Typesetting spectral sequences in LaTeX with sseq.sty

Tilman Bauer*

November 12, 2003

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 X$_Y$-pic package by K. H. Rose and R. Moore. 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).

This package automates the following functions and thereby offers an improvement over pure X$_Y$-pic graphics:

- 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

- Simplified "turtle graphics" syntax. Every primitive element of a chart is typeset with a macro defined by `sseq`; no cryptic command lines like in X$_Y$-pic.

- More flexibility. Control structures (loops, if/then, etc.) are allowed inside the `sseq` environment.

---

*tbauer@math.uni-muenster.de

2 Global options

The sseq package is loaded with

  \usepackage[*options...*]{sseq}.

  The following options are defined:

- **nops** will tell X_Y-pic not to generate PostScript code but to use special TEXfonts to typeset the diagrams. This is preferable if you do not have a PostScript printer; however, the overhead in memory and time usage is so enormous that this should only serve as a last resort against incompatibilities

- **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 typeout the X_Y-pic code generated by every sseq environment and wait for the user to confirm.

- any other options are fed into X_Y-pic. Thus, for example, the option **dvips** will tell X_Y-pic to generate code for the **dvips** driver.

3 Basic usage

A spectral sequence is typeset by the code

  \begin{sseq}{x}{y}
  (sseq commands...)
  \end{sseq}

  Here x and y are nonnegative integers specifying the size of the grid. By default, the bottom left corner of the chart will have coordinates $(0,0)$; however, this can be changed by issung

  \sseqxstart=*(starting x position)*
  \sseqystart=*(starting y position)*

  By default, the size of one coordinate square is .4cm; this can be changed by

  \sseqentrysize=*(new length)*

  Normally, all even labels are printed (for space reasons); this can be changed by

  \sseqxstep=*(x-axis label step)*
  \sseqystep=*(y-axis label step)*

  Setting these to 1 will typeset every label; setting it to n will typeset every nth label.

  All of the above have to be issued **before** the \begin{sseq} command. They remain valid for subsequent charts until changed again.

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 clipping region!). All 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 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{*mathcode*} will display *mathcode* at the current cursor position. The argument is always interpreted in math mode.
\ssdropbull	This command is a shorthand for and improvement over \ssdrop{\bullet}, i.e. it will typeset a thick black dot at the current cursor position. It is better than the longer version because the bounding circle of a \bullet is not accurate in most fonts. However, some dvi drivers do not support this and typeset a square instead of a bullet. If that is the case, either use a different driver (like dvips), or use \ssdrop{\bullet} instead at the cost of a slightly poorer display.
\ssdropboxed	This command is the same as \ssdrop, except that it will also draw a box around its argument.
\ssdropcircled	This command is the same as \ssdrop, except that it will also draw a circle around its argument. If the argument is a single digit, the symbol dropped will be prettified by replacing it with a character from the dingbats font.
\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 *prefixname*. 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 can be used either in the form

\ssdroplabel{*label*}

or in the form

\ssdroplabel[*pos*]{*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). The optional argument *pos*, which defaults to U, can be one of U,LU,RU,L,R,LD,RD,D, and denotes one of eight directions in which the label is positioned relative to the object it labels.

\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 suit of commands \ssstroke, \sscurve, \ssdashedstroke, \ssdashedcurve, \ssarrowhead, which all behave similarly. \ssstroke requires that the cursor recently moved from one object to another. It takes no arguments and typesets a

straight 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`.

`\sscurve`	This command behaves like `\ssstroke`, except that it takes an additional real number argument which denotes a curving factor for the line. 0 means straight.
`\ssdashedstroke`	Like `\ssstroke`, but typesets a dashed line.
`\ssdashedcurve`	Like `\sscurve`, but typesets a dashed curve.
`\ssarrowhead`	Typesets an arrow head onto the connection most recently typeset.
`\ssline`	The second way of producing connections is the suit of commands that follow, starting with `\ssline` x y. This command draws a straight line 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.
`\sscurvedline`	Like `\ssline`; the third argument is a real number denoting a curvature factor.
`\ssdashedline`	Like `\ssline`; typesets a curved line
`\sscurveddashedline`	Like `\sscurvedline`; typesets a dashed curve
`\ssarrow`	Like `\ssline`; but adds an arrowhead at the end
`\sscurvedarrow`	Like `\sscurve`; but adds an arrowhead at the end
`\ssdashedarrow`	Like `\ssdashedline`; but adds an arrowhead at the end
`\sscurveddashedarrow`	Like `\sscurveddashedline`; but adds an arrowhead at the end
`\ssvoidline`	This command behaves like `\ssline` except it disregards the target and just typesets a line to the given position. This is useful if the line is not supposed to have an object as its target. Unlike `\ssline` and its companions, it does **not** move the cursor.
`\ssvoidarrow`	Same as `\ssvoidline`, but typesets an arrow.
`\ssbullstring`	`\ssbullstring{x}{y}{n}` is a shortcut for `\ssdropbull` followed by n copies of `\ssline{x}{y} \ssdropbull`.
`\ssinfbullstring`	`\ssinfbullstring{x}{y}{n}` is a shortcut for `\ssdropbull` followed by n copies of `\ssline{x}{y} \ssdropbull`, followed by `\ssvoidarrow{x}{y}`.

5 Additional parameters

For typesetting dropped objects, `sseq` uses the color `\ssplacecolor`, for connections, `\ssconncolor`, and for labels, `\sslabelcolor`. These all default to black, but can be `\def`'d at any point to any valid X$_Y$-pic color. There are pre-defined colors `ssblack`, `sseqgr`, and `ssred`, denoting black, light gray, and red.

At any point, clipping can be switched on or off by setting
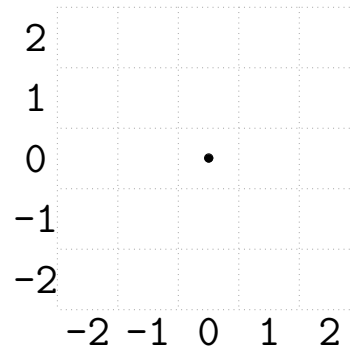`\setboolean{sseqclip}{true}` or `\setboolean{sseqclip}{false}`

6 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. A bullet is drawn at coordinate (0,0).*

```
\sseqxstart=-2
\sseqystart=-2
\sseqentrysize=.8cm
\sseqxstep=1
\sseqystep=1
\begin{sseq}{5}{5}
\ssdropbull
\end{sseq}
```
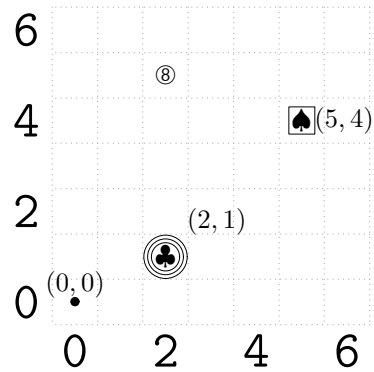
Example 2 *This example demonstrates how to move the cursor and drop objects and labels. Note how the last bullet, which is dropped at position (8,4), is outside the grid and thus clipped.*

```
\begin{sseq}{7}{7}
\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}
```
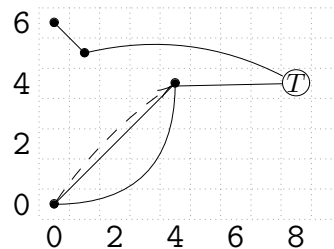
Example 3 *This example illustrates the different ways of drawing connections. Note how the size of the axis labels is adjusted to the grid size.*

```
\sseqentrysize=.4cm
\begin{sseq}{10}{7}
\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}
```
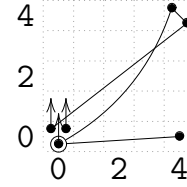
Example 4 *This sample code shows how to use names for objects dropped in*

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.
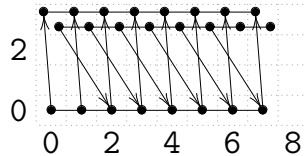
```
\begin{sseq}{5}{5}
\ssdropbull \ssname{a} \ssvoidarrow 0 1
\ssdropbull \ssname{b} \ssvoidarrow 0 1
\ssdropbull \ssname{c} \ssvoidarrow 0 1
\ssdropextension \ssname{d}
\ssmove 4 4
\ssdropbull \ssname{e}
\ssdropbull \ssname{f}
\ssgoto a \ssgoto f \ssstroke
\ssgoto e \ssstroke
\ssgoto d \sscurve{.2}
\ssline 4 0 \ssdropbull
\end{sseq}
```

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

```
\newcount\cnti              \def\drawlines#1#2#3{      \begin{sseq}{9}{4}
\def\drawstring#1#2{         \ifnum#3>0                \ssmoveto 0 3
 \ifnum#2=1                   \cnti=#3                  \drawstring{a}{8}
  \ssdropbull \ssname{#1}     \ssgoto{#1} \ssgoto{#2} \ssmoveto 0 3
 \else                        \ssstroke \ssarrowhead   \ssresetprefix
  \ssdropbull \ssname{#1}     \ssprefix{i}             \drawstring{b}{8}
  \ssline 1 0                 \advance \cnti by -1     \ssmoveto 0 0
  \ssprefix{i}                \drawlines{#1}{#2}       \ssresetprefix
  \cnti=#2                      {\the\cnti}            \drawstring{c}{8}
  \advance \cnti by -1      \fi                        \ssresetprefix
  \drawstring{#1}          }                           \drawlines{c}{a}{8}
   {\the\cnti}                                         \ssresetprefix
 \fi                                                   \drawlines{b}{iic}{6}
}                                                      \end{sseq}
```

The result is shown in the following chart:

7   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

- Other things ought to be customizable: The grid style, the axis labels, etc.

- 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.