \drawstring{#1}
   {\the\cnti}
 \fi
}
```

```
\def\drawlines#1#2#3{
 \ifnum#3>0
  \cnti=#3
  \ssgoto{#1} \ssgoto{#2}
  \ssstroke \ssarrowhead
  \ssprefix{i}
  \advance \cnti by -1
  \drawlines{#1}{#2}
   {\the\cnti}
 \fi
}
```

```
\begin{sseq}[grid=none]
    {0...8}{0...3}
\ssmoveto 0 3
\drawstring{a}{8}
\ssmoveto 0 3
\ssresetprefix
\drawstring{b}{8}
\ssmoveto 0 0
\ssresetprefix
\drawstring{c}{8}
\ssresetprefix
\drawlines{c}{a}{8}
\ssresetprefix
\drawlines{b}{iic}{6}
\end{sseq}
```

The result is shown in the following chart:



6 Final remarks

This package has been extremely helpful for my own mathematical work, and it most likely carries the characteristics of a tool initially developed for my own purposes only. Before this published version, there was an earlier version of `sseq` which was much less powerful; and what is worse, this version is not fully upward compatible with the previous one. (Every object that was dropped was forgotten right afterwards; thus connections could not properly connect objects but were always drawn from the center of the box corresponding to a coordinate to the center of the box corresponding to the target coordinate, resulting in fairly ugly pictures.)

Many things remain to be desired:

- `sseq` seems to be too voracious in memory
- While object are placed next to each other, no attempt is made not to make connections overlap
- A "grayout" option that automatically turns the source and target of any arrow into a different color would be highly useful for spectral sequence charts

Given time and leisure, I might or might not implement one or more of these improvements and make them available; of course, I would be even more happy if somebody else did it. (Needless to say, I would request that I be informed of and sent the enhancements.) I do guarantee that all further versions of `sseq` that might or might not be written by me will be compatible with the documented code written for this version.