Module-III

Tables and Figures

A table is used for presenting data or items row- and column-wise in a concise form. In LaTeX, the **tabular**, **tabularx**, and **longtable** environments are used for preparing different types of tables. However, tables produced by the **tabular** and **tabularx** environments cannot be assigned any serial number or title, which are generally required to identify a table. Moreover, they produce a table as a single object in running texts, which may cause a problem in drawing a big table in the limited space of a page. All such drawbacks can be overcome by nesting the **tabular** and **tabularx** environments with another environment, such as the **table**, **wraptable**, or **sidewaystable** environment (a serial number and a title to the table under the **longtable** environment can be assigned directly).

7.1 Table Through the tabular Environment

Tables are widely prepared through the **tabular** environment, where the columns of a table are generated through the mandatory argument of the environment. For example, **\begin{tabular}{|l|c|c|c|c|}** in Table 7.1 generates a five-column table

Table 7.1 A simple table through the **tabular** environment

LaTeX input

```
\begin{table}[!hbt]
\centering
\caption{Obtained marks.}
\label{tab-marks}
\begin{tabular}{|l|c|c|c|c|}
\hline Name   & Math & Phy & Chem & English\\
\hline Robin  & 80   & 68  & 60   & 57\\
\hline Julie  & 72   & 62  & 66   & 63\\
\hline Robert & 75   & 70  & 71   & 69\\
\hline
\end{tabular}
\end{table}
%
Table~\ref{tab-marks} shows the ...
```

Output

Table 1: Obtained marks.

Name	Math	Phy	Chem	English
Robin	80	68	60	57
Julie	72	62	66	63
Robert	75	70	71	69

Table 1 shows the marks obtained by three students in the final examination.

D. Datta, *LaTeX in 24 Hours: A Practical Guide for Scientific Writing*,
DOI 10.1007/978-3-319-47831-9_7

(**\begin{tabular}[]{}** may also be used with optional provision in **[]** for vertical positioning). A column can be generated through one of the three letters of **l**, **r**, and **c** (other types of columns are discussed in §7.2 and §7.5). Each of these letters represents a column as well as the alignment of the entries in that column (**l** for left alignment, **r** for right alignment, and **c** for center alignment). The **|** symbol in the argument of **\begin{tabular}{}** is used either to mark a boundary or to separate two columns by a vertical line in the specified location, covering the full height of the table. Following the **\begin{tabular}{}** command, the column-wise entries of a row are inserted, separating two entries by an **&** and ending the row by a line break command ****. Further, the **\hline** command is used either to mark a boundary or to separate two rows by a horizontal line in the specified location, covering the full width of the table (a **\hline** command before a row draws a horizontal line above the row). Finally, the **tabular** environment is ended by the **\end{tabular}** command (the **\hline** command just above the **\end{tabular}** command draws the lower horizontal boundary of the table).

Note that the **tabular** environment in Table 7.1 is nested inside the **table** environment for creating the table in a separate paragraph as well as for captioning and labeling it. The **table** environment is first created through the **\begin{table}[!hbt]** command (the optional argument **!hbt** is for the preferred vertical positioning of the table, which is explained in §7.3). The next command in Table 7.1 is **\centering**, which instructs for width-wise center alignment of the table (other commands could be **\flushleft** for left alignment or **\flushright** for right alignment). The **\caption{**`attl`**}**[1] command used in the **table** environment (but outside the **tabular** environment) assigns a serial number to the table preceded by the default label-word `Table` and followed by a colon, along with its argument `attl` as the title (caption) of the table (since the title usually comes on the top of a table, the **\caption{}** command is used before the **tabular** environment). Following the **\caption{}** command, the **\label{}** command is inserted with a unique reference key, which as shown in Table 7.1 can be used in the **\ref{}** command for referring the table anywhere in the document. Also note that **\label{}** is always used after **\caption{}**. Moreover, **\label{}** does not have any effect without **\caption{}**, in which case the table is not assigned any serial number.

7.2 Table Through the tabularx Environment

In the **tabular** environment discussed in §7.1, a column is generated by one of the options of **l**, **c**, and **r**. The width of a column under any of these options is made equal to the length of the longest entry in that column. This may extend a table even beyond the width of a page if the table has some very long entries.

The **tabularx** package provides the **tabularx** environment, which can calculate automatically the width of a column so as to restrict a table within

[1] If the **\caption{**`attl`**}** command is used inside the **table**, **longtable**, or **sidewaystable** environment (but outside the **tabular** or **tabularx** environment), the table is assigned a serial number along with the argument `attl` as the title (caption) of the table.

a pre-specified horizontal width irrespective of the lengths of the entries in the table. The **tabularx** environment takes two mandatory arguments, i.e., **\begin{tabularx}{**awidth**}{** acols**}**, where awidth is the horizontal width of the table and acols is its columns. The columns in the **tabularx** environment are generated in the same way as in the **tabular** environment. A fixed-width column is generated through **l**, **c**, or **r**, while a **X** is used to generate a flexible-width column (i.e., a column whose width is to be calculated automatically)[2]. All the flexible-width columns of a table are of equal width, which is calculated internally as the difference of the total width (awidth) of the table and total width of the fixed-width columns, divided by the number of flexible-width columns. Entries in a flexible-width column are made full aligned. Other alignments can be obtained using either **>{\raggedright\arraybackslash}**, **>{\centering\arraybackslash}**, or **>{\raggedleft\arraybackslash}** before **X**, which make the entries left, center, and right aligned, respectively (without the **\arraybackslash** command, the line breaking command **** used for terminating a row may not work properly in some cases). Table 7.2 shows an application of the **tabularx** environment

Table 7.2 A simple table through the **tabularx** environment

LaTeX input

```
\begin{table}[!hbt]
\centering
\caption{Scored points.}
\begin{tabularx}{0.8\linewidth}
        {|X|c|>{\raggedleft\arraybackslash}X|}
 \hline  {\bf Name}  &  {\bf Sex}  &  {\bf Points}\\
 \hline  Milan       &     M       &        1,500\\
         Julie       &     F       &        1,325\\
         Sekhar      &     M       &          922\\
         Dipen       &     M       &          598\\
         Rubi        &     F       &           99\\
 \hline
\end{tabularx}
\end{table}
```

Output

Table 2: Scored points.

Name	Sex	Points
Milan	M	1,500
Julie	F	1,325
Sekhar	M	922
Dipen	M	598
Rubi	F	99

for generating a three-column table of a total width of 80% of the page width, i.e., 0.8**\linewidth** (a fixed value, say 10cm or 6in, can also be used). Since the middle column is generated by the option **c**, its width is fixed by the longest entry in that column. The extreme two columns are generated by the option **X**, for which their widths are equal and calculated internally to accommodate all the three columns in

[2] In the **tabularx** environment, a fixed-width column is generated through **l**, **c**, or **r**, while a flexible-width column is generated through a **X**.

the pre-specified width (i.e., `0.8\linewidth`) of the table. Moreover, the last column is made right aligned by generating it through `>{\raggedleft\arraybackslash}X`, instead of just through `X`. All other matters of Table 7.2 are same with those of Table 7.1.

7.3 Vertical Positioning of Tables

As shown in Tables 7.1 and 7.2, the preferred vertical position of a table on a page can be specified as an optional argument to the **table** environment, i.e., `\begin{table}[avp]`, where `avp` is the specifier for vertical positioning of the table. The commonly used specifiers are **h**, **b**, and **t**, which stand for here, bottom of the page, and top of the page, respectively. These specifiers can be used individually or in a combination of two or three. Moreover, for placing the table in the specified position even if enough space is not available on the current page, the specifier or the combination of the specifiers may be preceded by a **!** symbol, like **!h**, **!b**, or **!hbt**. Irrespective to the order of the specifiers in a combination, LaTeX always uses the following order for positioning a table:

- ▷ If **!** is used, many default or preset restrictions are ignored and a table is attempted to put in the specified position.
- ▷ If **h** is given, the table is attempted to put in the exact position. If fails and no more specifier is given, by default LaTeX considers the specifier **t** for placing the table on the top of the next page.
- ▷ If **t** is given, the table is attempted to put on the top of the current page.
- ▷ If **b** is given, the table is attempted to put at the bottom of the current page.

Besides **h**, **b**, and **t**, there is another specifier **H**, which is defined in the **float** package. Usually, if a table cannot be put on the current page due to space limit, it is taken to the next page and the remaining space of the current page is filled by the texts which are typed in the LaTeX input file after the table. However, **H** instructs to put a table *here* only. If the blank space on the current page is not sufficient to hold the table, it is taken to the top of the next page along with the texts that follow the table, by leaving the current page incomplete. The specifier **H** is used alone, i.e., it should not be combined with **!** or any of **h**, **b**, and **t**. Refer §8.9 on the page 80 for putting all the tables at the end of a document, regardless of their actual positions in the LaTeX input file.

7.4 Sideways (Rotated) Texts in Tables*

If a table contains some long entries, space can be saved by printing such entries in vertical direction through the **sideways** environment defined in the **rotating** package. An application of the **sideways** environment is shown in Table 7.3 on the next page.

Table 7.3 Table with entries in vertical direction

LaTeX input

```
\begin{tabular}{|l|c|c|c|}
\hline Name& \begin{sideways}Mathematics\,\end{sideways}&
           \begin{sideways}Physics\end{sideways}&
           \begin{sideways}Chemistry\end{sideways}\\
 \hline Robin  & 80 & 68 & 60\\
 \hline Julie  & 72 & 62 & 66\\
 \hline Robert & 75 & 70 & 71\\
 \hline
\end{tabular}
```

Output

Name	Mathematics	Physics	Chemistry
Robin	80	68	60
Julie	72	62	66
Robert	75	70	71

7.5 Adjusting Column Width in Tables*

The width of a column, generated through **l**, **r**, or **c**, is set automatically based on the length of the longest entry in that column. This may suffer from the drawback of extending a table beyond the page width if the table has some long entries (many users tackle the situation by manually splitting a long entry into multiple rows). On the other hand, the **X** option in the **tabularx** environment generates columns of equal width irrespective of the lengths of their entries. This may also suffer from the disadvantage of allocating excess width to columns having short entries only, while some columns not having sufficient width to accommodate their long entries.

Above drawbacks can be alleviated by generating columns of a table through **p{}**, **m{}**, or **b{}** command defined in the **array** package. The arguments of **p{}**, **m{}**, and **b{}** specify the width of a column, and the letters **p**, **m**, and **b** make an entry, respectively, vertically top aligned, middle aligned, and bottom aligned relative to the alignment in the previous column (entries in all the three cases are horizontally full aligned). Applications of these three options for fixing the width of a column are shown in Table 7.4 on the next page. The same vertical alignment is made to all the three columns in the first three cases. While in the fourth case, three different alignments are made to the three columns of the table. The first column in the fourth case is top aligned (**p{}**), middle one is middle aligned (**m{}**), and the last one is bottom aligned (**b{}**). As a result, the vertical alignment of a column has become relative to that of its previous column. The second column is middle aligned about the top line of the first column which is top aligned. Similarly, the middle of the second column is made the bottom of the third column which is bottom aligned.

Fixing the width of a column by an absolute value, like **p{**`1.5`**cm}**, may make a table too small or extending beyond the width of the page (or the column of a multi-column document), particularly if the page or font size is changed in a later stage. Therefore, a good practice would be to fix the width of a column as a fraction of the **\linewidth** command in a single-column document and **\columnwidth** command in a multi-column document[3], e.g., **p{**`0.3`**\linewidth}** or **m{**`0.2`**\columnwidth}**.

[3] A good practice would be to specify a length as a fraction of **\linewidth** in a single-column document and **\columnwidth** in a multi-column document (instead of a fixed length, like `5`**mm**), particularly to avoid any unpleasant output upon changing the page or font size in a later stage.

Table 7.4 Fixing column widths in tables with p{}, m{}, and b{}

LaTeX input	Output

`\begin{tabular}{|p{1.7cm}|p{1.5cm}|p{1.6cm}|}`
`\hline This is the first and the ...&`
`A medium size entry&`
`This is another long entry\\`
`\hline`
`\end{tabular}`
	This is the first and the longest entry | A medium size entry | This is another long entry

`\begin{tabular}{|m{1.7cm}|m{1.5cm}|m{1.6cm}|}`
`\hline This is the first and the ...&`
`A medium size entry&`
`This is another long entry\\`
`\hline`
`\end{tabular}`
	This is the first and the longest entry | A medium size entry | This is another long entry

`\begin{tabular}{|b{1.7cm}|b{1.5cm}|b{1.6cm}|}`
`\hline This is the first and the ...&`
`A medium size entry&`
`This is another long entry\\`
`\hline`
`\end{tabular}`
	This is the first and the longest entry | A medium size entry | This is another long entry

`\begin{tabular}{|p{1.7cm}|m{1.5cm}|b{1.6cm}|}`
`\hline This is the first and the ...&`
`A medium size entry&`
`This is another long entry\\`
`\hline`
`\end{tabular}`
	This is the first and the longest entry | A medium size entry | This is another long entry

In the columns of a table, entries are printed leaving some blank space on both sides defined by the \tabcolsep command. The length of such a horizontal blank space between two columns can be changed by changing the value of \tabcolsep (default is 6 pt), e.g., \setlength{\tabcolsep}{2mm}. Similarly, the blank space before or after a particular entry can be eliminated using @{}, e.g., \begin{tabular}{|@{}l|l|@{}} will omit blank space on either side of a table, and \begin{tabular}{|l@{}|@{}l|} will omit the blank space between the two columns (@{~} can also be used for leaving a blank space of length equal to that of ~). On the other hand, the indentation of an entry can be increased by redefining the length of the \parindent command (default is 0 pt), e.g., >{\setlength{\parindent}{5mm}}p{} will generate a column, in which entries will be indented by 5 mm.

7.6 Additional Provisions for Customizing Columns of Tables*

Besides the provisions discussed in §7.1–7.5, the tabular and tabularx environments have many more provisions for customizing a table, some of which are outlined here (all of these provisions are defined in the array package).

▷ The style of the entries in a particular column can be altered using >{command} before the column-generating option l, c, r, X, p{}, m{}, or b{}. For example, >{\bfseries}l for printing all the entries of that column in boldface fonts, or >{\centering}p{5cm} for making the entries center aligned.

- ▷ A column-generating option can be preceded and followed by `>{$}` and `<{$}`, respectively, for converting the column into math-mode, e.g., `>{$}l<{$}` will generate a left-aligned math-mode column so that a mathematical expression can be inserted in that column without creating any more math-mode.
- ▷ For repeated use of a particular type of column, a new column type can be defined in the preamble through the `\newcolumntype{}{}` command. For example, `\newcolumntype{C}{>{$}c<{$}}` can be used for generating directly a center-aligned math-mode column with `C`, or `\newcolumntype{R}{>{\raggedleft\arraybackslash}X}` for generating a right-aligned flexible-width column with `R`.
- ▷ Instead of repeating a column type for generating multiple number of consecutive columns of the same type, `*{n}{ctype}` may be used, which means `n` number of columns of type `ctype`. For example, `\begin{tabular}{|l|*{5}{c|}}` will generate a left-aligned column first and then five number of center-aligned columns, with vertical lines on both sides of each column (Table 7.6 on page 67 shows an application).
- ▷ For changing the width of a column-separating vertical line (default width is `0.4pt`), the `|` sign may be replaced by `!{\vrule width aval}` with `aval` as the width of the vertical line, e.g., `!{\vrule width 0.9mm}` will generate a vertical line of 0.9 mm width.
- ▷ The width of vertical and horizontal lines created by `|`, `\vline`[4], `\hline` or `\cline{}` can be controlled by setting the value of the `\arrayrulewidth` command (default value is 0.4 pt), e.g., `\setlength{\arrayrulewidth}{2pt}` for obtaining 2 pt thick lines.
- ▷ The `booktabs` package provides some commands for drawing horizontal lines of different widths, as well as of different spacings below or above a horizontal line. These commands include `\toprule[]`, `\midrule[]`, `\bottomrule[]`, and `\addlinespace[]`, where the width or spacing value, as applicable, is taken as the argument in `[]`. For example, `\toprule[3pt]`, `\midrule[1pt]`, and `\bottomrule[2pt]` (instead of `\hline`) for producing the top, middle, and bottom lines of a table of widths 3 pt, 1 pt, and 2 pt, respectively. On the other hand, `\toprule[3pt]\addlinespace[2pt]` for leaving 2 pt blank space below the top line, or `\addlinespace[1pt]\bottomrule[2pt]` for a blank space of 1 pt above the bottom line.
- ▷ The vertical space between a column entry and a horizontal line, produced by `\hline` or `\cline{}` (refer §7.7 for `\cline{}`), is controlled by the `\extrarowheight` command defined in the `tabularx` package. A suitable value can be assigned to `\extrarowheight` (default value is `0pt`) for increasing such space, e.g., `\setlength{\extrarowheight}{3mm}` for creating an extra space of 3 mm. The `\setlength{\extrarowheight}{}` command is to be placed before starting the `tabular` or `tabularx` environment.

[4]The `\vline` command in the `tabular` and `tabularx` environments draws a vertical line, in the place of its application, having a height equal to that of a row.

Table 7.5 Some additional provisions for customizing a table

LaTeX input

```
\begin{table}[!hbt]
\centering
\setlength{\extrarowheight}{4mm}
\begin{tabular}{
        !{\vrule width 0.8mm}>{\bfseries}l|
        >{$}l<{$}!{\vrule width 0.8mm}}
 \hline Length        & {\bf In Metre}\\
 \hline 1 Millimeter  & 10^{-3}\\
 \hline 1 Centimeter  & 10^{-2}\\
 \hline 1 Decimeter   & 10^{-1}\\
 \hline 1 Decameter   & 10\\
 \hline 1 Hectometer  & 10^2\\
 \hline 1 Kilometer   & 10^3\\
 \hline
\end{tabular}
\end{table}
```

Output

Length	In Metre
1 Millimeter	10^{-3}
1 Centimeter	10^{-2}
1 Decimeter	10^{-1}
1 Decameter	10
1 Hectometer	10^{2}
1 Kilometer	10^{3}

Applications of some of the above provisions in a **tabular** environment are shown in Table 7.5. The **>{\bfseries}** command before the option **l** in the first column prints all the entries of that column in boldface fonts (while **{\bf }** in the second column prints only the heading in boldface fonts). The **>{$}l<{$}** command converts the second column into math-mode, for which the mathematical entries of that column (all other than its heading) could be inserted directly without creating a separate math-mode for an entry. Two vertical lines, each of width 0.8 mm, on both sides of the table have been obtained by the **!{\vrule width 0.8mm}** commands. On the other hand, the **\setlength{\extrarowheight}{4mm}** command before the **tabular** environment creates an extra vertical space of 4 mm above each row of the table.

7.7 Merging Rows and Columns of Tables

When presenting different types of information in a table, some cells are often required to be merged into a single one. The **multirow** package provides the **\multicolumn{}{}{}** and **\multirow{}{}{}** commands for merging two or more columns and rows, respectively. The applications of the commands are shown in Table 7.6 on the facing page.

In **\multicolumn{**n_c**}{calign}{centry}**, n_c is the number of columns to be merged, `calign` is the alignment of the merged column, and `centry` is the entry of that merged cell. Since four columns in the first row in Table 7.6 are merged into a single cell, the number of entries in that row is reduced from six to three (the **\multicolumn{}{}{}** command spanning a single column can also be used for changing the alignment in that column). The permitted `calign` in the **tabular** environment is **l** (for left alignment), **r** (for right alignment), or **c** (for center alignment). Note that the option **X** as `calign` in **\multicolumn{}{}{}** may not work properly under the **tabularx**

Table 7.6 Merging two or more cells of a table into a single one

LaTeX input

```
\begin{tabular}{|l|*{5}{c|}}
\hline \multirow{2}{*}{Name}&  \multicolumn{4}{c|}{Subjects}&
                                        \multirow{2}{*}{Total}\\
\cline{2-5}          &  Math  &  Phy  &  Chem  &  English  &    \\
\hline
\hline      Robin    &   80   &   68  &   60   &    57     &  265\\
\hline      Julie    &   72   &   62  &   66   &    63     &  263\\
\hline      Robert   &   75   &   70  &   71   &    69     &  285\\
\hline
\end{tabular}
```

Output

Name	Subjects				Total
	Math	Phy	Chem	English	
Robin	80	68	60	57	265
Julie	72	62	66	63	263
Robert	75	70	71	69	285

environment. Although many people suggest to use `>{\setlength{\hsize}{n_c\hsize}}X` instead of simply `X`, it also may not work properly in some LaTeX compilers. Hence, a good option is to use `p{}` with manually adjusted argument value (the option `X` generates a column using `p{}` internally with automatically adjusted argument value).

Similarly, in `\multirow{n_r}{cwidth}{centry}`, n_r is the number of rows to be merged, `cwidth` is the width of the merged cell, and `centry` is the entry of that merged cell. The value of `cwidth` can be set manually (e.g., `25mm` or `1.0in`), or can be obtained an auto-adjusted one using an `*` only. The entry in the merged cell, obtained through `\multirow{}{}`, is vertically center aligned. Other alignment can be obtained by assigning an optional argument after the second mandatory argument of the command, e.g., `\multirow{4}{2cm}[3mm]{centry}` for merging four rows to produce a single cell of width 2 cm and to print `centry` in it 3 mm above the vertical center line (a negative value to the optional argument will print `centry` below the vertical center line). When some rows in a column are merged, `\multirow{}{}` is used in the first row to be merged and the column in each of the remaining merged rows is left blank (i.e., the column is ended simply by a `&` or `\\`) as shown in the first and last columns in the second row of Table 7.6.

Further, the `\cline{m-n}` command is used in Table 7.6 for drawing a horizontal line covering columns `m` to `n` only. Another noticeable thing in Table 7.6 is the double horizontal line after the heading of the table. This is done using two consecutive `\hline` commands without any line break between them. Similarly, more than one vertical line can also be drawn using additional `|` symbol in the argument of the `\begin{tabular}{}` command.

Note that both the `\multirow{}{}` and `\multicolumn{}{}` commands can be used together for creating a single cell by merging a number of rows and columns. One such example is `\multicolumn{3}{|c|}{\multirow{2}{*}{Outcome}}`, where three columns and two rows are merged into a single cell for printing '`Outcome`' with center alignment.

7.8 Table Wrapped by Texts*

If the size of a table is very small compared to the width of a page, the **wraptable** environment, supported by the **wrapfig** package, can be used to wrap around the table by texts. The **wraptable** environment needs two mandatory arguments, i.e., **\begin{wraptable}**`{aside}{asize}`, where `aside` and `asize` are, respectively, the location and size of the table. The location can be specified by **l** (left side of the page) or **r** (right side of the page), while the size is specified in units (e.g., `25`**mm**, `1.0`**in**, or `0.3`**\linewidth**). The **wraptable** environment is similar with the **table** environment; the only difference lies in creating the environment. A self-explanatory application of this environment is shown in Table 7.7.

Table 7.7 Table wrapped by texts through the **wraptable** environment

LaTeX input

```
\begin{wraptable}{r}{5cm}
\centering
\caption{Obtained marks.}
\label{wrap-table}
\begin{tabular}{|l|c|c|c|}
 \hline Name   & Math & Phy & Chem\\
 \hline Robin  & 80   & 68  & 60\\
 \hline Julie  & 72   & 62  & 66\\
 \hline Robert & 75   & 70  & 71\\
 \hline
\end{tabular}
\end{wraptable}
Marks obtained by ... are shown in
Table~\ref{wrap-table}, ...
```

Output

Marks obtained by Robin, Julie and Robert in Math, Phy and Chem are shown in Table 3, which is wrapped around by texts for saving space. However, the total marks obtained by the students could not shown in the table due to space limit. You can find it by hand calculation only.

Table 3: Obtained marks.

Name	Math	Phy	Chem
Robin	80	68	60
Julie	72	62	66
Robert	75	70	71

7.9 Table with Colored Background*

In order to make some entries of a table prominent, the **colortbl** package provides the **\rowcolor{}**, **\columncolor{}**, and **\cellcolor{}** commands for coloring, respectively, any row, column, and cell of a table by the color specified as the argument of a command (refer §2.4 on page 13 for detail of colors). For **gray** color, optional provision is there for specifying its intensity also, in which case the commands take the forms of **\rowcolor[gray]{**`x`**}**, **\columncolor[gray]{**`x`**}**, and **\cellcolor[gray]{**`x`**}**, where `x` is the intensity of the **gray** color to be specified by a number between 0 and 1. The row to be colored is to be started with a **\rowcolor{}** command, while a **\cellcolor{}** command is to be entered in the particular cell to be colored. On the other hand, a colored column is to be generated using a **\columncolor{}** command, in the form of **>{\columncolor{}}**, in the argument of the table-generating environment **tabular** or **tabularx**, e.g., **\begin{tabular}{|l|>{\columncolor{green}}c|r|}** command will generate the middle column center aligned and colored by green color.

As a major drawback, \rowcolor{}, \columncolor{}, and \cellcolor{} may override column-separating vertical lines and row-separating horizontal lines in some cases. There is no formal rule to preserve them. Column-separating vertical lines can be preserved by controlling the amounts of color panel overhang on either side of a column. This can be done though two optional arguments to \rowcolor{} and \columncolor{} in the forms of \rowcolor{}[lhang][rhang] and \columncolor{}[lhang][rhang], where lhang and rhang are, respectively, the amounts of overhang on the left and right sides of a column. Without these two optional arguments, a color panel overhangs by default amount of \tabcolsep, while lhang equals rhang if only one is present. On the other hand, for preserving a row-separating horizontal line, \rule{0pt}{rhgt}\noindent may be used, where rhgt is the height of the zero-width rule generated by the \rule{}{} command. Based on some trials, the values of lhang, rhang, and rhgt may be fixed manually, e.g., \rowcolor{green}[0.9\tabcolsep] or \columncolor{blue}[0pt], or \rule{0pt}{2.6ex}\noindent.

Applications of the \rowcolor{}, \columncolor{}, and \cellcolor{} commands, as stated above, are shown in Table 7.8. Because of the repeated application of

Table 7.8 Table with colored background through the \rowcolor{}, \columncolor{}, and \cellcolor{} commands

LaTeX input

```
\begin{table}[!hbt]
\begin{tabular}{|c|l|r|r|}
\hline SN& Item& Price& Amount\\
\hline \rowcolor{red}[0.9\tabcolsep]
    \rule{0pt}{2.9ex}\noindent 1& Rice& 35& 140\\
\hline \rowcolor[gray]{0.7}[0.91\tabcolsep]
    \rule{0pt}{2.7ex}\noindent 2& Sugar& 55& 55\\
\hline \rowcolor{green}[0.91\tabcolsep]
    \rule{0pt}{2.7ex}\noindent 3& Atta& 75& 75\\
\hline \multicolumn{3}{|r|}{Total}&
        \cellcolor{blue} 270\\
\hline
\end{tabular}
\end{table}
```

Output

SN	Item	Price	Amount
1	Rice	35	140
2	Sugar	55	55
3	Atta	75	75
		Total	270

```
\newcolumntype{B}[2]
    {>{\columncolor{#1}[0.91\tabcolsep]}#2}
\begin{table}[!hbt]
\begin{tabular}{|c|B{gray}{l}|B{red}{r}|B{green}{r}|}
\hline \rowcolor{white} \rule{0pt}{2.8ex}\noindent
        SN& Item& Price& Amount\\
\hline 1& Rice& 35& 140\\
\hline \rowcolor{white} \rule{0pt}{2.8ex}\noindent
        2& Sugar& 55& 55\\
\hline 3& Atta& 75& 75\\
\hline \rowcolor{white} \multicolumn{3}{|r|}{Total}&
        \cellcolor{blue} 270\\
\hline
\end{tabular}
\end{table}
```

Output

SN	Item	Price	Amount
1	Rice	35	140
2	Sugar	55	55
3	Atta	75	75
		Total	270

\columncolor{} in the second example, a new column type with two arguments, B{ccol}{calign}, is defined through \newcolumntype{}[]{}, where ccol is the color argument of \columncolor{} and calign is the alignment of the column to be

generated. Note that any of **\rowcolor{}**, **\columncolor{}**, and **\cellcolor{}** overrides their earlier use in a table, which is shown in the second example in Table 7.8, where **\rowcolor{}** overrides **\columncolor{}** and **\cellcolor{}** overrides **\rowcolor{}**.

Also note that the **\rowcolor{}**, **\columncolor{}**, and **\cellcolor{}** commands may not work properly along with the column margin adjusting command **@{}** discussed in §7.5 on page 63 (in such a requirement, however, the **\tabcolsep** command may be redefined).

Hour 9

Figure Insertion

LaTeX has the provision for inserting a figure from an external file in different formats. As stated in §1.4 on page 4, a LaTeX file can be compiled using either the latex or pdflatex command. When a LaTeX file involves figures from external files, either of the compilation commands is to be used based on the format of the figures. Note that the file formats of all the figures inserted in a LaTeX document must be supported by a single compilation command, either latex or pdflatex. Commands for compiling LaTeX files involving some standard and widely used figure formats are given in Table 9.1. It is to be mentioned that different tools, like `xfig` and `gimp` on Unix

Table 9.1 LaTeX compilation commands and supported figure formats

Compilation command	Supported figure format	
	Short name	**Full name**
latex	`eps`	Encapsulated PostScript
	`ps`	PostScript
pdflatex	`pdf`	Portable Document Format
	`jpeg`	Joint Photographic Expert Group
	`tiff`	Tag Index File Format
	`png`	Portable Network Graphic

system or `ImageMagick` and `netpbm` on both Unix and Windows systems, can be used for exporting figures from one format to another.

9.1 Commands and Environment for Inserting Figures

An `eps` format figure can be inserted using the **\epsfig{file=**`fname`**}** command defined in the **epsfig** package, where `fname` is the name of the figure file with or without the '`eps`' extension. Apart from the mandatory `fname`, the size of a figure can also be specified in **\epsfig{}** through two optional fields, **width** and **height**, one separated from another by a comma. Without any of the **width** and **height**, a figure is printed in its original size. If one of them is specified, the other one

D. Datta, *LaTeX in 24 Hours: A Practical Guide for Scientific Writing*,
DOI 10.1007/978-3-319-47831-9_9

is automatically taken in proportion. On the other hand, the presence of both **width** and **height** prints a figure in the specified fixed size (in this case, the figure may get distorted if their values are not set properly). In addition to specifying the size, a figure can also be rotated through the option **angle=**`theta`, where a positive value of `theta` (in degree) will rotate the figure in counter-clockwise direction and a negative value in clockwise direction. With such provisions, a figure can be inserted as **\epsfig{file=**`myfig.eps`**}** or **\epsfig{file=**`pics/myfig.eps`**, width=**`0.5`**\linewidth}** or **\epsfig{file=**`myfig.eps`**, width=**`30`**mm, height=**`40`**mm, angle=**`30`**}**, where `myfig.eps` is the name of the figure file and `pics` is the folder containing the figure file.

The more general command for inserting a figure from an external file is **\includegraphics[**`aopt`**]{**`fname`**}** defined in the **graphicx** package, where `fname` is the name of the figure file without its extension, and `aopt` is(are) the option(s) like **width**, **height** and **angle**. The advantage of using **\includegraphics[]{}** is that a figure in any format can be inserted without making any change in the LaTeX input file[1].

Similar to nesting the **tabular** or **tabularx** environment in the **table** environment as discussed in §7.1 on page 59, the **\epsfig{}** and **\includegraphics[]{}** commands can be used in the **figure** environment, so that a figure can be assigned a serial number and a caption through the **\caption{}** command, as well as a reference key through the **\label{}** command for the purpose of referring it anywhere within a document. Further, similar to the **table** environment, the **figure** environment can also be created as **\begin{figure}[]** with optional preferences in **[]** for vertical positioning of a figure. The standard preferences for vertical positioning are **H**, and any or combination of **h**, **b** and **t** along with **!** (refer §7.3 on page 62 for detail of **[H]** and **[!hbt]**).

9.2 Inserting a Simple Figure

Three examples of inserting a figure, named as `girl.eps`, through the **\epsfig{}** command are shown in Table 9.2 on the next page. In the first example, the size of the figure is specified by both width and height, which have produced the figure in a distorted form due to the consideration of their non-proportionate values. In the second example, since only the width of the figure is specified, a scaled form of the original figure is produced by automatically adjusting its height. In the third example, apart from specifying the width of the figure, it is also rotated by 30° in the counter-clockwise direction.

The first command in the **figure** environment in Table 9.2 is **\centering**, which instructs for width-wise center alignment of its figure (other commands could be **\flushleft** for left alignment or **\flushright** for right alignment). After inserting the figure through **\epsfig{}**, the **\caption{}** command is used for assigning a serial number to the figure. The **\caption{}** command will also produce a title of the figure, if any is provided as the argument of the command (since the title usually comes at the bottom of a figure, **\caption{}** is used after **\epsfig{}**). The **\caption{}** command is followed

[1]The advantage of using the **\includegraphics[]{}** command for inserting figures from external files is that a figure in any format can be inserted without making any change in the LaTeX input file.

Table 9.2 Figure insertion through the \epsfig{} command

LaTeX input	Output
\begin{figure}[!hbt] \centering \epsfig{file=girl.eps, width=2.0cm, height=2.0cm} \caption{A girl.} \label{girl1} \end{figure}	 Figure 1: A girl.
\begin{figure}[!hbt] \centering \epsfig{file=girl.eps, width=2.0cm} \caption{A girl.} \label{girl2} \end{figure}	 Figure 2: A girl.
\begin{figure}[!hbt] \centering \epsfig{file=girl.eps, width=2.0cm, angle=30} \caption{A girl.} \label{girl3} \end{figure}	 Figure 3: A girl.

by the \label{} command for assigning a unique reference key, which can be used for referring the figure through the \ref{} command. Note that \label{} does not have any effect without \caption{}, in which case a figure will not be assigned any serial number for referring it.

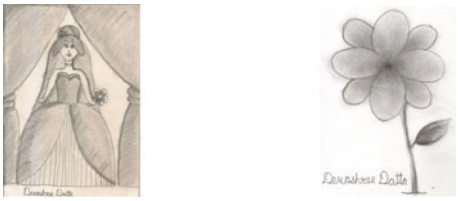
The same outputs, shown in Table 9.2, can be produced through the \includegraphics[]{} command also. In that case, just the \epsfig{} command from the examples is to be replaced by \includegraphics[width=2cm, height=2cm]{girl}, \includegraphics[width=2cm]{girl} and \includegraphics[width=2cm, angle=30]{girl}, respectively. Since a figure in any format can be inserted through \includegraphics[]{} without making any change in the LaTeX input file, now onwards all figures in this book will be inserted through \includegraphics[]{} only, otherwise mentioned specifically.

9.3 Side-by-Side Figures*

In the examples in §9.2, only one figure is inserted in a row. Provision is also there in LaTeX for inserting multiple figures side by side in a single row. As

Table 9.3 Side-by-side figures in a single row

LaTeX input	Output

`\begin{figure}[!hbt}`
`\centering`
`\includegraphics[width=2.0cm]{girl}\hfill`
`\includegraphics[width=2.0cm]{flower}`
`\caption{A girl and a flower.}`
`\label{girl_flower}`
`\end{figure}`
	 Figure 4: A girl and a flower.

shown in Table 9.3, this can simply be done by inserting each figure through a separate `\includegraphics[]{}` in a single `figure` environment. The requirements for the same are: there should not be any line break or new line command between two `\includegraphics{}`, and total width of all the figures should not exceed the page width; otherwise the figures will be inserted one below another. However, `\hfill` can be used between two `\includegraphics{}` for separating the corresponding figures by available horizontal space.

Note that both the side-by-side figures in Table 9.3 are assigned a single serial number as a whole. If the side-by-side figures are to be assigned individual serial number, the `minipage` environment may be used. As stated in §8.3 on page 73, the `minipage` environment splits a page width-wise into a number of parts, each of which can be used for inserting a figure, drawing a table, or even for writing selected texts. Table 9.4 shows two figures, inserted side-by-side by using the `minipage` environment, which are assigned individual serial number by each `\caption{}` command. Within a single `figure` environment, two `minipage` environments, each of size `0.4\linewidth`, are created for inserting the two figures.

Table 9.4 Side-by-side figures through the `minipage` environment

LaTeX input	Output

`\begin{figure}[!hbt]`
`\begin{minipage}[c]{0.4\linewidth}`
`\centering`
`\includegraphics[width=3.0cm]{girl}`
`\caption{A girl.}`
`\label{girl}`
`\end{minipage}\hfill`
`%`
`\begin{minipage}[c]{0.4\linewidth}`
`\centering`
`\includegraphics[width=2.5cm]{flower}`
`\caption{A flower.}`
`\label{flower}`
`\end{minipage}`
`\end{figure}`
	 Figure 5: A girl.  Figure 6: A flower.

9.4 Sub-numbering a Group of Figures

In some cases, a group of figures may need to be sub-numbered under a main number, e.g., 3(a) or 5(e). Within the **figure** environment, the **\subfigure[**`atitle`**]{**`afig`**}** command defined in the **subfigure** package[2] (or the new **\subfloat[**`atitle`**]{**`afig`**}** command defined in the **subfig** package) may be used for inserting a figure with a sub-numbering, where optional `atitle` is the title of the figure, and mandatory `afig` is the insertion of the figure either through the **\epsfig{}** or **\includegraphics[]{}** command. For the purpose of referring, a subfigure can be assigned a reference key through the **\label{}** command inside the mandatory argument of **\subfigure[]{}**. Moreover, the group of subfigures can be captioned and labeled as a whole using respectively the **\caption{}** and **\label{}** commands inside the **figure** environment. Table 9.5

Table 9.5 Sub-numbering a group of figures using the **\subfigure[]{}** command

LaTeX input

```
\begin{figure}[!htb]
\centering
\subfigure[A girl.]
{   \includegraphics[width=2.0cm]{girl}
    \label{girl}
}\hfill
\subfigure[A flower.]
{   \includegraphics[width=2.0cm]{flower}
    \label{flower}
}\\
\subfigure[A finger work.]
{   \includegraphics[width=4.0cm]{finger}
    \label{finger-work}
}
\caption{Girl, flower and finger work.}
\label{girl_flower_finger}
\end{figure}
%
In Figure~\ref{girl_flower_finger}, \ref{girl}
and \ref{flower} display a girl and a flower,
while \ref{finger} displays a beautiful finger
work.
```

Output

(a) A girl.

(b) A flower.

(c) A finger work

Figure 7: Girl, flower and finger work.

In Figure 7, 7(a) and 7(b) display a girl and a flower, while 7(c) displays a beautiful finger work.

shows such an example, which contains three figures in a group. It is also shown in Table 9.5 that the subfigures can be inserted in a single row or even in multiple rows (for inserting a subfigure in the next row, a line break command **** is to be used at the end of the previous **\subfigure[]{}** command).

[2] The **subfigure** package loaded as **\usepackage[tight]{subfigure}** with the **tight** option reduces the excess vertical blank space between a subfigure and its caption (the function of this **tight** is similar with those of the **\abovecaptionskip** and **\belowcaptionskip** commands addressed in §8.8 and §9.10).

The \subfigure[]{} command numbers a group of subfigures as (a), (b), etc. This default numbering can be changed by redefining the \thesubfigure command, e.g., \renewcommand{\thesubfigure}{(\roman{subfigure})~} for numbering subfigures by (i), (ii), etc. Note that a subfigure is not assigned any sub-number without the optional argument to \subfigure{}. However, it is counted while numbering the remaining subfigures of the group.

9.5 Figure Wrapped by Texts*

Like a table as stated in §7.8 on page 68, a figure can also be wrapped around by texts, which is done through the wrapfigure environment[3] defined in the wrapfig package. The wrapfigure environment, which creates a space for inserting a figure, takes one optional and two mandatory arguments, i.e., \begin{wrapfigure}[aline]{aside}{asize}, where aside is the location of the space (l for left side or r for right side), while asize is the size of horizontal space and aline is the number of lines of texts for vertical space. If insufficient number of lines (i.e., aline) is opted, the figure will be overlapped with other lines of texts. If number of lines to be wrapped is not opted, it is set automatically to cover the size of the figure. An example of the wrapfigure environment is shown in Table 9.6, where

Table 9.6 Figure wrapped by texts

LaTeX input	Output
\begin{wrapfigure}[10]{r}{2.3cm} \includegraphics[width=2.0cm]{girl} \caption{Girl.} \label{girl} \end{wrapfigure} % The picture shown on the right side was drawn by Devoshree, a 8 year old girl....	The picture shown on the right side was drawn by Devoshree, a 8 year old girl. She loves arts from her childhood, starting from the age of around 2 years. Seeing her sketching something on a wall and enlarging it every day at that age, we did not stop her but encouraged for the same. Today she can do very beautiful sketching and other artistic works. Only three simple works from her activities are included in this Hour.  Figure 8: Girl.

a figure of width 2.0 cm is inserted through \includegraphics[]{} in a horizontal space of 2.3 cm on the right side of the page wrapping 10 lines of texts of the document.

[3]The wrapfigure environment may not work properly inside other environments. Moreover, the environment should be put at the top of a paragraph, or between two words where a line break exists.

9.6 Rotated Figure

It is shown in Table 9.2 on page 83 that a figure can be rotated by a given angle through the **angle** option to the **\epsfig{}** (also to **\includegraphics[]{}**) command. Further, like the **sidewaystable** environment for producing a table on a landscape-size page as discussed in §8.4 on page 75, the **sidewaysfigure** environment can be used for producing a figure in landscape-mode on a new page. An example of the **sidewaysfigure** environment is shown in Table 9.7 (output is not shown).

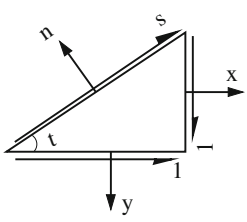
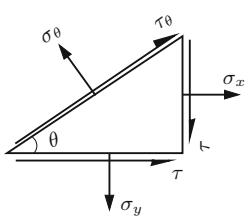
Table 9.7 Rotated figure on a landscape page through the **sidewaysfigure** environment

```
\begin{sidewaysfigure}
\includegraphics[width=\linewidth]{computer}
\caption{A computer on a landscape page.}
\end{sidewaysfigure}
```

9.7 Mathematical Notations in Figures*

There is always a question how to put mathematical notations in figures, like σ_x or e^z. Generally graphics plotters do not have such provisions, but for plain texts and symbols only. The **\psfrag{atag}{acomm}** command, defined in the **psfrag** package, can replace an alphabetical or a numerical tag (`atag`) in an encapsulated postscript (`eps` format) figure with an arbitrary instruction including LaTeX commands (`acomm`). An application of **\psfrag{}{}** is shown in Table 9.8, in which the alphabetical and

Table 9.8 Mathematical notations in figures through the **\psfrag{}{}** command

Original figure	LaTeX input	LaTeX output
	**\begin{figure}[!hbt]** **\centering** **\psfrag{x}{σ_x}** **\psfrag{y}{σ_y}** **\psfrag{n}{σ_{θ}}** **\psfrag{s}{τ_{θ}}** **\psfrag{t}{θ}** **\psfrag{1}{τ}** **\includegraphics[width=5cm]{psfrag}** **\end{figure}**	

numerical tags of the original `eps` figure shown in the left column are replaced, in the figure shown on the right column, by the LaTeX commands inserted as the second argument of the **\psfrag{}{}** commands shown in the middle column. Note that the **\psfrag{}{}** commands are to be inserted before inserting the figure.

If a figure is prepared in the `xfig` software, LaTeX commands can directly be inserted in the `.fig` file. Then the required `.eps` file can be obtained from the `.fig` file using the `fig2eps` command in a terminal, e.g., '$ `fig2eps mypic.fig`' for

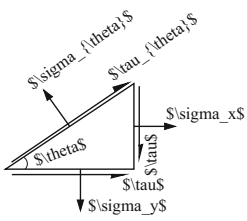
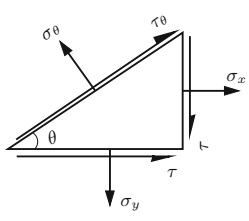
generating `mypic.eps` from `mypic.fig`. However, the process needs the `fig2ps` software installed in the computer.

Alternatively, if the `fig2ps` software package is not available, the `fig` file may be exported in a different way to generate a `.pstex_t` file, the contents of which can then be inserted (pasted) in the LaTeX input file in order to achieve the desired effect. The step-by-step procedure of this process is given below:

1. Draw the figure in the `xfig` software package.
2. Click the text-mode '`T`' button on the top-left side of the `xfig` window. Then click the '`Text Flags hidden=off`' button on the bottom side of the `xfig` window, which will open a small dialogue box. In that dialogue box, change the status of '`Special Flag`' from '`Normal`' to '`Special`' and then click '`Set`'.
3. Now typeset LaTeX commands in the required locations of the figure. Math-mode commands are to be inserted between a pair of **$** symbols.
4. Save the file by a name, say `myfig.fig`.
5. Click '`Export`' under the '`File`' button in the menu bar on the top of the `xfig` window, which will open a new dialogue box. Under '`Language`' on the top side of that dialogue box, select '`Combined PS/LaTeX (both parts)`' and then click '`Export`' button.
6. Two new files now should be available in the working folder, `myfig.pstex` and `myfig.pstex_t` (if the figure file in Step (5) above was named by `myfig.fig`).
7. Open `myfig.pstex_t` in a text-editor, and then copy and paste its contents in the required location of the LaTeX input file.

An application of inserting mathematical notations in figures by putting LaTeX commands in the `xfig` software is shown in Table 9.9, in which the LaTeX commands of

Table 9.9 Mathematical notations in figures through `xfig` software

Original .fig figure	Contents of .pstex_t file	LaTeX output
	**\begin{picture}**(0,0) **\includegraphics{**myfig.pstex**}%** **\end{picture}%** **\setlength{\unitlength}{**3947sp**}%** ⋮ **\end{picture}%**	

the original `fig` figure shown in the left column are executed, in the figure shown on the right column, by inserting the contents of the `myfig.pstex_t` file in the LaTeX input file as shown in the middle column. Since directly copied and pasted, it is not necessary to understand the contents of the self-generated `.pstex_t` file.

9.8 Figures in Tables*

Sometimes a figure may need to be produced in a cell of a table. In that case also, the figure can be inserted in the cell through **\includegraphics[]{}** just like an ordinary entry in a cell of a table. However, since the **figure** environment cannot be nested inside the **table** environment, **\includegraphics[]{}** cannot be put in a **figure** environment, and hence, the figure can neither be captioned nor numbered. A number of examples can be found in this book, where figures are inserted in tables, refer, e.g., Tables 9.2, 9.3, 9.4, 9.5 and 9.6.

9.9 Figures in Multi-column Documents

In a multi-column document, where the contents are produced in multiple columns on a page, a figure is also placed in a column (refer §4.3 on page 29 for detail of multi-column documents). However, if the width of the column is not large enough to accommodate a figure in it, the **figure*** environment can be used for inserting the figure on the entire width of the page[4]. In that case, the **\begin{figure}** and **\end{figure}** commands are to be replaced by the **\begin{figure*}** and **\end{figure*}** commands, respectively.

9.10 Changing Printing Format of Figures*

The serial number of a figure is preceded by the default label-word '`Figure`', which can be changed by defining **\def\figurename{}** in the preamble, e.g., **\def\figurename{**`Fig.`**}** will replace '`Figure`' by '`Fig.`'. Moreover, the size and type of fonts for the label-word and caption can also be changed by using **\captionsetup{}** as discussed in §8.8 on page 79. Similarly, **\abovecaptionskip** and **\belowcaptionskip**, mentioned in §8.8 for reducing the excess vertical blank space before and after the main caption of a table, is applicable in the case of figures also.

In the document-class **book**, the serial number of a figure is composed of two parts. For example, refer Fig. 4.1 on page 27, where the figure number is composed of 4 and 1 separated by a period mark. In this case, 4 is the number of the Hour and digit 1 is the number of the figure in that Hour. In contrast, a figure in the document-class **article** is assigned its serial number only, i.e., not preceded the number of the section in which the figure belongs (§11.4.3 on page 105 and §19.2.5 on page 189 discuss the process for obtaining section-wise numbers to figures in the document-class **article**).

[4]If the column width of a multi-column document is not sufficient to accommodate a figure in it, the **figure*** environment may be used for inserting the figure over the entire width of the page.

9.11 Figures at the End of a Document

Refer §8.9 on page 80 for detail.

9.12 Editing LaTeX Input File Involving Many Figures*

The compilation time of a LaTeX file involving a lot of figures may be very large. Hence, the **\psdraft** command may be inserted in the preamble when the file needs repeated editing and compilation. Instead of producing a figure, the command instructs the LaTeX compiler just print the name of the figure file in a box in the specified location, thus saves compilation time. However, **\psdraft** is to be deleted or commented before the final compilation. In an alternate way, the **epsfig** or **graphicx** package, as applicable, may be loaded with the **draft** option as **\usepackage[draft]{epsfig}** and **\usepackage[draft]{graphicx}** (or jointly as **\usepackage[draft]{epsfig,graphicx}**) for performing the same job. In this case also, the **draft** option is to be omitted before the final compilation.