The autobreak package*

Takahiro Ueda

19 June 2016

Abstract

This package implements a simple mechanism of line/page breaking within the `align` environment of the amsmath package; new line characters are considered as possible candidates for the breaks and the package tries to put breaks at adequate places. It is suitable for computer-generated long formulae with many terms.

Contents

1 Introduction

Sometimes people want to put long formulae in their documents, which do not fit in a line and may span over multiple pages. The following is an equation of explicitly writing down the first 50 terms in the sum of the well-known Basel problem:

$$\zeta(2) = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{25} + \frac{1}{36} + \frac{1}{49} + \frac{1}{64} + \frac{1}{81} + \frac{1}{100} + \frac{1}{121} + \frac{1}{144} + \frac{1}{169} + \frac{1}{196} + \frac{1}{225} + \frac{1}{256} + \frac{1}{289} + \frac{1}{324} + \frac{1}{361} + \frac{1}{400} + \frac{1}{441} + \frac{1}{484} + \frac{1}{529}$$

*This document corresponds to autobreak v0.1, dated 2016/06/03.

$$+\frac{1}{576}+\frac{1}{625}+\frac{1}{676}+\frac{1}{729}+\frac{1}{784}+\frac{1}{841}+\frac{1}{900}+\frac{1}{961}+\frac{1}{1024}+\frac{1}{1089}$$
$$+\frac{1}{1156}+\frac{1}{1225}+\frac{1}{1296}+\frac{1}{1369}+\frac{1}{1444}+\frac{1}{1521}+\frac{1}{1600}+\frac{1}{1681}+\frac{1}{1764}$$
$$+\frac{1}{1849}+\frac{1}{1936}+\frac{1}{2025}+\frac{1}{2116}+\frac{1}{2209}+\frac{1}{2304}+\frac{1}{2401}+\frac{1}{2500}+\dots \quad (1)$$

The above example might seem nonsense, but putting long formulae may have a meaning in some cases and become inevitable for completeness of documents, writing self-contained papers, or just to impress readers. They are typically generated as outputs of computer algebra systems, and would have the form of a sum of many terms while each term is short.

Then, the question is how to break long formulae in such a way that the expressions do not make any overfull lines for LaTeX. Certainly, one can attempt to manually insert line breaks by trial and error, checking whether LaTeX warns overfull lines, and this process could be automatized by external scripts at some extent. A shortcoming of such 'manual' approaches is that line breaks have to be reexamined whenever the layout of the document is changed, e.g., replacing the document class or reusing existing equations into another document with a different format.

The goal of the autobreak package is to give a reasonably simple solution for (semi-)automatic line breaking of long formulae within LaTeX[1].

2 Usage

The autobreak package is supposed to be used together with the amsmath package[2][3]:

```
\usepackage{amsmath}
\usepackage{autobreak}
```

When your document contains long equations over multiple pages, you might want to use `\allowdisplaybreaks` of amsmath package:

```
\allowdisplaybreaks
```

```
\begin{autobreak}
⟨long-equations⟩
\end{autobreak}
```

[1] There is another package breqn (https://www.ctan.org/pkg/breqn), which adopts a more automatic fashion and is useful for more sophisticated line breaking, unless you get "Dimension too large" error for really big expressions.

[2] https://www.ctan.org/pkg/amsmath.

[3] Actually autobreak internally loads amsmath, but it is still a good practice to explicitly include all packages providing macros used in your document.

The `autobreak` environment is used for breaking lines in long formulae in the `align` environment of `amsmath`[4].

```
\begin{align}
  \begin{autobreak}
    \zeta(3) =
    1
    + \frac{1}{8}
    + \frac{1}{27}
    + \frac{1}{64}
    + \frac{1}{125}
    + \frac{1}{216}
    + \frac{1}{343}
    + \frac{1}{512}
    + \frac{1}{729}
    + \frac{1}{1000}
    + \frac{1}{1331}
    + \frac{1}{1728}
    + \frac{1}{2197}
    + \frac{1}{2744}
    + \frac{1}{3375}
    + \frac{1}{4096}
    + \frac{1}{4913}
    + \frac{1}{5832}
    + \frac{1}{6859}
    + \frac{1}{8000}
    + \dots
  \end{autobreak}
\end{align}
```

$$\begin{aligned}\zeta(3) = 1 &+ \frac{1}{8} + \frac{1}{27} + \frac{1}{64} + \frac{1}{125} \\ &+ \frac{1}{216} + \frac{1}{343} + \frac{1}{512} + \frac{1}{729} \\ &+ \frac{1}{1000} + \frac{1}{1331} + \frac{1}{1728} \\ &+ \frac{1}{2197} + \frac{1}{2744} + \frac{1}{3375} \\ &+ \frac{1}{4096} + \frac{1}{4913} + \frac{1}{5832} \\ &+ \frac{1}{6859} + \frac{1}{8000} + \dots \end{aligned} \tag{2}$$

The magic happens from the simple fact that `autobreak` interprets all new line characters appearing between `\begin{autobreak}` and `\end{autobreak}` as *breakable points*, at which any line breaks can be logically inserted. To be more exact, the first non-empty block, separated from the rest by a new line character, determines the indentation of the successive lines. Then `autobreak` tries to fill the line with the rest of the blocks, and puts a line break when they do not fit in a line. This is clarified by the following example:

[4] Technically, `align` (with `\notag` to suppress equation numbers except the last line) is the only option we can use for page-break aligned equations within `amsmath` because `split`, `gathered`, `aligned` and `alginedat` do not allow page breaking. `dmath` of `breqn` with `\eqinterlinepenalty=0` allows page breaking, but may fail to find a reasonable tag place.

```
\begin{align}
  \begin{autobreak}
    \NumberedBox{1}
    \NumberedBox{2}
    \NumberedBox{3}
    \NumberedBox{4}
    \NumberedBox{5}
    \NumberedBox{6}
    \NumberedBox{7}
    \NumberedBox{8}
    \NumberedBox{9}
    \NumberedBox{10}
    \NumberedBox{11}
  \end{autobreak}
\end{align}
```

$$\boxed{1}\ \boxed{2}\ \boxed{3}\ \boxed{4}\ \boxed{5}$$
$$\boxed{6}\ \boxed{7}\ \boxed{8}\ \boxed{9}$$
$$\boxed{10}\ \boxed{11} \tag{3}$$

It is also possible to put more than one `autobreak` in one `align`:

```
\begin{align}
  \begin{autobreak}
    \NumberedBox{1} =
    \NumberedBox{2}
    + \NumberedBox{3}
    + \NumberedBox{4}
    + \NumberedBox{5}
    + \NumberedBox{6}
    + \NumberedBox{7}
    + \NumberedBox{8}
    + \NumberedBox{9}
    + \NumberedBox{10} ,
  \end{autobreak}
  \\
  \begin{autobreak}
    \LongerNumberedBox{1} =
    \NumberedBox{2}
    + \NumberedBox{3}
    + \NumberedBox{4}
    + \NumberedBox{5}
    + \NumberedBox{6}
    + \NumberedBox{7}
    + \NumberedBox{8}
    + \NumberedBox{9}
    + \NumberedBox{10} .
  \end{autobreak}
\end{align}
```

$$\boxed{1} = \boxed{2} + \boxed{3} + \boxed{4}$$
$$+ \boxed{5} + \boxed{6}$$
$$+ \boxed{7} + \boxed{8}$$
$$+ \boxed{9} + \boxed{10}, \tag{4}$$
$$\boxed{1} = \boxed{2} + \boxed{3} + \boxed{4}$$
$$+ \boxed{5} + \boxed{6}$$
$$+ \boxed{7} + \boxed{8}$$
$$+ \boxed{9} + \boxed{10}. \tag{5}$$

For a technical reason, it often requires more than one run of LaTeX, and in such

cases one will get informed by the following warning:

```
Package autobreak Warning: Layout may have changed.
(autobreak)                Rerun to get layout correct.
```

In the next run, the layout of the equations will be corrected.

`\everybeforeautobreak {`⟨*tokens*⟩`}`
`\everyaftereautobreak {`⟨*tokens*⟩`}`

They specify token lists inserted before and after automatically inserted line breaks in `autobreak`. For example,

```
\begin{align}
  \everyafterautobreak{\times}
  \begin{autobreak}
    \cos\left(\frac{\pi x}{2}\right) =
    \left(1-x^2\right)
    \left(1-\frac{x^2}{9}\right)
    \left(1-\frac{x^2}{25}\right)
    \left(1-\frac{x^2}{49}\right)
    \left(1-\frac{x^2}{81}\right)
    \left(1-\frac{x^2}{121}\right)
    \left(1-\frac{x^2}{169}\right)
    \left(1-\frac{x^2}{225}\right)
    \left(1-\frac{x^2}{289}\right)
    \left(1-\frac{x^2}{361}\right)
    \left(1-\frac{x^2}{441}\right)
    \dots
  \end{autobreak}
\end{align}
```

$$\begin{aligned}\cos\left(\frac{\pi x}{2}\right) = {} & \left(1-x^2\right)\left(1-\frac{x^2}{9}\right)\left(1-\frac{x^2}{25}\right)\left(1-\frac{x^2}{49}\right)\left(1-\frac{x^2}{81}\right) \\ & \times\left(1-\frac{x^2}{121}\right)\left(1-\frac{x^2}{169}\right)\left(1-\frac{x^2}{225}\right)\left(1-\frac{x^2}{289}\right) \\ & \times\left(1-\frac{x^2}{361}\right)\left(1-\frac{x^2}{441}\right)\dots \end{aligned} \tag{6}$$

3 Caveats

Because `autobreak` tries to insert line breaks at any of new line characters, you must not make any new lines at which the line cannot be broken[5]. For example

[5] You may put "%" at the end of the line to avoid a new line.

```
\begin{align}
  \begin{autobreak}
    x =
    % A problematic line break.
    \frac{1}
    {2} .
  \end{autobreak}
\end{align}
```

gives an error in the typesetting:

```
! Missing } inserted.
<inserted text>
                }
l.8   \end{align}
```

Putting '\\' or '&' inside `autobreak`, which tries to insert these special stuffs automatically, also causes typesetting errors.

The `autobreak` environment uses `\linewidth` as the maximum width that expressions in its body can occupy. There is no way for `autobreak` to know how much other formulae consume the space outside it. Therefore it fails to determine the adequate maximum width when there are expressions outside `autobreak` and then LaTeX gives overfull line warnings:

```
\begin{align}
  \text{some stuff outside autobreak}
  \begin{autobreak}
    \text{LHS} =
    ...
  \end{autobreak} .  % Even just a "." can be problematic.
\end{align}
% May give overfull line warnings
```

One may want to separate long formulae from the main document file to other files and include them via `\input{`⟨*file*⟩`}`, for example,

```
\begin{align}
  \begin{autobreak}
    \input{longeqn.inc}  % It works!
  \end{autobreak}
\end{align}

\begin{align}
  \begin{autobreak}
    lhs =
    \input{longrhs.inc}  % It also works!
    .
  \end{autobreak}
\end{align}
```

The current version of `autobreak` supports these cases: the file content of `\input{`⟨*file*⟩`}` is expanded before recognizing the lines, with the help of the catchfile package[6], when it appears at the beginning of each line. But it does not support `\input{`⟨*file*⟩`}` in the middle of the lines:

```
\begin{align}
  \begin{autobreak}
    x + \input{longexpr.inc}  % Sorry, it does not work.
  \end{autobreak}
\end{align}
```

The difficulty comes from the fact that it needs to be expanded before `autobreak` scans lines. By the same reason, `autobreak` fails to detect new lines defined inside macros[7]:

```
\newcommand{\foo}{
  a
  + b
  + c
  + d
}
\begin{align}
  \begin{autobreak}
    \foo + \foo + \foo + \foo  % No new lines can be detected.
  \end{autobreak}
\end{align}
```

4 Implementation

```
1 ⟨*package⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{autobreak}%
4   [2016/06/03 v0.1 simple line breaking of long formulae]
```

4.1 Registers and constants

`\everybeforeautobreak` The list of tokens that gets inserted before every line break generated by `autobreak`.

```
5 \newtoks\everybeforeautobreak
```

`\everyafterautobreak` The list of tokens that gets inserted after every line break generated by `autobreak`.

```
6 \newtoks\everyafterautobreak
```

[6] https://www.ctan.org/pkg/catchfile.

[7] Actually, when the definition of `\foo` is parsed, the new line characters inside it are usually lost.

`\@autobreak@alltoks` The token register to store the whole result of `autobreak`.

```
\newtoks\@autobreak@alltoks
```

`\@autobreak@linetoks` The token register for the current line.

```
\newtoks\@autobreak@linetoks
```

`\@autobreak@lhswidth` The width of the current left-hand side.

```
\newdimen\@autobreak@lhswidth
```

`\@autobreak@rhswidth` The width of the current right-hand side.

```
\newdimen\@autobreak@rhswidth
```

`\@autobreak@maxlhswidth` The width of the longest left-hand side occupied. Affected by the `.aux` file generated in the previous run.

```
\newdimen\@autobreak@maxlhswidth
```

`\@autobreak@realmaxlhswidth` The width of the longest left-hand side occupied. Not affected by the `.aux` file generated in the previous run.

```
\newdimen\@autobreak@realmaxlhswidth
```

`\@autobreak@maxrhswidth` The maximum width that the right-hand sides can occupy.

```
\newdimen\@autobreak@maxrhswidth
```

`\@autobreak@sep` The additional space needed for putting the left-hand side and the right-hand side in one line, in other words, the separation for `&`. TODO: how can we know the exact extra space to be inserted?

```
\def\@autobreak@sp{7\p@}
```

`\@autobreak@newlinechar` The macro representing an active `^^M`.

```
\begingroup
  \catcode`\^^M=\active
  \gdef\@autobreak@newlinechar{^^M}
\endgroup
```

4.2 Interaction with `.aux` files

When there are two or more `autobreak` in one `align`, each `autobreak` has to know the maximum width of the left-hand side of the all `autobreak` in the same `align`. Instead of violating 'causality' (e.g., how LaTeX parses a file from the beginning to the end), we use `.aux` file to store the maximum left-hand side width, which provides the correct value in the next run.

`\if@autobreak@invalidlayout` The switch to be turned on when an invalid layout is detected.

```
\newif\if@autobreak@invalidlayout
\@autobreak@invalidlayoutfalse
```

Show a warning if the user needs to rerun.

```
\AtEndDocument{%
  \if@autobreak@invalidlayout
    \if@filesw
      \PackageWarningNoLine{autobreak}{Layout may have changed.
        \MessageBreak Rerun to get layout correct}%
    \else
      \PackageWarningNoLine{autobreak}{Layout may be wrong}%
    \fi
  \fi
}
```

`\@autobreak@getmaxlhswidth` To be expanded to a value saved in `.aux` in the previous run, or `0pt` if not found.

```
\def\@autobreak@getmaxlhswidth#1{%
  \@ifundefined{@autobreak@w@#1}{%
    \z@
  }{%
    \@nameuse{@autobreak@w@#1}%
  }%
}
```

`\@autobreak@setmaxlhswidth` Called in `.aux`.

```
\def\@autobreak@setmaxlhswidth#1#2{%
  \global\@namedef{@autobreak@w@#1}{#2}%
}
```

`@autobreak@eqnindex` The counter to identify each `align`.

```
\newcounter{@autobreak@eqnindex}
```

`@autobreak@subeqnindex` The counter to store the number of `autobreak` in an `align`.

```
  \newcounter{@autobreak@subeqnindex}[@autobreak@eqnindex]%
```

`\@autobreak@loadmaxlhswidth` Loads `\@autobreak@maxlhswidth` for the next `align`.

```
\def\@autobreak@loadmaxlhswidth{%
  \stepcounter{@autobreak@eqnindex}%
  \@autobreak@maxlhswidth=%
    \@autobreak@getmaxlhswidth{\arabic{@autobreak@eqnindex}}%
  \@autobreak@realmaxlhswidth=\z@
}
```

`\@autobreak@savemaxlhswidth` Saves `\@autobreak@realmaxlhswidth` for the next run.

```
\def\@autobreak@savemaxlhswidth{%
  \ifnum\arabic{@autobreak@subeqnindex}>0
    \ifdim\@autobreak@maxlhswidth=\@autobreak@realmaxlhswidth
    \else
```

We have used the wrong value of `\@autobreak@maxlhswidth` (was too much). Need to rerun.

```
      \global\@autobreak@invalidlayouttrue
    \fi
```

Note that `\@autobreak@maxlhswidth` becomes problematic only when two or more `autobreak` appear in one `align`. In the case with one `autobreak`, the default value `0pt` is safe for the next run.

```
    \ifnum\arabic{@autobreak@subeqnindex}>1
      \if@filesw
```

We should provide `\@autobreak@setmaxlhswidth` in `.aux`.

```
        \@ifundefined{@autobreak@auxinited}{%
          \immediate\write\@mainaux{%
            \string\providecommand
              \string\@autobreak@setmaxlhswidth[2]{}%
          }%
          \gdef\@autobreak@auxinited{}%
        }{}%
        \immediate\write\@auxout{%
          \string\@autobreak@setmaxlhswidth%
            {\arabic{@autobreak@eqnindex}}%
            {\the\@autobreak@realmaxlhswidth}%
        }%
      \fi
    \fi
  \fi
}
```

4.3 Hacking **amsmath**

```
\RequirePackage{amsmath}
```

`\if@autobreak@newlinedef` The switch to be turned on when `\@autobreak@newlinedef` applies.

```
\newif\if@autobreak@newlinedef
\@autobreak@newlinedeffalse
```

`\@autobreak@newlinedef` Installs the definition of `^^M` as a space. This is virtually harmless in math mode.

```
\begingroup
  \catcode`\^^M=\active
  \gdef\@autobreak@newlinedef{%
    \def^^M{ }%
    \@autobreak@newlinedeftrue
  }
\endgroup
```

`\collect@body` We need to override `\collect@body` such that it keeps `^^M`.

```
\def\collect@body#1{%
  \@envbody={\expandafter#1\expandafter{\the\@envbody}}%
  \edef\process@envbody{%
    \the\@envbody\noexpand\end{\@currenvir}%
  }%
  \@envbody=\@emptytoks
  \def\begin@stack{b}%
  \begingroup
```

Actually, the following three lines need to be inserted to the original code.

```
    \if@autobreak@newlinedef
      \catcode`\^^M=\active
    \fi
    \expandafter\let\csname\@currenvir\endcsname=\collect@@body
    \edef\process@envbody{%
      \expandafter\noexpand\csname\@currenvir\endcsname
    }%
    \process@envbody
}
```

`align` Hack `align` of amsmath.

```
\let\@autobreak@oldstart@align=\start@align
\def\start@align{%
  \@autobreak@loadmaxlhswidth
  \@autobreak@newlinedef
  \@autobreak@oldstart@align
}
```

```
\let\@autobreak@oldendalign=\endalign
\def\endalign{%
  \@autobreak@savemaxlhswidth
  \@autobreak@oldendalign
}
```

4.4 `autobreak` environment

`autobreak` Checks if we are in `align` (and `\@autobreak@newlinedef` is applied), increments the counter and collects its body via `\collect@body`.

```
\newenvironment{autobreak}{%
  \if@autobreak@newlinedef
  \else
    \PackageError{autobreak}{%
      autobreak is not allowed here
    }{%
      Use autobreak inside align.
    }%
  \fi
  \stepcounter{@autobreak@subeqnindex}%
  \collect@body\@autobreak
}{}
```

`\@autobreak` Called from `\collect@body`. The parameter `#1` is the whole body. It takes also `#2` and `#3`, which are always `\end` and `autobreak`, to remove them from the successive tokens.

```
\def\@autobreak#1#2#3{%
```

First, close the group of `autobreak`.

```
  \end{autobreak}%
```

Then parse the given body of the environment and construct lines to be passed to `align`.

```
\@autobreak@init
\def\@tempa{\expandafter\@autobreak@scanline
  \@autobreak@newlinechar#1}%
\expandafter\@tempa\@autobreak@newlinechar\@autobreak@end
}
```

`\@autobreak@init` Initialization.

```
\def\@autobreak@init{%
  \@autobreak@alltoks={}%
  \@autobreak@linetoks={}%
  \@autobreak@lhswidth=\z@
}
```

`\@autobreak@end` Finalization. It generates the whole lines in one go.

```
\def\@autobreak@end{%
  \expandafter\@autobreak@addtoks\expandafter\@autobreak@alltoks
    \expandafter{\the\@autobreak@linetoks}%
  \the\@autobreak@alltoks
}
```

`\@autobreak@scanline` Takes a line from the input stream. Here a line ends with `^^M`.

```
\begingroup
  \catcode`\^^M=\active
  \gdef\@autobreak@scanline#1^^M{\@autobreak@scanline@{#1}}
\endgroup
```

If the next token is a punctuation, then we merge it into the current line. (Otherwise it can make a line only with a period, for example).

```
\def\@autobreak@scanline@#1{%
  \@autobreak@ifnextpunct{%
    \@autobreak@scanline@gobble{#1}%
  }{%
    \@autobreak@scanline@@{#1}%
  }%
}
```

A helper macro of `\@ifnextpunct{`⟨*if-yes*⟩`}{`⟨*if-no*⟩`}`.

```
\def\@autobreak@ifnextpunct#1#2{%
  \@ifnextchar.{%
    #1%
  }{%
    \@ifnextchar,{%
      #1%
    }{%
      \@ifnextchar;{%
        #1%
      }{%
        \@ifnextchar:{%
```

```
          #1%
        }{%
          #2%
        }%
      }%
    }%
  }%
}
```

Merge punctuations as possible (usually there is only one period in a line, though).

```
\def\@autobreak@scanline@gobble#1#2{%
  \@autobreak@ifnextpunct{%
    \@autobreak@scanline@gobble{#1#2}%
  }{%
    \@autobreak@scanline@@{#1#2}%
  }%
}
```

Pass the current line to `\@autobreak@processline`. Then, repeat scanning lines until `\@autobreak@end` appears as the next token.

```
\def\@autobreak@scanline@@#1{%
  \@autobreak@processline{#1}%
  \@ifnextchar\@autobreak@end{}{%
    \@autobreak@scanline@@@
  }%
}
```

One may expect `\input{`⟨*file*⟩`}` in `autobreak` is expanded by the file content and `autobreak` treats new lines in it correctly. But it needs more work. Because handling of `\input` in the middle of the lines is rather involved, for now we support only `\input` at the beginning of each line (which is what sane people usually do). This can be done via the catchfile package.

```
\IfFileExists{catchfile.sty}{
  \RequirePackage{catchfile}
  \def\@autobreak@scanline@@@{%
    \@ifnextchar\input{%
      \@autobreak@scanline@input
    }{%
      \@autobreak@scanline
    }%
  }
}{
  \def\@autobreak@scanline@@@{%
    \@ifnextchar\input{%
      \PackageWarning{autobreak}{%
        Cannot handle new lines in a file via \protect\input,
        \MessageBreak which requires the catchfile package
      }%
    }%
    \@autobreak@scanline
  }
```

```
}
```

The argument `#1` is `\input` and `#2` is the file name.

```
\def\@autobreak@scanline@input#1#2{%
  \CatchFileDef\@tempa{#2}{\catcode`\^^M=\active}%
  \expandafter\@autobreak@scanline\@tempa
}
```

`\@autobreak@processline` Each line from `\autobreak@scanline` should be regarded as a 'block' in the equation. The first block (typically the left-hand side + '=') determines the indentation for the successive lines. From the second block, try to append the block to the end of the line and insert a line break if it does not fit in a line.

```
\def\@autobreak@processline#1{%
  \ifdim\@autobreak@lhswidth=\z@
```

For the first block. The rest of the width for the right-hand sides is determined from `\linewidth` and `\@autobreak@maxlhswidth`.

```
    \@autobreak@settowidth\@autobreak@lhswidth{#1}%
    \ifdim\@autobreak@lhswidth>\z@
      \ifdim\@autobreak@lhswidth>\@autobreak@maxlhswidth
        \ifdim\@autobreak@maxlhswidth>\z@
```

The previous one used the wrong value of `\@autobreak@maxlhswidth` (was too short). Need to rerun.

```
          \global\@autobreak@invalidlayouttrue
        \fi
        \global\@autobreak@maxlhswidth=\@autobreak@lhswidth
      \fi
      \ifdim\@autobreak@lhswidth>\@autobreak@realmaxlhswidth
        \global\@autobreak@realmaxlhswidth=\@autobreak@lhswidth
      \fi
      \@autobreak@maxrhswidth=\linewidth
      \advance\@autobreak@maxrhswidth by -\@autobreak@maxlhswidth
      \advance\@autobreak@maxrhswidth by -\@autobreak@sp
      \@autobreak@alltoks={#1{}&}%
    \fi
  \else
```

For the rest of the blocks.

```
    \@autobreak@settowidth\@autobreak@rhswidth
      {\the\@autobreak@linetoks#1\the\everybeforeautobreak}%
    \ifdim\@autobreak@rhswidth>\@autobreak@maxrhswidth
```

Adding the next block gives an overfull line. Need a line break.

```
      \edef\@tempa{\the\@autobreak@linetoks\the\everybeforeautobreak}%
      \expandafter\@autobreak@addtoks\expandafter\@autobreak@alltoks
        \expandafter{\@tempa\notag\\&}%
      \@autobreak@linetoks=\everyafterautobreak
    \fi
    \@autobreak@addtoks\@autobreak@linetoks{#1}%
  \fi
}
```

`\@autobreak@addtoks` Appends `#2` to the token register `#1`.

```
\def\@autobreak@addtoks#1#2{%
  #1=\expandafter{\the#1#2}%
}
```

`\@autobreak@settowidth` Same as `\settowidth` but in math mode. We assume `\displaystyle`. (Anyway `align` issues `\displaystyle` at the beginning of every cell.)

```
\def\@autobreak@settowidth#1#2{%
  \settowidth#1{$\displaystyle#2$}%
}
```

⟨/package⟩

Change History

v0.1

General: Initial version 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols