Anchored Figures at Either Margin

Daniel Comenetz

1 Setting Figures

A figure in a box can be placed in text at one margin or the other, by measuring the box and adjusting the paragraph shape parameters so as to allow room for it. Macros that try to accomplish this automatically must be resourceful enough to decide what to do in a variety of special circumstances; the correctness or appropriateness of each decision depends on the requirements of the user. In my case, I need to set figures in mathematics text, and I have to be concerned with three problems, namely: (1) what happens if there isn't enough room left on the page for the figure at the point in the text where it is requested; (2) shape parameters are cleared at the start of a paragraph, so how are they to be reset if a new paragraph, possibly preceded by vertical space, is begun partway down the figure; (3) TEX assumes a math display will last for 3 lines ([103]†), but if a math display occurring to the side of the figure takes more than 3 lines (perhaps the figure is intended to illuminate the long calculation), how are the shape parameters to be reset correctly after the display.

One approach to the first problem is, to let the figures "float" to wherever there is room for them. The `\topinsert` and `\midinsert` commands of plain TEX can be used to place figures in this way, although of course those commands do not try to surround the figure with automatically shaped paragraphs. One says a figure inserted that way floats in the main vertical list. Thomas Reid wrote macros (*TUGboat* 8, no. 3, p. 315) which, instead of using insertions, place figures at the right while letting them float in the list of paragraphs. His macros were extended and adapted for LATEXby Thomas Kneser (*TUGboat* 12, no. 1, p. 28). They do shape the text surrounding the figure, and they provide a solution of problem (2), at least for paragraphs which are separated by normal `\parskip` glue. But when the typesetting is finished, if a figure which is supposed to clarify or is repeatedly referred to in a portion of text has floated off some distance away from it, the reader may find such separation inconvenient.

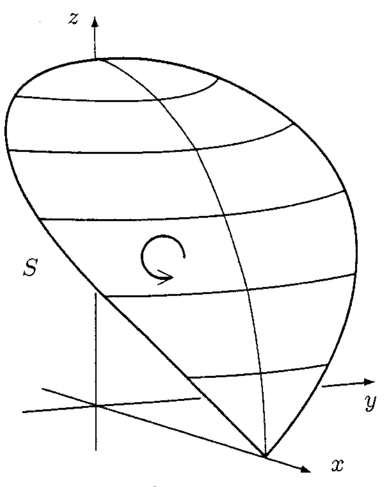


FIGURE 1

At any rate, in the scheme given here, the figure always goes exactly where it was placed in relation to the text. For example, the figure anchored to the left (and it *is* at the left) was set there by the command `\leftpic` (defined below), given at the start of this paragraph. The solution to the first problem is simply to break the page just above the figure, if there really is no room on that page for the figure. The white space this creates can be erased, when revisions are done, simply by moving the figure command a bit.

The TEXbook offers solutions to the second problem in Exercises 14.23 and 14.24. The first idea given there is to combine paragraphs by the side of the figure into a single one, which is shaped to fit the figure and then reformed into the original arrangement of paragraphs. The other idea is to use `\prevgraf` to count the lines that remain to be indented at the start of each new paragraph. However, these approaches are not really suited for use by macros which try to place figures automatically. For one thing, they would require that some unusual instructions be given to start each paragraph, or vertical skip and paragraph, by the side if a figure — you couldn't just leave a blank line or say `\medskip`, etc. Indeed, the first approach would require the whole document initially to be a single paragraph, if there are figures all through it.

But the main difficulty is that the number of lines a figure takes up — that is, the vertical extent of the figure divided by `\baselineskip` (in compatible units) — will not be the number of lines to indent if there are vertical skips between some of the lines, while the solutions to the exercises only take number of lines into account. If for instance, by way of experiment, you say '`\hangindent=1in \hangafter=-3`' at

† Following David Salomon, we use '[xyz]' to refer to page xyz of *The TEXbook*.

the start of a paragraph and put `\vadjust{\vskip5in}` in the second line, the typeset copy will have the first 3 lines indented an inch, but there will be a large gap between the second and third lines. Also, for this reason they don't help to solve the third problem, since as mentioned TeX doesn't directly keep track of the number of lines actually used by a math display. However, rather than counting lines, the macros given here employ `\everypar` to issue instructions at the start of each paragraph, which will freshly shape the paragraph according to where its head happens to lie in relation to the figure (Reid's method does this, too); and math displays are measured correctly by employing `\everydisplay` to arrange matters so that the remainder of the paragraph following the display is treated as a new paragraph and is shaped properly.

2 Summary of commands

Of the several macros described in the next section only four are actually used as commands when one sets figures. They are,

`\leftpic`, `\rightpic`, `\morelines`, `\pixitem...\done`.

The first two of these take as argument the name of a file that contains instructions to draw a figure, and are used to set that figure respectively to the left or right of the surrounding text. `\morelines` is used to change the number of indented lines of text surrounding a figure (this is seldom necessary), while `\pixitem` replaces the `\item` macro of plain TeX in such text.

Definitions and usage of these commands appear below.

3 The macros

To get them going we must allocate registers and set some parameters (see figure 3 below; the "gap" values are of course subject to change as desired):

```
\newcount  \spts    \spts=\baselineskip             % convert <dimen> to <number>
 \newcount \lines   \newcount \moreline  \newcount \pp  \newcount \qq
\newdimen  \breadth  \newdimen \length  \newdimen \cobreadth
 \newdimen \overgap   \overgap=2\baselineskip       % space above the figure
 \newdimen \sidegap   \sidegap=35pt                 % space to the side
 \newdimen \undergap  \undergap=3\baselineskip      % space below, if there's room
 \newdimen \drop      \drop=-1pt
\newbox\picbox    \newbox\bottom    \newskip\prskp  \prskp=\parskip
\newif\ifright  \newif\ifroom  \newif\ifshort  \newif\ifflag  \newif\ifmore
 \newif\ifeqo  \newif\ifleqo
```

Here `\drop` is a quantity which, once the setting of a figure has begun, measures the distance from the top of the page to the bottom of the figure, including space allowance at the bottom; but it needs to be less than zero at the start of a page, as we'll soon see. So it is initialized to −1pt, and is reset to that value page by page, as part of the output routine.

```
\everypar={\checkpic}  \def\emptypar{\everypar={}}  \everydisplay={\adjustdisplaylines}
\output={\fancyoutput}
\def\fancyoutput{                          % When short is true,
  \ifshort\unvbox255\setbox\bottom=\lastbox % look at the bottom line:
    \ifdim\wd\bottom>\cobreadth
      \penalty-201\box\bottom              % if it's too long, break before it.
      \else\ifvoid\bottom\else\restoreleftindent
      \ifdim\wd\bottom<\cobreadth\parskip=0pt\noindent\vadjust{\penalty-1}%
        \ifleqo\global\leqofalse           % Is the bottom line a displayed formula?
        \else\advance\cobreadth-\wd\bottom % if there's no \leqno, restore its indent;
          \ifeqo\global\eqofalse\else\divide\cobreadth2\fi\hskip\cobreadth\fi\fi
      \box\bottom\restorepenalty           % finally put the line back.
    \fi\fi
    \global\holdinginserts=0\global\shortfalse
  \else\plainoutput\global\drop=-1pt\global\onpenalty\fi}  % when short is false
\def\offpenalty{\widowpenalty=1\clubpenalty=2\brokenpenalty=1}
\def\onpenalty{\widowpenalty=150\clubpenalty=150\brokenpenalty=100}
\def\restorepenalty{\ifnum\outputpenalty=10000\else\penalty\outputpenalty\fi}
\def\restoreleftindent{\ifright\else\parskip=0pt\noindent
  \vadjust{\penalty-1}\hskip\breadth\fi}
\let\eqo=\eqno  \let\leqo=\leqno    \def\eqno{\ifshort\global\eqotrue\fi\eqo}
```

```
  \def\leqno{\ifshort\global\leqotrue\fi\leqo}  \def\lzip{\phantom7}
```

`\checkpic` (in `\everypar`) will essentially compare `\drop` and `\pagetotal` at the start of each paragraph, in order to see whether paragraph shaping is needed at that point. If `\pagetotal` ≥ `\drop`, it must be that the head of the paragraph lies below the figure, so shaping is not needed; otherwise, the paragraph shape parameters are set to fit the remainder of the figure. `\pagetotal` is zero when a page begins ([114]), hence if `\drop` < 0 then, a new paragraph at page top will not be shaped to fit some figure which was placed on the previous page. This accounts for some of the last part of `\fancyoutput`; we'll get back to the rest later, but usually it all reduces to `\plainoutput`, unless the figure is being set close to the bottom of the page.

If you are putting other tokens into `\everypar`, then your list should include `\checkpic` when these macros are being used. By the same token you need to keep `\adjustdisplaylines` on your `\everydisplay` list.

3.1 Reading in figures. I use PICTEX to draw my figures: it happens to be just right for my "chalkboard" style diagrams, such as figure 2 at the right, and has the obvious advantages that go with its being part of TEX, while its disadvantages of time and memory requirements are tending to evaporate as equipment improves and titanic TEXs come along; and I put the PICture instructions in a separate file which is then input to the main TEX file. Accordingly, these macros expect to see the name of a file which, when read in by means of `\setbox\picbox=\hbox{\input`⟨file name⟩␣`}`, will cause the figure to be drawn on the page when one says '`\box\picbox`'; or '`\unhbox\picbox`', as given below in `\setpic`, since PICture instructions already define an hbox. (The space after the file name keeps the '`}`' out of the name. `\input` itself does not use curly braces.) Naturally, a different way of producing figures may call for modifications, but the macros as they are will set a figure as long as it is found in an hbox in `\box\picbox`. The perceived size of the figure will be the size of the hbox. One handy feature of PICTEX is that '`\hbox{\input`⟨file name⟩␣`}`' produces a box in which the PICture just fits; if the figure is created differently you may need to specify the box dimensions. (By the way, if you are using PICTEX — these macros intentionally ignore the location of the "reference point" assigned by the `\setcoordinatesystem` command of PICTEX, or at least the location of the point in relation to the PICture as TEX sees it.)

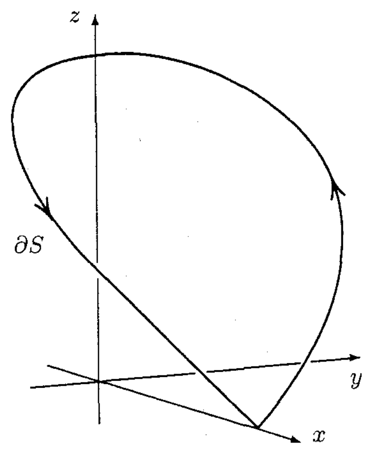


FIGURE 2

To set the figure described by a file named picfoo.tex at the left margin, to the left of surrounding text, you say '`\leftpic{picfoo}`'; on the other hand, '`\rightpic{picfoo}`' will set the figure to the right. For instance, figure 1 above was drawn by the file pictug2.tex and figure 2 was drawn by pictug3.tex, and the figures were inserted by the commands '`\leftpic{pictug2}`' and '`\rightpic{pictug3}`', respectively. The text following the figure command will then be placed around the figure, meaning that a paragraph will be started and shaped to fit the box containing the figure; the command ends the paragraph of text preceding it. A blank line following the figure command makes no difference.

```
\def\leftpic#1{\rightfalse\putpic{#1}}        % file named #1 draws the figure
\def\rightpic#1{\righttrue\putpic{#1}}
\def\putpic#1{\par\previous\stall
  \setbox\picbox=\hbox{\input#1 }             % now \box\picbox holds the figure
  \measure
  \ifroom\setpic
  \else\toss\measure\stall\setpic\fi}
\def\previous{\progress\ifnum\dimen7>0        % take care about the previous figure: if
  \ifshort\shortfalse\toss                    % it's 'short,' go now to the next page;
  \else\kern\dimen7\fi\fi}                    % otherwise leave enough space.
\def\progress{\dimen7=\drop\ifflag\advance\dimen7-\dimen3\flagfalse
  \else\advance\dimen7-\pagetotal\fi}         % set \dimen7 by progress alongside figure
\def\toss{\vfill\eject}
\def\stall{\pp=\pageshrink\divide\pp\spts\advance\pp1 \qq=\pp \parskip=0pt
  \loop\ifnum\pp>0 \line{}\advance\pp-1 \repeat\kern-\qq\baselineskip\penalty0}
```

```
                                   % in case the figure is just past page bottom, or at page top
```

The `\par` which starts `\putpic` is supposed to terminate horizontal mode, thus you should not use these figure commands inside an hbox ([286]). After the figure is packed into `\box\picbox`, the figure box is measured, by `\measure`, and is set on the page by `\setpic`, if there is room for it. But if there happens to be no room on the current page, in the sense of problem (1), the page is ended just above the figure and the figure is set at the top of the next page.

The `\previous` macro has the job of keeping things orderly when there are several figures marching in close succession; `\progress` essentially sets `\dimen7` equal to `\drop` minus `\pagetotal`; `\stalling` is necessary in case a figure command is encountered just past what will be the bottom of a page. We'll get back to those macros eventually.

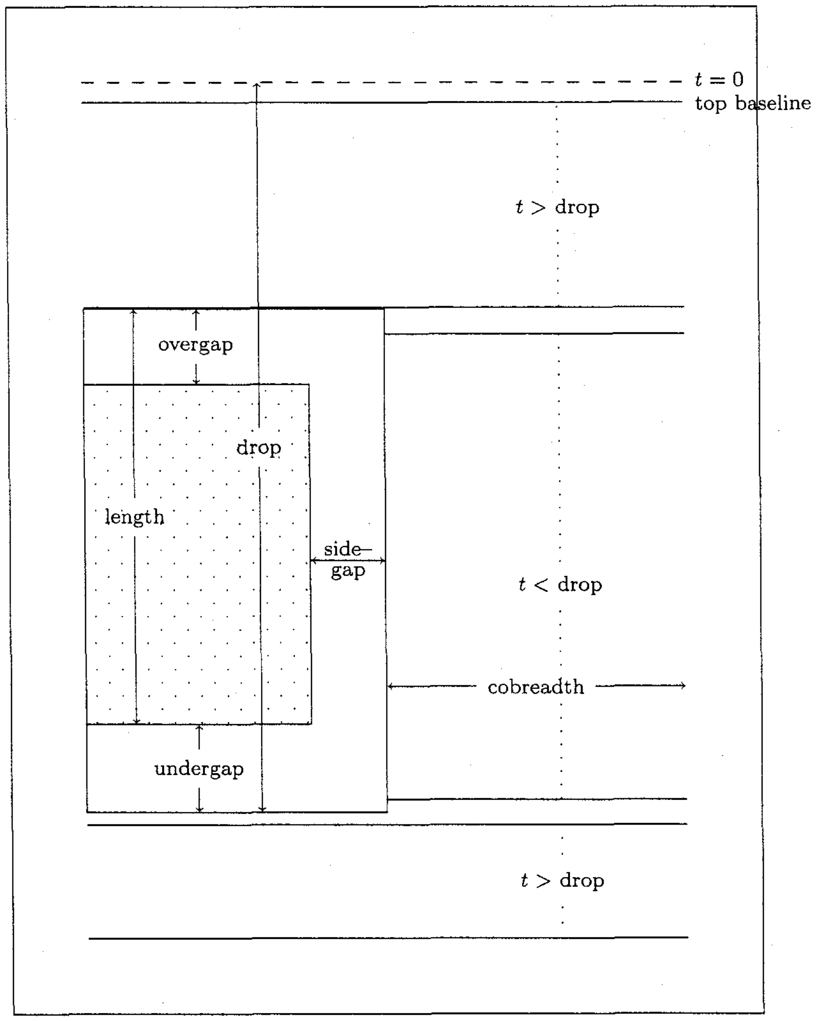


FIGURE 3

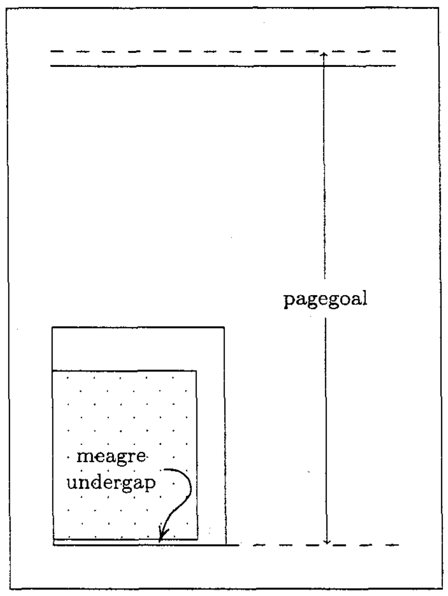
3.2 Measure the figure box. `\measure` is mode-independent; it just measures things, sets `\drop`, etc., and makes some decisions.

```
\def\measure{\flagtrue\dimen3=\pagetotal
  \breadth=\wd\picbox\advance\breadth\sidegap
  \cobreadth=\hsize\advance\cobreadth-\breadth       % correct width of indented text
  \length=\ht\picbox\advance\length\dp\picbox\advance\length\overgap
\drop=\length\advance\drop\dimen3
\ifdim\drop>.99\pagegoal\roomfalse                 % no room on this page
\else\dimen0=\pagegoal\advance\dimen0-\drop        % there is room
 \ifdim\dimen0<\undergap\advance\drop\dimen0       % short space at the bottom
   \shorttrue\offpenalty\holdinginserts1           % [400]
 \else\advance\drop\undergap\shortfalse\fi         % plenty of space at the bottom
\roomtrue\fi}
```

The 'flag' being true will just mean, in the `\progress` macro, that paragraph shaping is going to occur right at the start of a figure, rather than partway down; the current value of `\pagetotal` is saved for later reference. The box containing the figure is measured, and the measurements are then increased, on top by `\overgap` and on the side by `\sidegap`.

If the bottom of that so far augmented box would extend past the nominal page bottom, as determined by the size of `\pagegoal`, we set `\room` false, which will tell `\putpic` to toss the present page and hustle to the next one. Otherwise we allow some space beneath the box: if there is plenty of room we allow `\undergap`, but if doing that would exceed the `\pagegoal` we leave just enough space to reach the nominal page bottom; in this last case we say that the figure, or the gap beneath it, is "short."

In the illustration at the right, the space left beneath the figure box is very narrow since the figure comes nearly to the bottom of the page.



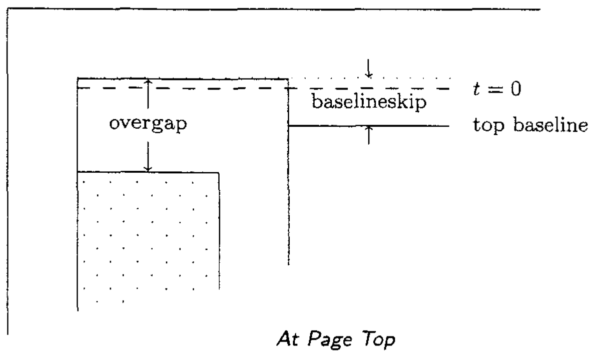
A "Short" Figure

FIGURE 4

3.3 Place the figure. Sooner or later the figure gets set on the page.

```
\def\setpic{\kern\overgap
   % \stall preserves the \kern at page top
  \dimen5=\prevdepth\offinterlineskip
   % save the depth of the previous line
  {\emptypar\parskip=0pt\ifright\hfill\unhbox\picbox % empty \everypar inside the group
   \else\noindent\unhbox\picbox\hfill\fi}%
  \par\kern-\length\penalty10001            % go back up; don't break now!
  \normalbaselines\prevdepth=\dimen5\ignorespaces}
                                            % the next paragraph invokes \checkpic
```

The basic plan here is to place the figure at one margin or the other, then hop back up† and continue with the text following the command. But there is some joinery required. If the figure is at page top, as in figure 5, the first line of indented text should be at the normal position for the first line on a page. Otherwise, the first indented line should be the correct distance below the line of text just preceding the figure. And, in both cases the top of the figure should be in the same location relative to the first line of indented text.



At Page Top

FIGURE 5

Well — actually, `\setpic` does the first two things, and ordinarily it does the third, but a first indented line containing a tall character will be lower than ordinary in relation to the figure. In fact, the top of the figure box is placed the distance `\overgap` below the baseline of the box of text just above the figure; if the figure is at page top, an empty `\line{}`, one baselineskip higher than normal, is substituted for the box

† I am obliged to Karl Berry for suggesting this idea.

above. That line, which keeps TeX from discarding the \overgap kern at the top of the page, will be set there by \stall, which we'll get to later.

TeX will want to start a new paragraph when it has to set the figure box, so the effects, vertical and otherwise of its doing so are temporarily annulled (which means the reference point of a PiCture gets ignored, as mentioned above). When the figure box is in place and we are back up again, the value of \prevdepth is restored so that the first indented line will be set properly.

3.4 Shaping. But how do the lines get indented. We exit \setpic, hence the figure command, in vertical mode. Recall that \everpar contains \checkpic:

```
\def\checkpic{\progress\ifnum\dimen7>0 \shapepar\fi\parskip=\prskp}
\def\shapepar{\lines=\dimen7                % convert <dimen> to <number>
  \divide\lines\spts
  \ifnum\lines>0
   \ifright\hangindent=-\breadth\else\hangindent=\breadth\fi
  \hangafter=-\lines
   \ifmore\advance\hangafter-\moreline\morefalse\fi\fi}
```

and \checkpic will if necessary summon \shapepar to shape the lines in the paragraph following the figure command, that is unless \pagetotal $\geq$ \drop as we said earlier. When the figure is being set on the page, the value of \pagetotal is changed a little by \setpic, and the amount of change depends on whether the figure is at page top or not; but this change is ignored by \progress, hence by \checkpic and \shapepar, instead they all use the original value of \pagetotal saved as \dimen3. Once the figure is in place, \flag is turned off and they just use the current \pagetotal value. Then, the amount of indentation and the number of lines to be indented are reckoned, according to current conditions. (We have to exclude the case that \lines=0, since TeX will indent the entire paragraph if \hangafter=0.)

It may sometimes happen that a paragraph by the side of a figure begins with text which is inside a group, for instance so as to be slanted when typeset. \checkpic will be inserted at the beginning of the paragraph, but then TeX will forget that the paragraph was shaped to fit the figure once the group ends, so in that case you need to say \checkpic following the group. For instance, the fourth manuscript paragraph of this article begins with the *TUGboat* macro \TB, which sets 'The TeXbook' in slanted type, and continues with '\/\checkpic\ offers ...'

3.5 Ending math displays; getting more lines. The last line of \shapepar comes into use if you want to change the number of indented lines next to the figure, perhaps to accommodate a math display that ends or begins near the bottom of the figure, or to compensate for vertical glue directly following a figure command (where \checkpic will not see it).

```
\def\morelines{\global\moretrue\global\moreline}      % put _before_ a closing $$
\def\adjustdisplaylines#1$${#1$$\par\vskip-\parskip\noindent\ignorespaces}
                                                % the next paragraph invokes \checkpic
```

To get one more indented line in the last of the indented paragraphs next to a figure, you say '\morelines1', or '\morelines=1', just *before* the start of that paragraph, that is before the \everypar tokens for the paragraph have been inserted; '\morelines-2' will give you two fewer indented lines, etc.

\everydisplay contains \adjustdisplaylines; this macro takes the displayed math material as argument for its delimited parameter, the delimiter being the $$ which closes the math display, and the macro repeats the displayed material and the closing $$. Then the text immediately following the display (if there is any) is put into a new paragraph, unparindented and unparskipped but shaped properly by \checkpic and \shapepar. Thus \morelines, if it is to be used to alter the number of lines indented in that paragraph, needs to go in *before* the closing $$ (hence the '\globals' in its definition), since the paragraph shape parameters will already have been set when material after the closing $$ is being read.

This use of \everydisplay follows a suggestion of Stephan von Bechtolsheim. It is convenient in that displays are enclosed by pairs of $$s in customary fashion, but because the displayed material is treated as a macro argument, occasional problems can arise. For instance, if you use a construction such as $$\vbox{\settabs\+⟨sample line⟩\cr...}$$, TeX will complain because \+ is \outer; replacing \+ with \tabalign doesn't fix things because the \settabs will then expect to see a number of columns ([355]). One way out is locally to redefine \+ as \tabalign, thus erasing its \outerness. Another approach is to define a replacement for the closing $$ of a display, say by \def\enddisplay{$$\par\vskip-\parskip\noindent \ignorespaces}; then the displayed material will not be a macro argument. However, in that case you have to say `\endisplay` to close a display (instead of `$$`) if you want the text immediately following it to be shaped to fit some recent figure.

If the text to be indented by the side of the figure happens to contain a `\vbox` or a `\vtop` in which the `\hsize` has been lessened, you should probably put `\emptypar` (defined as `\everypar={}`) at the start of the material in the box; otherwise TeX will try to leave room for the figure inside the box. The effect of `\emptypar` is confined to the group in the box. For instance, in the following sample, `$$\eqalign{S_j&=\vtop{\emptypar\hsize=130pt` the $(n-1)$-box...`}\cr...}$$` was used next to the figure.

... Let $S=\partial K$. S consists of n pairs of facing sides. Let S_j, S'_j denote the j^{th} pair, where

$$\begin{aligned} S_j &= \text{the } n-1\text{-box which is spanned at } P \text{ by all the } \mathbf{v}\text{'s } \textit{except } v_j;\\ S'_j &= \text{the parallel translate of } S_j \text{ to the head } Q_j \text{ of } v_j. \end{aligned}$$

S_j and S'_j are $n-1$-boxes, and are oriented as part of the boundary of the n-box K. S_j is spanned at P by v_i's as we indicated, but their increasing order may not be the positive order on S_j; likewise S'_j is spanned by the parallel translates of the v_i's to Q_j, though not necessarily in that order ...

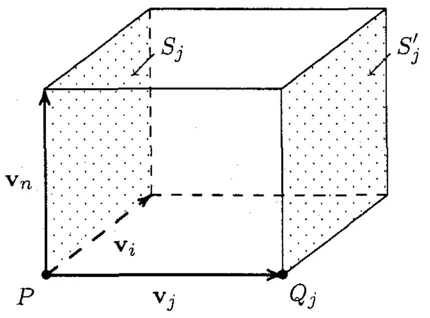


FIGURE 6

3.6 The output routine. About that `\output` routine. It went like so:

```
\def\fancyoutput{                              % When short is true,
 \ifshort\unvbox255\setbox\bottom=\lastbox % look at the bottom line:
   \ifdim\wd\bottom>\cobreadth
     \penalty-201\box\bottom                   % if it's too long, break before it.
     \else\ifvoid\bottom\else\restoreleftindent
     \ifdim\wd\bottom<\cobreadth\parskip=0pt\noindent\vadjust{\penalty-1}%
       \ifleqo\global\leqofalse                % Is the bottom line a displayed formula?
       \else\advance\cobreadth-\wd\bottom      % if there's no \leqno, restore its indent;
         \ifeqo\global\eqofalse\else\divide\cobreadth2\fi\hskip\cobreadth\fi\fi
     \box\bottom\restorepenalty                % finally put the line back.
   \fi\fi
   \global\holdinginserts=0\global\shortfalse
 \else\plainoutput\global\drop=-1pt\global\onpenalty\fi}  % when short is false
\def\offpenalty{\widowpenalty=1\clubpenalty=2\brokenpenalty=1}
\def\onpenalty{\widowpenalty=150\clubpenalty=150\brokenpenalty=100}
\def\restorepenalty{\ifnum\outputpenalty=10000\else\penalty\outputpenalty\fi}
\def\restoreleftindent{\ifright\else\parskip=0pt\noindent
 \vadjust{\penalty-1}\hskip\breadth\fi}
\let\eqo=\eqno  \let\leqo=\leqno    \def\eqno{\ifshort\global\eqotrue\fi\eqo}
\def\leqno{\ifshort\global\leqotrue\fi\leqo}  \def\lzip{\phantom7}
```

If `\short` is false then this is just `\plainoutput`, along with the resetting of `\drop`. Otherwise— Recall that `\short` is set true by `\measure` when the distance from the bottom of the figure box to the nominal pagebottom, as given by the value of `\pagegoal`, is positive but less than `\undergap`. In that case the number of indented lines is based on the location of the nominal pagebottom, and the page break is expected to occur at or just before that point. But the page builder might decide to break the page a line later than expected, with the result that the last line on the page will be full width, or at least wider than `\cobreadth` if it is a wide displayed formula; it will not be indented and will probably run over part of the figure, or its caption. This will not do, so we have to persuade the page builder to make the right page break. As `\short` is true, the `\output` routine will on its first pass look at what TeX thinks the last line should be. If in fact that line is wider than `\cobreadth`, a negative penalty is inserted before it, to encourage the desired breaking at the penalty item, that is just before the line; otherwise the line is put back where it was, re-indented as necessary (see below); then the page is reprocessed.

More exactly, in the absence of penalties, the cost c of a page break is (according to the page-breaking algorithm, [111]) just the badness b when $b<10000$, that is when the `\pagetotal` t is near the `\pagegoal` g.

Hence c is a decreasing function of t when t is close to but less than g, and it is an increasing function when t is close to but greater than g. (Roughly, $c \approx |t - g|^3$ times a factor which depends inversely on the amount by which the page can stretch (if $t < g$) or shrink (if $t > g$), [77, 97, 111].) Of the two least values of c for t on either side of g, sometimes one will be less, sometimes the other; but each of them will be less than all the other costs on its side, so one of those two corresponding lines will be selected as the place to break. Only the higher line on the page, that is the line with $t < g$, will be indented (by `\shapepar`), the next line will be full length. When the cost of breaking after the lower line would in fact be less, a penalty is inserted by the `\output` routine to discourage that break, as we said. (The `\penalty-201` used here is sufficient for pages of ordinary length, but a more negative penalty would be necessary for pages of small `\vsize`, since when b changes on such pages it tends to do so very suddenly.)

If, however, there were club, widow or broken penalties attached to some of the lines around the breakpoint, all this reasoning would be susceptible of failure, so it is necessary to suppress those penalties for the time being; this is done by placing `\offpenalty` in `\measure`, in the part where `\short` is set true. (The penalties are here set to 1 or 2 instead of 0 just so that they will continue to show up in diagnostic output from `\tracing`things. You may prefer that they = 0.)

Of course, penalties which are suppressed will be unable to give the protection they were designed for, and you may get club lines, etc., in consequence. Furthermore, other penalties which affect page breaking remain unsuppressed here. For instance, a math display near the bottom of a figure near the bottom of a page may drag the short line preceeding it, along with it to the top of the next page (and perhaps cause an underful page), because of the `\predisplaypenalty`. Or the text may end with a `\bye` just past a page that contains a figure at the bottom and some shrinkable glue, such as appears when there are displayed equations ([189], [348]); then TEX will get to read the `\bye`, and the `\supereject` penalty issued by the `\bye` may force the last lines of text to be part of that page, so that they run over the lower portion of the figure.

But in such cases it is probably better to fix by rewriting.

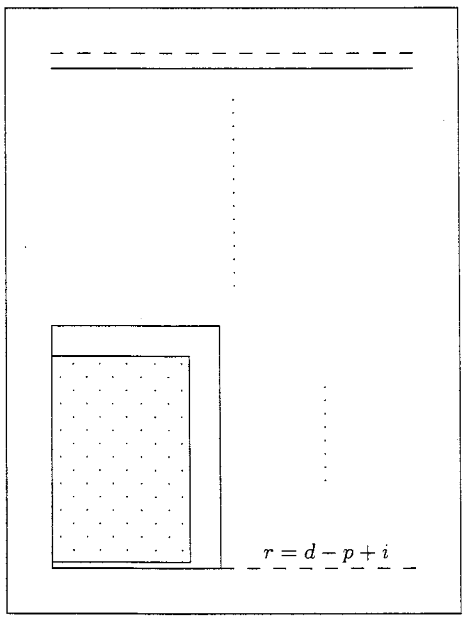


FIGURE 7

When the higher line is going to be chosen for the last line on the page, we want to put the contents of `\box255` back onto the main vertical list so as to achieve a "seamless join between the completed page and the subsequent material" ([254]), thus if the breakpoint is a penalty item we should reinsert it; and in any case we shall have to hold insertions in place during the first pass through `\output` (see [125] and [254]). In TEX 3, `\outputpenalty=10000` unless the breakpoint is a penalty item ([125]), so we only restore it in the latter case. Insertions are held in place by setting `\holdinginserts=1` in `\measure`, in the part where `\short` is set true, then they are released in time for the second pass. The point of holding them is to be sure that TEX leaves enough room during the second pass, otherwise `\pagegoal` will be reset to full size and the insertions will be crammed onto an already full page, with an overfull page as the distressing result.

Sometimes the last line on a page has to be re-indented. Assume that `\short` is true and the length of the last line is less than `\cobreadth`. The case that `\lastbox` be void is excluded,† so on its first pass the `\output` routine will set `\box\bottom` to an hbox containing the line, and will then put that hbox back onto the main vertical list, where the last line was. But any instruction there may originally have been to shift the line will be lost in this process; it will not be saved in `\box\bottom`. On its second pass, the output

† `\lastbox` will be void for instance on the final page of text, when the last item on the vertical list is the `\supereject` penalty issued by a `\bye`.

routine is `\plainoutout` and, if we do nothing, it will place that line unshifted, flush left; so we may be obliged to re-indent the line by ourselves. In fact, this will be necessary if `\leftpic` is being used, or if the last line is a displayed formula, as in figure 7. In the first case, we insert, with `\vadjust`, a small penalty item following the shifted line, so as to permit the page break after the line. In the second case the correct indentation will depend on whether there are equation numbers or not, according to the rules for positioning equation numbers ([188–189]). Also, in that case there is a complication: a displayed formula with a restored indent is followed on the vertical list by an infinite penalty, which prevents the desired page break. To remedy this we again insert an ameliorating penalty item following the line, so as to permit the break after all.

It could happen that `\short` is true and there are several displayed formulas next to the figure, some of which have equation numbers put there by `\eqno` or by `\leqno` (we assume that the numbers are either all at the right or all at the left), and some with no numbers. In that case, for the sake of correct indentation, it is necessary to supply bogus equation numbers to formulas without real ones. You can do this by typing '`$$`⟨formula⟩`\eqno$$`' or '`$$`⟨formula⟩`\leqno\lzip$$`' for those formulas, depending on the circumstances.

There is one possibility which is overlooked by our re-indenting scheme, namely that in which the last line is a displayed formula with an equation number, and its actual width is less than `\cobreadth`, but the line is considered to be below the figure, not an indented line. Then it belongs on the next page but it will not be put there by `\fancyoutput`, and the spacing for the equation number will be too wide. However, an application of `\morelines` is an effective, if not an automatic, fix for this.

3.7 The previous figure. When there are more figures than text, the natural separation by prose may not keep one figure from treading on another, so we must make arrangements to avoid overlaps. The `\previous` macro does the job:

```
\def\previous{\progress\ifnum\dimen7>0      % take care about the previous figure: if
  \ifshort\shortfalse\toss                  % it's 'short,' go now to the next page;
  \else\kern\dimen7\fi\fi}                  % otherwise leave enough space.
```

Recall that `\dimen7` is set by `\progress`, essentially to `\drop` − `\pagetotal`. If `\dimen7` > 0 when a figure command is read, text is still being set by the side of the previous figure. If the previous figure on this page was short, we certainly don't want to try to fit the current figure on the same page, so it is tossed to the next page. As the last line will then be empty there is no need to take a close look at it, thus `\short` is set false. But ordinarily a figure is not short, and when it is not, at least the `\undergap` space is kept between its bottom and the top of the `\overgap` space above the next figure (as illustrated just below).

If you want to bring successive figures closer together than the current `\undergap` + `\overgap`, the way to do it is to decrease the size of `\undergap`, before the start of the upper figure, or of `\overgap`, before the lower figure. A negative `\vskip` just before the second figure command will be ignored, since `\progress` will take the skip into account when it computes `\dimen7`, and leave that much more vertical space.

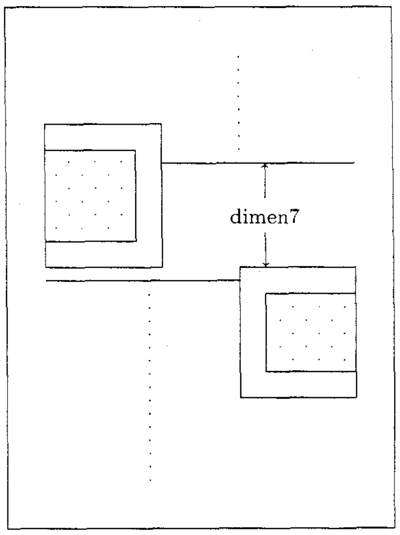


FIGURE 8

3.8 Stalling. Why is it we have to `\stall`, at the beginning of `\putpic`?

```
\def\stall{\pp=\pageshrink\divide\pp\spts
  \advance\pp1 \qq=\pp \parskip=0pt
  \loop\ifnum\pp>0 \line{}\advance\pp-1
  \repeat\kern-\qq\baselineskip\penalty0}
```

This maneuver becomes necessary when a figure command turns up just past what will be the eventual breakpoint on a page that contains some shrinkable glue. In that case, the page builder has TeX continue to read lines until it finds one of infinite cost, but if there is shrinkable glue that could well be several lines past the best breakpoint; and by then TeX may have encountered and begun reading the figure command. Big mess! With `\pagetotal` > `\pagegoal` by that point, `\measure` will find no room at all on the current page for the figure, so `\putpic` will issue

`\vfill\eject`, meaning to toss the figure to the next page; but by the time everything is sorted out there may be a very short page where the figure was supposed to go, with the figure itself on the page after that. Under such circumstances we must stall, and issue a series of junk `\line{}`s to prevent TEX's continuing to read the command, until the page builder has caught up in its exercise and decided on the breakpoint for the page, and has initiated the `\output` routine. The number of junk lines issued depends on the amount of shrinkable glue on the page. `\stall` ends with `\penalty0` so that the effect of the negative `\kern` which cancels the junk lines may be recorded in `\pagetotal`.

The number of junk lines issued is always at least one. This conveniently serves a different purpose than that just mentioned, namely to set a figure correctly when it has to go at page top: the junk line preceeding the figure saves the `\overgap` kern at the start of `\setpic` from being discarded in that case.

Of course, `\stalling` is superfluous for most figures; but in a document of any length, and subject to revision, you never know when you'll need it.

3.9 One more item. Plain TEX provides the macro `\item` ([102], [355]), which allows one to typeset numbered, indented items. This macro can be useful in connection with figures, for instance if the figure illustrates several notions which you want to itemize by its side; but `\item` achieves hanging indentation by setting `\hangindent=\parindent`, so we cannot use it in combination with our figure commands since they need `\hangindent` themselves. However, we can get the same result using `\leftskip`.

```
\def\pixitem#1 {\par\removelastskip\smallskip\noindent\llap{#1\enspace}%
  \ignorespaces\leftskip=0.75\parindent\begingroup\parindent=0.5\parindent}
\def\done{\par\endgroup\par\smallskip\leftskip=0pt}
       % use like \item; end with \done
```

The parameter has the same usage as `\item`'s parameter, except it is delimited by a space. You can have more than one paragraph in each item, but to finish each item you need to say '`\done`'. The indentation is diminished since there will be less room than normal next to the figure. Items are set off by `\smallskips`, fore and aft.

The pair of `\pars` surrounding the `\endgroup` in the definition of `\done` is necessary because of the grouping ([100]). The first `\par` allows the final paragraph inside the group to be shaped properly, the second does the same for the paragraph following the group. Without the first `\par`, the group would end before its final inside paragraph, and that paragraph would be shaped according to the values that `\hangindent` and `\hangafter` had at the *start* of the group, but with `\leftskip=0pt`. With the first `\par` but without the second, the final inside paragraph would have `\leftskip=0.75\parindent`, but now the text following the group would be shaped according to the values that `\hangindent` and `\hangafter` had in the final inside paragraph.

◇ Daniel Comenetz
46 Burnham Street
Belmont, MA 02178