This document was prepared in `\sectionstyle{center}` with `\indentaftersection`.
This heading was printed with `\thispageheading` command from the **watermark** package

NCC-LaTeX

by Alexander I. Rozhenko

Abstract. This document describes classes and packages from NCC-LaTeX. We compare the **ncc** class with standard LaTeX's classes and point out a difference between the **ncc** class and **ncclatex** package. Packages from the *ncclatex* and *ncctools* bungles are partially described here.

Contents

1. The ncc class

To mark out features of the **ncc** class, we compare it with standard LaTeX classes **article**, **book**, and **report**. The first and the main distinction is that preparing of articles, books, and reports in NCC-LaTeX is provided with special options (called **styles**) of the **ncc** class instead of using different classes. The following style options are allowed for this:

`article` prepares an article.

`preprint` prepares an article with a title going on a separate page. The `\preprint{`*No*`}` command allows inserting a preprint number under the preprint title.

`book` prepares a book. The `\bookeditor{`*text*`}` command allows inserting a text under the book title.

`report` is equivalent to the `book` option used together with the `oneside` option.

The `titlepage` and `notitlepage` options are not used in the **ncc** class. The appearance of a title on a separate page is controlled in a style used. An `article` style specifies running title and others specify a title on a separate page. The `abstract` environment is defined in all styles. It is prepared on a separate page if a title is prepared on a separate page.

The page style `headings` is set by default in all styles except articles for which the `myheadings` style is used.

The default options for the **ncc** class are `10pt`, `a4paper`, `article`, `twoside`, `onecolumn`, `final`, `openany`.

The class sets `\unitlength` to 1 mm (i.e. all coordinates in the `picture` environment are calculated in millimeters).

1.1. Page layout

All standard paper size options are supported, namely `a4paper`, `a5paper`, `b5paper`, `letterpaper`, `legalpaper`, `executivepaper`, and `landscape`. One more paper size option is introduced: `a5a4paper`. It is useful for printing a5 papers on a4 printer having centered tray.

In contrast to standard classes, the width and height of the text field isn't automatically expanded on the entire page. Text field dimensions are defined depending on the base point size:

Point size	Text width	Text height
10 pt	110 mm	157 mm
11 pt	126.5 mm	199 mm
12 pt	145 mm	233 mm
14 pt	160 mm	240 mm

If you want to fit the text field to the entire page in the same manner as in standard classes, use the `fittopage` option.

The text field is centered on the page taking into account the header field also, but the margin from the top is corrected to be at most 1.5 inch. If you want to center the text field taking into account the marginal notes, header, and footer, use the command

```
\ToCenter[hfm]{\textwidth}{\textheight}
```

in the preamble of a document. The optional parameter of this command enumerates fields counted while centering. Using this command, you can change the width and height of the text field and center the result on the page. Other way for changing the width and height is using the `\FromMargins` command. For example, the following command

```
\FromMargins[hf]{20mm}{10mm}{25mm}{15mm}
```

calculates the text field dimensions and margins in such a way to provide 20 mm distance from the left border of the page, 10 mm distance from the right border, 25 mm distance from the top, and 15 mm distance from the bottom in assumption that header and footer are in use. In two-side printing the left and right distances are swapped for even pages.

1.2. Base point size options

Along with standard base point size options `10pt`, `11pt`, and `12pt`, the `14pt` size option is introduced. In 14 pt, font sizes `\huge` and `\Huge` are equal to `\LARGE`.

Sometimes, the standard sizes of fonts used in section markup commands are a bit larger than necessary. This case appears when a page size is small enough or a base font size is too large. The `small` option can be used in this cases to reduce font sizes used in section markup commands. The `14pt` option applies it.

1.3. Other options

The following standard options can be used with the **ncc** class: `oneside`, `twoside`, `draft`, `final`, `openright`, `openany`, `onecolumn`, `twocolumn`, `openbib`, `fleqn`, and `leqno`.

The `russian` option loads the **babel** package with the `[russian]` optional parameter.

1.4. Packages loaded by the ncc class

The **ncc** class loads many packages from *ncclatex* and *ncctools* bungles. They are:

ncclatex is the kernel of NCC-LaTeX;

nccltrus is loaded when the `russian` option is used. It redefines titles of captions to russian ones;

nccold provides some obsolete commands from old NCC-LaTeX, namely `\no`, `\eref`, `\Eq`, `\Eqs`, `\Eqalign`, `\tbox`, `\bbox`, `\tbbox`, `\emline`, and `\emlinewidth`;

dcounter provides dynamic counters;

desclist improves the `description` environment, and provides the `desclist` environment with better control of marker in descriptions;

extdash provides shortcut commands `\-/`, `\--`, `\---`, `\=/`, `\==`, and `\===` for better control of hyphenation of compound words;

nccboxes provides more boxes;

nccfloats provides commands `\minifig`, `\fig`, `\figs`, `\minitabl`, `\tabl`, and `\tabls` covering standard LaTeX floats and preparing the floating material in NCC-LaTeX style (a text in NCC-LaTeX floats is centered and prepared in the `\footnotesize`). New features, `\sidefig` and `\sidetabl`, simplify creation of minifloats posed on the outer side of page;

nccfoots provides commands `\Footnote`, `\Footnotemark`, and `\Footnotetext` for manual management of footnote markers.

nccmath extends the **amsmath** package with NCC-LaTeX commands for preparing display equations (`\eq`, `\eqs`, `\eqalign`), redefines the `eqnarray` environment to work properly together with $\mathcal{AMS}$ display equations, introduces the `darray`, `fleqn`, and `ceqn` environments;

nccpic is an envelop for the **graphicx** package (provides extension lists for useful dvi-drivers, introduces the `\putimage` command for compatibility with old NCC-LaTeX, and defines the `\ipic` command);

nccsect provides an improved section, caption, and TOC-entries management;

nccthm theorems in NCC-LaTeX style (customizable layout, many new features);

tocenter provides the `\ToCenter` and `\FromMargins` commands;

watermark provides watermarks.

The **ncc** class loads some standard LaTeX packages. They are **amsmath**, **graphicx**, and **makeidx**. The **babel** package is also loaded if the `russian` option is used.

1.5. New commands of the ncc class

We describe here new commands introduced in the **ncc** class only. Other new commands are introduced in packages of *ncclatex* and *ncctools* bungles (most of them are common for the **ncc** class and **ncclatex** package).

`\partmark{`*text*`}` is used for control the header and footer marks after a new part.

`\openrightorany` produces the `\clearpage` or `\cleardoublepage` command depending on open mode selected in the class options. If an empty even page is created, its header mark is set empty.

`\AuthorBeforeTitle` and `\TitleBeforeAuthor` determine what item must go first in the title: author text or title text. In article style the title text goes first and in other styles the author goes first. These commands should be used in the preamble.

`\noeqbreak` and `\alloweqbreak` manage the possibility of page break between a text and display equation following after. The first command forbids such page breaks and the last command allows them. The default is `\alloweqbreak`.

`\SetTOCStyle{`*style*`}` sets a style for table of contents, list of tables, and list of figures. The default style is empty. For example, the use of `\SetTOCStyle{\small}` in the preamble of the document sets a small font for output in TOC.

`\setyear{`*year*`}`, `\theyear`: the first command sets a year and the second one prints an year set before. The `\setyear` command doesn't change the `\year` counter. The default value for the `\theyear` command is the current value of the `\year` counter.

`biblist` is a new environment:

`\begin{biblist}[`*starting No*`]{`*prototype*`}` ... `\end{biblist}`.

It prints a pure bibliography list without heading. The first optional parameter sets an initial value for the bibliography counter. The style of preparing the bibliography is controlled with the `\bibliststyle` command (its default value is `\small`). The `thebibliography` environment is prepared on the base of the `biblist`.

`theglossary` environment is introduced. The `\printindex`, `\see`, and `\seealso` commands are loaded by the **makeidx** package. The `\printglossary` command is introduced.

The `figure` and `table` counters are declared dynamically. So, they are controlled with the `\countstyle` command. The default count style for article and preprint is empty, and for book and report the `\countstyle{chapter}` is used. The `equation` counter is also controlled with the `\countstyle` command.

Specific commands for the article style. Three new commands

`\setseries{`*text*`}`, `\setvolume{`*Volume No*`}`, and `\setissue{`*Issue No*`}`

are allowed in the article style. They produce the commands `\theseries`, `\thevolume`, and `\theissue` which can be used in publishing. In the article style the **ncc** class tries to load an **nccadd.sty** file. This style file is useful in publishing to set common information for all articles in a journal issue. The number of the last page is marked with the `NCC@lastpage` label. Its `\pageref` value can be used in publishing to automate the substitution of article last page number. If the `openright` option is used, `\cleardoublepage` action is applied at the end of article.

Specific commands for the preprint style. The `\preprint{`*`Preprint No`*`}` command is introduced. If a preprint number is defined, the following text

Preprint number

is inserted below the title of preprint.

Specific commands for the book and report styles. The `\frontmatter`, `\mainmatter`, and `\backmatter` commands can be used. The `\bookeditor{`*`text`*`}` command inserts the specified text below the title of book. A style of chapter prefixing in running head and in the table of contents is controlled with the `\ChapterPrefixStyle{`*`list`*`}` command. The list can contain comma separated words `header` and `toc`. Using them you set prefix style for header and/or toc respectively. In prefix style the chapter number is preceded with the **Chapter** or **Appendix** text.

1.6. Using fancy headers with the ncc class

The headers in **ncc** class are maintained a bit different from standard classes. So, if you want to use fancy headers together with the **ncc** class, type

```
\usepackage[list-of-page-styles]{ncchdr}
```

in the preamble of document. The *`list-of-page-styles`* parameter must contain a list of page styles to be redefined, i.e., something from `empty`, `plain`, `headings`, and `myheadings`. The last style in the list is set after using of this package.

Note: the `headings` and `myheadings` styles have a common part in the **ncc** class which is redefined in this package. So, redefining one of these page styles you automatically redefine another style also. Additional fancy page style `title` is provided with the **ncchdr** package. It uses three title marks, namely `\lefttitlemark`, `\titlemark` (in center), and `\righttitlemark`. To load this style use the `title` option with the package.

The package is based on the **nccfancyhdr** package from the *ncctools* bungle.

2. The ncclatex package

The most part of the **ncc** class is loaded in the **ncclatex** package. To provide almost all functionality of NCC-LaTeX with standard LaTeX classes, you can use this package. The package loads many packages also. It is easy to say what packages it does not load from the list specified in the Section 1.4. Only three of them are not loaded, namely **nccold**, **tocenter**, and **watermark**.

If you want to pass additional options to a package loaded with **ncclatex**, load it in the `\usepackage` command before loading the **ncclatex**.

The **ncclatex** package has two options: `small` and `russian`. Using of the `small` option provides smaller fonts in section markup commands. The `russian` option tries to load the **babel** package with `[russian]` optional parameter, redefines captions of mathematical statements in Russian by loading the **nccltrus** package, and redefines `\alph` and `\Alph` commands to produce

russian alphanumeric numbers. Original latin variants of alphanumeric numbering are saved in `\alphlatin` and `\Alphlatin`.

The package sets `\unitlength` to 1 mm (millimeters are default units in NCC-LaTeX).

2.1. Sections, captions, and toc-entries in NCC-LaTeX

2.1.1. Sections

Section markup commands of level 0 and greater are redefined using the **nccsect** package from *ncctoools* (the level 0 command is `\chapter` or `\part` depending on class used; the commands of greater levels are `\section`, `\subsection`, ..., `\subparagraph`). All section markup commands are divided into two categories: display sections (until `\subsubsection`) and run-in sections (`\paragraph` and `\subparagraph`).

The package uses floating horizontal alignment for display sections which is controlled with the

`\sectionstyle{`*style*`}`

command. The following styles are allowed:

`hangindent` is the standard LaTeX style.

`hangindent*` is the same as `hangindent`, but ragged right.

`parindent` sections are indented on `\parindent` without hanged number.

`parindent*` is the same as `parindent`, but ragged right.

`center` section headers are centered.

The `\indentaftersection` and `\noindentaftersection` commands control indentation after sections. The **nccsect** package sets style and indentation to standard LaTeX defaults, `\sectionstyle{hangindent}` and `\noindentaftersection`. But the **ncclatex** package uses `\sectionstyle{hangindent*}`.

The **nccsect** package essentially improves the management of section number tags, running heads, and writing to aux-file:

- If you want to suppress generation of running head in a section markup command, type `\norunninghead` before it.
- If you want to replace the text of running head to another one, type `\runninghead{`*text*`}` before a section.
- If you want to suppress a number in a section or in a caption of float, type `\noheadingtag` before the section or caption markup command. Note that the text of used markup command goes to the running head and to the aux-file. This is the easier way to produce a section without number which must appear in the table of contents and in the header.
- If you want to replace a number in a section or caption to something else, type the `\headingtag{`*other number*`}` before the section or caption markup command. Note that the replaced number goes to the running head and to the aux-file. It must contain robust commands only. Fragile commands in it must be protected with the `\protect` command. The associated counter is not incremented in this case.
- If you want entirely replace a number in a section or caption (including prefix and suffix automatically inserted by section and caption markup commands) to something else, type `\headingtag*{`*other number*`}` before the section or caption markup command. In this case, the writing to running head and to aux-file is suppressed.
- If you want to suppress writing text of a section or caption to aux-file, type the `\skipwritingtoaux` command before the section or caption markup command.

2.1.2. Captions

The `\caption` command useful in floats is also redefined in the **nccsect** package. The caption creation method internally differs from the standard method. You can use line breaking commands in it, but be sure that line break in caption does not go to the aux-file (use an optional parameter with caption to define an argument going to the aux-file). The `\centering` command can be used in captions without worrying (it is automatically suppressed while writing to aux-file). The `\caption*` command is allowed to produce a caption without number tag (the words 'Figure' or 'Table' belong to the tag!). A text of such a caption does not go to the aux-file.

The package defines *different* caption commands for tables and figures. A table caption usually goes above a table. So, a vertical skip after table caption is necessary, but a skip before table caption is unnecessary. A figure caption usually goes below a figure. In this case vertical skips are different: skip before is needed, but skip after is unnecessary. This is the reason why table and figure captions are prepared differently.

2.1.3. TOC-entries

The management of toc-, lot- and lof-entries is also redefined in the **nccsect** package. Every toc-entry is specified with 2 parameters: a left margin and a value of hang indentation. The **nccsect** package calculates the left margin for a toc-entry by summation of margin skips defined in toc-entries of lower levels. For example, if a toc-entry of 2nd level is typed out, its left margin will be a sum of margin skips defined in toc-entries of 0th and 1st levels. This is much more flexible way than the standard one in which the left margin was coded just in the toc-command.

The margin skip and hang indent skip are determined using prototyping technique. This method provides very easy way for redeclaring toc-entries. For example, if your book contains not more than 9 sections, and more than 9 subsections in some sections, you can redeclare the subsection toc-entry as follows:

```
\DeclareTOCEntry{2}{}{}{9.99}{}[9.99]
```

This command contains the following parameters: the toc-entry level, an action applied before toc-entry (you can insert a vertical skip for example), a prefix before a toc-entry number (a section sign for example), a number prototype for calculation of hang indentation, and a style applied to the toc-entry. The 6th optional parameter in square brackets is used for calculation of margin skip for the next toc-level. If it is omitted, the skip is equal to the hang indentation value of this toc-entry.

2.1.4. Customization commands

In conclusion, we show customization commands with their default values for the **ncclatex** package:

`\SectionTagSuffix{.\hskip .6em}` is a suffix inserted after a section number tag (excluding sections of 0th level). Numbers in sections in NCC-LaTeX are ended with the decimal point.

`\CaptionTagSuffix{.\hskip .6em plus .2em minus .1em}` is a suffix inserted after a caption number tag.

`\NumberlineSuffix{.\hskip .6em}{.\hskip .4em}` describes 2 suffices: the first one is used in calculation of hang indentation of toc-entries and the last one is inserted after a number in toc-entries. The last suffix is usually narrower that the first one. This allows to decrease a distance between number and text if the width of number exceed the prototype width just a little. Note: the `\numberline` command from standard LaTeX is modified in the **nccsect** package to prevent an overlapping of a text on a number if a number is wider than hung indentation value.

`\PnumPrototype{99}` is a prototype of page number used for calculation of right margin in toc. It is supposed that a book contains not more than 99 pages by default. To change the page number width to 3-digit pattern, use the `\PnumPrototype{999}` command.

2.2. Theorems in NCC-LaTeX

The **ncclatex** package defines a number of math statements using the **nccthm** package from the *ncctools* bungle. A brief introduction is needed to explain the possibilities of the **nccthm** package. In this section we call all math statements as *theorems* for simplicity.

2.2.1. Overview

In mathematical manuscripts, two modes of theorems numbering are useful. In ordinary mode, a number goes after the theorem title (e.g., **Theorem 1**, **Example 1**, **Theorem 2**, etc.). In this mode, theorems with different titles are usually numbered independently. Another mode requires a number going before a theorem title. We call this mode the *apar* mode. In apar mode, theorems with different titles are numbered sequentially (e.g., **1 Theorem**, **2 Example**, **3 Theorem**, etc.). Both modes can be mixed together in one manuscript.

Theorems with different titles can be shown differently. For example, a Theorem statement is shown with bold header and italic body (a comment after header is prepared with normal font). But an Example statement can be shown with italic header and normal body. We call a presentation of a theorem by the *theorem type*. The **nccthm** package provides two default theorem types named `theorem` and `remark`. New theorem types can be created and existing theorem types can be redefined.

The next parameter of theorems is the indentation style. A theorem can open a new paragraph without indentation (default behaviour) and with paragraph indentation. A theorem number in the apar mode can be shown on margins. The indentation style is controlled via options `indent`, `noindent`, `margin`, and `nomargin` of the **nccthm** package.

The last parameter of theorems is the break style. A line break after theorem header (comment included) can be used. The break and no break style can be coded in theorems while the definition of them using switches `\TheoremBreakStyle` and `\TheoremNoBreakStyle` before the definition or redefinition of theorems (in contrast to standard LaTeX, the `\renewtheorem` command is provided for redefinition of theorems). The break style of an individual theorem can be changed on the fly with the commands `\breakafterheader` and `\nobreakafterheader` inserted before a theorem.

All described properties of theorems are provided with the **nccthm** package. Moreover, the counters used in theorems are declared as dynamic counters (see the **dcounter** package from *ncctools*). A dynamic counter is created at the first use. So, a space of TeX count registers is not occupied with useless counters (this is too important because a number of counters is limited). Using dynamic counters a package writer can prepare some predefined theorems without worry. Since a dynamic counter is created at the first use, a user can control the numbering style for it. This is provided with the `\countstyle{`*`base-counter`*`}` command. The *`base-counter`* is the name of counter a newly created dynamic counter must be subordinated to. An empty parameter means no subordination. So, the change of numbering style for all theorems in a manuscript is executed with one command only! If some counters must have a specific count style, you can create them manually or using the `\countstyle` command with an optional parameter,

```
\countstyle[list-of-counters]{base-counter}
```

This command creates new counters from the comma separated list and subordinates all counters in the list to the *`base-counter`*. It can be used for existing counters also if you want to *change the subordination* of them to another counter. For example, using

```
\countstyle[section]{}
```

in a book we reject the subordination of the `section` counter to the `chapter` counter!

2.2.2. Predefined theorems

The **ncclatex** package defines 8 theorems in 3 forms each: theorems with automatic numbering, theorems with manual numbering, and theorems in the *apar* mode. All the theorems can be used in two ways: as environments and as commands. In command style they must be concluded with another theorem, or `\qef`, or `\qed` command. The `\qef` command finishes a paragraph, inserts a proper vertical space, and returns the font shape to the normal font. The `\qed` command usually finishes the proof of theorem. It prints a qed symbol (white square) adjusted to the right margin and calls the `\qef`. The `\qed*` command prints the qed symbol only.

2.2.3. Theorems with automatic numbering

We show the using of these theorems in the environment style.

```
\begin{theorem}[Comment]       body  \end{theorem}
\begin{lemma}[Comment]         body  \end{lemma}
\begin{proposition}[Comment]   body  \end{proposition}
\begin{corollary}[Comment]     body  \end{corollary}
\begin{definition}[Comment]    body  \end{definition}
\begin{statement}[Comment]     body  \end{statement}
\begin{example}[Comment]       body  \end{example}
\begin{remark}[Comment]        body  \end{remark}
```

In environment style, the end of theorem is equivalent to the `\qef` command. Counter names coincide with the names of environments. First 4 environments are prepared using the `theorem` type and last 4 environments are prepared using the `remark` type.

2.2.4. Theorems with manual numbering

We show the using of these theorems in the environment style also.

```
\begin{Theorem}{Number}[Comment]       body  \end{Theorem}
\begin{Lemma}{Number}[Comment]         body  \end{Lemma}
\begin{Proposition}{Number}[Comment]   body  \end{Proposition}
\begin{Corollary}{Number}[Comment]     body  \end{Corollary}
\begin{Definition}{Number}[Comment]    body  \end{Definition}
\begin{Statement}{Number}[Comment]     body  \end{Statement}
\begin{Example}{Number}[Comment]       body  \end{Example}
\begin{Remark}{Number}[Comment]        body  \end{Remark}
```

If the *Number* parameter is empty, a theorem without number is prepared. These commands do not use counters at all. The presentation types for these theorems are the same as for corresponding theorems with automatic numbering.

Every theorem type has the corresponding `\like`*type* command used for preparing theorems of this type. For example, to prepare a math statement in the `theorem` type, write down the following:

```
\liketheorem{Title}{Number}[Comment]  body  \qef
```

To prepare a math statement in the `remark` type, typeset

```
\likeremark{Title}{Number}[Comment]  body  \qef
```

2.2.5. Theorems in apar mode

We show the using of these theorems in the command style.

```
\atheorem[Comment]       body  \qef
\alemma[Comment]         body  \qef
\aproposition[Comment]   body  \qef
\acorollary[Comment]     body  \qef
\adefinition[Comment]    body  \qef
\astatement[Comment]     body  \qef
\anexample[Comment]      body  \qef
\aremark[Comment]        body  \qef
```

These theorems are number with the `apar` counter. The presentation types for these theorems are the same as for corresponding theorems with automatic numbering. The star-forms of `\liketheorem` and `\likeremark` commands are used for preparing apar variants of math statements. They have no number parameter:

```
\liketheorem*{Title}[Comment]  body  \qef
\likeremark*{Title}[Comment]   body  \qef
```

One additional command, the `\apar`, can be used in the apar mode. Its syntax is the following:

```
\apar[Header]
```

It starts a new paragraph numbered with subsequent value of the `apar` counter. The header is prepared in bold font.

2.2.6. Proof command and support of different qed symbols

The `\proof[Comment]` opens the proof of a math statement. For example,

```
\proof[of Theorem 1] Some text \qed
```

looks as follows:

Proof of Theorem 1. Some text □

Along with the `\qed`, the `\qedsymbol` command can be used in the case when a proof finishes with the display formula. It is typed with the $\mathcal{AMS}$ `\tag*{\qedsymbol}` command within a display formula.

The **nccthm** package provides now two variants of the qed symbol, white and black. They are requested through package options `whiteqed` and `blackqed` respectively. When an option is used, two more commands are created for it: `\`*`option`* and `\`*`option`*`symbol`. For example, the `whiteqed` option generates the `\whiteqed` and `\whiteqedsymbol` commands, and the `blackqed` option generates the `\blackqed` and `\blackqedsymbol` commands. The later option in the list of options describes the default qed symbol. If no options used, only `\qed` and `\qedsymbol` commands are generated.

Other qed symbols are welcome!

2.2.7. Customization commands

In conclusion, we show customization commands with their default values for the **ncclatex** package:

`\TheoremCommentDelimiters{}{}` describes delimiters inserted before and after a theorem comment. In the **nccthm** package the parenthesis are used, but in the **ncclatex** package the delimiters are empty.

`\AfterTheoremHeaderChar{.}` describes the character inserted at the end of header of theorem and proof.

`\AfterTheoremHeaderSkip{\hskip .7em plus .2em minus .1em}` describes a horizontal skip inserted after a header of theorem and proof.

`\AparStyleParameters{\bfseries}{\bfseries}{.\enskip}` describes style parameters for preparing apar theorems: the first parameter is a style used in the `\apar` command, the second and third parameters describe the prefix and suffix inserted before and after `\theapar` command in typeout.

`\ProofStyleParameters{\bfseries}{\proofname}` describes style parameters used in the `\proof` command: the first is a font style and the last is a proof title (it is coded here in the `\proofname` command with the default definition `\newcommand\proofname{Proof}`).

2.3. Math extension

In old NCC-LaTeX, the standard LaTeX environments were used for preparing display formulas and some extension commands were developed to support manual numbering. In the new release of NCC-LaTeX, it was decided to use an excellent math extension provided with the **amsmath** package and enlarge it a bit. The enlargement of $\mathcal{AMS}$ package is provided with the **nccmath** package. The **nccmath** package passes all options to the **amsmath**.

Two major enlargements were done:

- In **amsmath**, the `eqnarray` environment leaves unchanged because alternative $\mathcal{AMS}$ environments exist. We redefine the `eqnarray` to work in the $\mathcal{AMS}$ style. The following improvements were done in it: an equation tag is prepared by the same manner as in $\mathcal{AMS}$ display formulas (`\tag` and `\tag*` are allowed); the `\displaybreak` command is allowed; the intercolumn distance is reduced to the distance between ordinary and relational math symbols; and the center field is prepared in the `\displaystyle`.
- In old NCC-LaTeX, a display equivalent for the `array` environment, namely `darray`, was implemented. In the new release, it is implemented also but in a bit different way to avoid conflicts with packages that redefine the `array` environment. The use of column specifications in the `darray` environment is restricted to `l`, `c`, `r`, `@`, and `*` commands. The intercolumn distance is just the same as for `eqnarray` environment and no space is inserted before the first and after the last column.

2.3.1. NCC-LaTeX equivalents to display formulas

The following NCC-LaTeX equivalents are provided with **nccmath**:

`\eq{`*formula*`}` = `\begin{equation}` *formula* `\end{equation}`.

`\eq*{`*formula*`}` = `\begin{equation*}` *formula* `\end{equation*}`.

`\eqs{`*formulas*`}` = `\begin{eqnarray}` *formulas* `\end{eqnarray}`.

`\eqs*{`*formulas*`}` = `\begin{eqnarray*}` *formulas* `\end{eqnarray*}`.

`\eqalign{`*formulas*`}` = `\begin{equation} \begin{darray}{rcl}` *formulas* `\end{darray} \end{equation}`.

`\eqalign*{`*formulas*`}` = `\begin{equation*} \begin{darray}{rcl}` *formulas* `\end{darray} \end{equation*}`.

The `\eqs` and `\eqs*` commands have an optional parameter describing a distance between columns. For example,

```
\eqs[0mm]{&& -\Delta u = f, \\ && u|_\Gamma = 0,}
```

removes the intercolumn distance because the 3rd column is only used here. The `eqnarray` environment has no optional parameter.

The `\eqalign` and `\eqalign*` commands have an optional parameter also. Its meaning is the column specification parameter: `\eqalign{`*formulas*`}` = `\eqalign[rcl]{`*formulas*`}`.

2.3.2. Remarks on display equations

The alignment style in display equations (except TEX's low level `$$`) can be changed using the following environments:

```
\begin{fleqn}[margin] body \end{fleqn}
\begin{ceqn} body \end{ceqn}
```

The `fleqn` environment sets the flush left alignment with the left margin specified by the optional parameter (default margin is `0pt`). The `ceqn` environment sets the centered alignment.

The `\intertext` command from **amsmath** now has an optional parameter,

```
\intertext[skip]{text between formulas}
```

The *skip* is a vertical skip value inserted before and after the text. If it is omitted, the standard spaces are inserted.

In LATEX, it is not recommended to start a display formula from a new paragraph (in vertical mode in other words). This is because the TEX's algorithm inserts an empty paragraph before such a formula. To avoid insertion of an empty paragraph, Donald Knuth recommended to use `\noindent` command before a display formula starting in the vertical mode. This trick is used in the **nccmath** package. All display environments are slightly corrected with insertion of `\NCC@ignorepar` command at their beginning. This command does a little more than simply applies this trick. It provides suppressing of the vertical space before a display formula if it opens a minipage. It also supports the use of short skip above a display environment by user's request.

In TEX, two types of skips above display formulas are used: the normal skip defined in the `\abovedisplayskip` register and the short skip defined in the `\abovedisplayshortskip` register. When a display formula is typed out, TEX decides what skip to insert depending on width of the formula, its style (centered or flushed left, numbered left or right), and the width of the rest of text in the last line of the previous paragraph. But this algorithm works for ordinary formulas only. It does not work in multiline formulas prepared with `\halign` command. So, a manual replacement of normal skip to short skip is required in some cases. To provides this, the `\useshortskip` command is introduced in the **nccmath** package. It forces the use of short skip in a display formula going after it.

2.3.3. Other math commands

The following commands were ported from the old NCC-LATEX:

`\nr` has just the same syntax as the `\\` but inserts an extra vertical space of `0.5ex`. It is quite useful in display formulas to separate some rows a bit more.

`\mrel{`*rows*`}` prepares a new math relation symbol from one-column table of math formulas. For example, the command `$\mrel{<\\[-.7ex]>}$` produces $\lesssim$.

`\underrel{`*base*`}{`*bottom*`}` is a twin to `\overrel`. For example,
`$A\underrel{\longrightarrow}{x\to 0}B$` produces $A \underrel{\longrightarrow}{x\to 0} B$.

2.4. Figures and tables in NCC-LaTeX

The standard LaTeX floating environments, namely `figure` and `table`, allow user to place floating material in a document. But they do not introduce a style in which this material must be prepared. In NCC-LaTeX, envelop commands are developed which join a style with a float and introduce more features. These commands are implemented in the **nccfloats** package from the *ncctools* bungle.

2.4.1. Basic commands

The `\FloatingStyle{`*style*`}` command sets a style of floats in the document. It affects on the material prepared with commands described below. The default style is

```
\FloatingStyle{\footnotesize\centering}
```

We start with the basic commands, namely `\minifig` and `\minitabl`. They prepare a material in a minipage and allow using the `\caption` command in the body. Their syntax is similar to the `\parbox` command:

```
\minifig[pos][height][inner-pos]{width}{body}
\minitabl[pos][height][inner-pos]{width}{body}
```

The *pos* is a vertical alignment parameter for minipage (`t`, `b`, or `c`) with respect to surrounding text; the *height* is a minipage height required; the *inner-pos* is a vertical alignment of text inside the minipage (`t`, `b`, `c`, or `s`); and the *width* is the minipage width. The *body* is prepared in the specified style and can contain a `\caption` command.

All other NCC-LaTeX floating extension commands are based on `\minifig` and `\minitabl`.

2.4.2. Side figure or table

For small figures and tables, it is preferable to insert them inside a text instead of using floating mechanism. The typographic rules usually require an illustrative material to occupy an outer side of paper. In two side printing, this means figure and tables should be on the right side if a page number is odd and on the left side if page number is even. In one side printing, figures and tables must occupy the right side of paper.

The following commands support such a placement:

```
\sidefig[pos](w1)(w2){figure body}{text body}
\sidefig*[pos](w1)(w2){figure body}{text body}
\sidetabl[pos](w1)(w2){table body}{text body}
\sidetabl*[pos](w1)(w2){table body}{text body}
```

For simplicity, we further use the term *minifloat* for the small illustrating material (figure or table), however understanding that it is not a float at all. It is inserted in the main flow next to a paragraph box specified in the last parameter of above described commands.

The no-star forms of above described commands place a minifloat next to the specified text on the outer side of page (to the right for odd page and to the left for even page). In two column or one side mode, minifloat is always posed to the right. The star forms provide the reverse placement. By default, minifloat is vertically centered with respect to the text and the `\strut` command is inserted at the beginning and at the end of the *text body* to provide normal baseline distances of the first and last lines of the text from surrounding text lines.

All parameters in square and round brackets are optional and mean the following:

pos specifies minifloat alignment (`t`, `b`, or `c`; default is `c`) with respect to text box and can contain additional chars controlling the text body preparation: `j` means the last line of the text to be justified to the right and `n` means suppressing of struts insertion (they should be inserted manually if necessary);

w_1 is the width of minifloat (if units are omitted, the `\unitlength` is supposed); and

w_2 is the width of the text box (if units are omitted, the `\unitlength` is supposed).

If both width parameters are absent, the widths are calculated as `(\linewidth-1.5em)/2`. If w_2 is absent, the text body width is calculated as `\linewidth-`w_1`-1.5em`.

While preparing a side minifloat, it is sometimes necessary to provide conditional placement depending on the side a minifloat is posed. The command

`\ifleftsidefloat{`*left-clause*`}{`*right-clause*`}`

provides this. It is useful in parameters of `\sidefig` or `\sidetabl` and processes *left-clause* if the minifloat is posed to the left or *right-clause* otherwise.

2.4.3. Floating figure or table

`\fig[`*placement*`](`w`){`*body*`}`
`\fig*[`*placement*`](`w`){`*body*`}`
`\tabl[`*placement*`](`w`){`*body*`}`
`\tabl*[`*placement*`](`w`){`*body*`}`

The *placement* is a float placement parameter describing places where a float can appear. The default value is `htp` (here, or at the top of the next page, or on the page with floats only). The optional `(`w`)` parameter defines a width of box occupied by the float (the width of nested `\minifig` or `\minitabl`). If it is omitted, the float has the maximum width equal to `\linewidth`.

The `\fig` and `\tabl` commands envelop the `figure` and `table` environments respectively. Their star-forms envelop corresponding starred environments.

2.4.4. Two floating figures or tables side by side

`\figs[`*placement*`](`w_1`)(`w_2`){`*body1*`}{`*body2*`}`
`\figs*[`*placement*`](`w_1`)(`w_2`){`*body1*`}{`*body2*`}`
`\tabls[`*placement*`](`w_1`)(`w_2`){`*body1*`}{`*body2*`}`
`\tabls*[`*placement*`](`w_1`)(`w_2`){`*body1*`}{`*body2*`}`

These commands place two figures or tables side by side. The *body1* is a body of the left figure or table and the *body2* is a body of the right figure or table. Other parameters are optional. The meaning and default value of the *placement* parameter is the same as described above. The `(`w_1`)` and `(`w_2`)` parameters are widths of left and right boxes. If they both are omitted, the left and right boxes will have the width equal to `(\linewidth-1em)/2`. If `(`w_2`)` is omitted, the right box will occupy the rest of horizontal space minus `1em`. If both parameters are specified, the rest space is inserted between boxes. If the total width of left and right floats exceeds the `\linewidth`, the floats will overlap at the middle of line (a negative horizontal space is inserted between them).

In `\tabls` command, boxes of the left and right bodies are top-aligned, but in `\figs` command, the bottom alignment is used.

2.5. Graphics in NCC-LaTeX

The **nccpic** package from the *ncctools* bungle envelops the standard **graphicx** package. The package passes all options to the underground package.

The main aim of the **nccpic** package is the preparing of graphics extensions list depending on a dvi-driver used with the **graphicx** package. This allows omit an extension of a graphics file in the `\includegraphics` command. When a file without extension is searched, this command sequentially tries extensions from the list until an appropriate file will be found.

Using this feature you can support multiple output from LaTeX with minimum changes in `.tex` sources. The only required thing is to prepare a number of versions for all graphics files called in a document. For example, the dvips program likes `.eps` files, the Yap previewer likes `.bmp` or `.eps` files, the pdflatex likes `.png` files. To satisfy their needs, you can prepare `.eps`, `.bmp`, and `.png` versions for all pictures and pass a required option to the **nccpic** package.

The graphics extensions list is customized now for most popular drivers, namely `dvips`, `dvipdf`, and `dvipdfm`, and for use with `pdftex` also. More customization is welcome!

The next aim of the **nccpic** package is the regulation of placement of graphics files in the file system. It is too inconvenient when pictures are stored together with `.tex` source files. We propose to store graphics files in the `graphics` subdirectory of the source directory. To support the search in this storage, the graphics path is customized in the **nccpic** package.

The following commands are introduced in the **nccpic** package:

`\ipic{`*`filename`*`}` command loads a file having the name *`filename`*`.pic` searching it in the graphics path. It allow to take actual graphics inclusion commands out of source files for automation purposes.

`\putimage(`x, y`)[`x_r, y_r`](`x_s, y_s`){`*`filename`*`}` command emulates a graphics inclusion technique used in the old NCC-LaTeX. Look in the **nccpic** package file for more detail description.

`\draftgraphics` and `\finalgraphics` are toggles between final and draft modes for preparation of graphics. In draft mode, a graphics object is shown as a framebox of the required dimensions with an object name in it. Using these commands, you can toggle graphics preparation mode on the fly. In draft mode, the `\putimage` command does not test an existence of required graphics file.

2.6. Additional boxes

The **nccboxes** package from the *ncctools* bungle provides additional boxes:

`\jhbox{`*`prototype`*`}[`*`align`*`]{`*`body`*`}` horizontally aligns the body in the text box which width is defined by the *`prototype`* parameter. The optional *`align`* parameter defines a position of alignment (`l`, `c`, `r`, or `s`; default is `c`).

`\jvbox{`*`prototype`*`}[`*`align`*`]{`*`body`*`}` vertically aligns the body with respect to the strut defined by the *`prototype`* parameter. The optional *`align`* parameter defines a position of alignment (`t`, `c`, or `b`; default is `c`). If `t` is used, the body is raised in such a way that its height will be equal to the height of the prototype's strut. For `b` case, the depths will be equal and, for `c` case, the body is vertically centered with respect to prototype's strut.

`\jparbox{`*`prototype`*`}[`*`align`*`]{`*`width`*`}{`*`body`*`}` prepares a paragraph box of required *`width`* and vertically aligns it with respect to the *`prototype`* by the same manner as `\jvbox`.

`\pbox[`*`align`*`]{`*`body`*`}` is a simple one-column table. The *`align`* parameter may consist of two letters defining a relative alignment in the column (`l`, `c`, or `r`) and the vertical alignment of the table with respect to surrounding text (`t`, `c`, or `b`). Centering is the default alignment. The distance between the table rows is independent on the `\arraystretch` value.

`\addbox{`*above*`}{`*below*`}{`*body*`}` prepares an hbox containing the *body* which height is adjusted to the value of the *above* length parameter and depth is adjusted to the value of the *below* length parameter. For example, `\addbox{.5ex}{.5ex}{text}` increases the height and depth of produced hbox on `0.5ex`.

`\picbox{`*body*`}` = `\begin{picture}(0,0)(0,0)` *body* `\end{picture}`.

To prepare a fancy table, the following commands can be used:

`\Strut/`*value*`/` is a strut which height and depth are calculated from the strut prototype (letter `A` by default) as follows: if *value*>0, the full height of the current `\strutbox` multiplied by the *value* is added to the height of prototype strut, otherwise the depth of prototype strut is increased with the modulus of *value* multiplied by the full height of `\strutbox`. For example, `\Strut/1/` inserts a strut which height exceeds the height of the letter `A` from the current font on the interline distance. The `\Strut` without parameter is equal to `\Strut/0/`.

`\tstrut`, `\bstrut`, and `\tbstrut` insert struts exceeding the height, depth, and height and depth of the strut prototype by a special small amount. This amount is calculated in such a way that the full height of `\tbstrut` will be equal to 1.5 of full height of the current `\strutbox`.

`\cbox/`*value*`/[`*align*`]{`*body*`}` prepares a box whose body is a one-column table. Its height and depth are enlarged using `\tstrut` at the beginning and `\bstrut` at the end of body. The horizontal alignment (`l`, `c`, or `r`) in the column and the vertical alignment (`t`, `c`, or `b`) are defined in the *align* parameter. Centered alignment is used by default. The resulting box is vertically aligned with respect to the `\Strut/`*value*`/` using the `\jvbox` command. The `\cbox*` command does the same but vanishes the height and depth of the resulting box.

The `\cbox` command is used in the headers of tables. Its star form is useful in cells having vertical spans. The style for typing its body is managed with the `\cboxstyle` command. Its default value is empty but in NCC-LaTeX this command is redefined as follows:

```
\renewcommand\cboxstyle{\scriptsize}
```

We demonstrate the use of struts and `\cbox` on the example

Vertically spanned head	**Simple head**	**Very long head of two lines**	
	subhead	**subhead**	**subhead**
Text	field	field	field
Text	field	field	field
Text	field	field	field

produced by the following code:

```
\begin{center}
\renewcommand\cboxstyle{\small\bf}
\setlength{\tabcolsep}{10pt}
\begin{tabular}{|l|c|c|c|}\hline
  \cbox*/-1.5/{Vertically\\spanned head} & \cbox{Simple head}
  &\multicolumn2{c|}{\cbox{Very long head\\of two lines}}\\\cline{2-4}
  &\cbox{subhead} &\cbox{subhead} &\cbox{subhead}\\\hline
  \Strut/1/ Text & field & field & field \\
            Text & field & field & field \\
  \bstrut   Text & field & field & field \\\hline
\end{tabular}
\end{center}
```

2.7. Miscellaneous commands

We describe here commands provided by the **ncclatex** package itself.

`\acknow` starts a new paragraph with the *Acknowledgements* title.

`\cref{`*`citelabel`*`}` produces a citation without brackets to the specified *citelabel*. It is used in the case of citations containing ranges, e.g. [2–4, 7].

`\mop{`*`name`*`}` is used for coding new math operations like sin. For example, `$k=\mop{sgn}x$` produces $k = \operatorname{sgn} x$ with proper font and spaces. This command is equivalent to $\mathcal{AMS}$ `\operatorname` command.

`\tg`, `\ctg`, `\arctg`, and `\arcctg` produce math operations for tangent, cotangent, and for their inverse functions in Russian typesetting tradition.

`\No` produces the numero symbol. If the `russian` option is not used, this command emulates the numero symbol with the text "No.".

`\tc{`*`body`*`}` adjust the body to the center using `\hspace*{\fill}` glues. This command is useful in tables when centered alignment of a particular cell is required but another alignment is specified for the column the cell belongs to.

The titles of proof, predefined theorems, and acknowledgements are declared with commands having the `name` suffix. For example, the proof title is coded in the `\proofname`, the title of lemmas is defined in the `\lemmaname`, and the acknowledgements title is coded in the `\acknowname` command. They all can be redefined to provide internationalization (the `russian` option redefines these titles in Russian).

3. The sibjnm class

This class describes a style used in the *Siberian Journal of Numerical Mathematics* (SibJNM), `http://www.sscc.ru/SibJNM/`, email: `sibjnm@oapmg.sscc.ru`. We recommend to use this class for submitting an article to this journal.

The **sibjnm** class is based on the **ncc** class. It sets the following options of **ncc** class: `a4paper`, `11pt`, `article`, `twoside`, and `onecolumn`. A user can manage the following options: `draft`, `final`, `openany`, `openright`, and `russian`. Other options are forbidden.

The SibJNM is a bilingual journal. An article can be prepared in English or in Russian. An abstract is prepared in both languages. Before abstracts, one of subject classification commands must be used: `\UDC{`*`indices`*`}` or `\AmSclassification{`*`indices`*`}`. The first command sets UDC classification indices and the last command sets $\mathcal{AMS}$ classification indices of the article subject.

The russian abstract must go right after the `\maketitle` command. Its syntax:

```
\begin{Rabstract}{Authors}{Title} body \end{Rabstract}
```

The english abstract must go after the russian one. Its syntax is similar:

```
\begin{Eabstract}{Authors}{Title} body \end{Eabstract}
```

The text field of SibJNM article is 155×225 mm.