AS$\TeX$3.0β

A Package for Typesetting Textbooks, Journals and Conference Proceedings

Alex Kasman

Table of Contents

1 Introduction **1**
1.1 Notation *1*
2 Installation **2**
2.1 Getting the Files *2*
2.2 What to Do With the Files *3*
3 Using ASTEX **3**
4 Your Format File **4**
4.1 How To Set Values in Your Format File *5*
4.2 Font Selection *6*
4.3 Page Layout *9*
4.4 Parameters for Sectioning Commands *11*
4.5 The defs File *13*
5 Articles or Textbook? **14**
6 Commands for Typesetting Articles **14**
6.1 File Structure for Articles *14*
6.2 Commands for the beginning of an article *15*
6.3 Bibliography *16*
6.4 Claims *18*
6.5 InfoBox *21*
6.6 Submissions from Authors *23*
7 Commands for Typesetting a Textbook **23**
7.1 An Outline of the File Structure of a Textbook *23*
7.2 Setting Up Your Format File For Textbook Production *24*
7.3 Commands for Exercises *28*
7.4 Generating A Solutions Manual *32*
7.5 Making an Index *33*
8 General ASTEX Commands **35**
8.1 Making a Table of Contents *35*
8.2 Auto-Page: Getting the Right Page Numbers *36*
8.3 Labels and References *38*
8.4 Paginating Commands *40*
8.5 Maintaining Different Versions of a Document: `\suffix` *41*
8.6 Flags *42*
8.7 AMS Compatability *43*
8.8 Miscellaneous Commands and Hacks *43*
9 Paper Marks **46**
10 Displaying Figures, Tables and Things **48**
10.1 ASTEX Float Environment: `floatfig` *48*

10.2 Putting things in the margin *50*
10.3 `\finput`, `\tinput` *and Captions* *51*
10.4 Aligning Pictures and Tables *53*
11 Graphics Commands **56**
11.1 The ASTEX Picture Environment *56*
11.2 Commands for Inside an `aspic` *Environment: Labelling a Figure* *58*
11.3 Flags to Help with Labelling *62*
11.4 Examples of Graphics Commands *62*
12 Postscript **66**
12.1 Using ASTEX Postscript Commands in Your Graphics *67*
12.2 Fixed Line Weights and Shades *67*
12.3 Colors and Color Separation *68*
13 Appendices **70**
A External Shell Scripts *70*
B For Users of `newcalcstyle` *and* `intlpress` *72*
C Instructions for Authors of Journal and Proceedings Submissions *74*
D Examples of Articles in ASTEX *76*
E Counters *81*
F Trouble Shooting *82*
14 Index **83**

ASTEX

Alex Kasman
Department of Mathematics
Boston University

1 Introduction

ASTEX is a LATEX style file designed for the production of professional textbooks, journals and conference proceedings. (The intended pronounciation is such that it would be a homonym of "Aztec".) It is intended to combine the best qualities of the various style files which I previously created for these purposes, including `newcalcstyle.sty`, and `intlpress.sty`. Because of the diverse applications this package may find, it has been designed with a "modular" structure in which you will be expected to choose and design your own format.

Although ASTEX has been designed to be compatible with LATEX2e, the new version which was recently released, it does not take advantage of many of that system's nicer features. This is because, at this point, most people who want to use ASTEX do not have LATEX2e and because I do not currently have the time to make these modifications. It should not be difficult for me or someone else to modify it at a later date.

That having been said, no one may modify ASTEX and then distribute it to others without my explicit permission. Of course, you may modify your own version. In fact, I'm sure you will get better use out of it if you do! I suggest, however, that all of your changes be done in your format or defs file (see below) without changing the file `astex.sty`. In this way, you can easily "upgrade" to newer versions merely by getting a new copy of `astex.sty`.

Although it was written for a few specific "clients" of mine, I hope that others find ASTEX to be useful. My effort in creating this package was supported (financially) by the Calculus Project at Harvard, International Press, and the ODE Project (Paul Blanchard, Dick Hall and Bob Devaney) at Boston University. Anyone else may use the package and create their own format files. I do ask, however, that this package and my efforts be acknowledged in any document which uses it. That is, please include a line such as "Typeset using the LATEX package 'ASTEX' by Alex Kasman." (I wouldn't complain about a complimentary copy of your document, as well, to add to my growing library of math and physics books.)

1.1 Notation

It is assumed that you already know about TEX and LATEX prior to reading these instructions. If not, there already exist many books at various levels which discuss these subjects.

When a new TEX object is introduced within these instructions, it will be done in the manner indicated below:

`\commands` (❖) will be indicated and described in this manner. Commands are TEX macros which you call to do something. Some commands (such as `\par` which starts a new paragraph) need no additional information. Other commands require an *argument* – that is, some *input* for the command. For example, the command `\stepcounter` will add one to a counter, but it needs to know what counter to step! So, to implement the command in a file, you would have to give it an argument such as "`chapter`": `\stepcounter{chapter}`. Some commands have *optional* arguments, which means the command works even if you do not provide the argument.

`environments` (`\begin-\end`) will be indicated and described in this manner. Environments, you will recall, are LaTeX commands which are called by using the `\begin` and `\end` commands. Environments act on the text which is placed between the `\begin` and `\end`. For example,

```
\begin{center}
Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Text Text
\end{center}
```

will center the text on the page, rather than running it from the left margin to the right as usual. Some environments require arguments (or have optional arguments) which should immediately follow the `\begin` statement. (For example, `\begin{aspic}(0,3)`.)

`\fonts` (✡) will be indicated and described in this manner. By a font, I actually mean a command which *calls* a font. Most of these commands are actually built into other commands, and so you will not necessarily see them. For example, the font `\secfont` is used for section headings in ASTeX – so it will be used automatically any time you use the `\section` command. It is discussed in these instructions because you will have to *choose* a font to be called by the command `\secfont`. The command `\nameasfont` will be used for defining fonts in this manner. All of the fonts are defined in an `as-fonts` file.

`\dimensions` (⟼) will be indicated and described in this manner. A dimension is a variable storing a linear distance. Although they can be described by any unit, dimensions are always stored in *points* by TeX. Again, after you set up the dimensions in your `as-format` file or defs file, you will probably not have to mention them again.

2 Installation

2.1 Getting the Files

The files which are distributed as ASTeX are:

- `astex.cls`: which contains all of the TeX and postscript definitions of ASTeX.
- `astex.sty`: is a short file, for use by people who do not have access to LaTeX2e; it basically just loads `astex.cls`.
- `as-formatBLANK.tex`: is a "blank" format file to help you create your own format file.
- `as-fontsBLANK.tex`: is a "blank" fonts file to help you creat your own ASTeX fonts file.
- `asbar.ps`: is a very simple postscript file (merely a grey bar) which is used in a few of the built in definitions to demonstrate how to use postscript graphics in your own definitions.
- `instructions.ps`: is these instructions as a postscript file.

The only file which you absolutely need to have to use ASTEX is `astex.cls`.

These files can be gotten from the Boston University Math Department's anonymous FTP site or WWW page. Find them on the Web at URL `http://math.bu.edu/INDIVIDUAL/alexk` or by FTP to `math.bu.edu` in the directory `pub/www/INDIVIDUAL/alexk/ASTeX`.

There are also subdirectories containing external programs which work well with ASTEX. In particular, there is `aspell` which is a spell checker, `asrename` which is a program for renaming question and figure files in a textbook, and `PS` which is a package of Mathematica, Unix and postscript commands which convert Mathematica graphics into ASTEX ready postscript. Further information on these packages can be found in the README files in each directory.

2.2 What to Do With the Files

There are certain locations in which TEX expects to look to find a file. On a Unix system, for example, there is an environment variable called `TEXINPUTS` which lists all of the directories in which TEX will look for a file when an input is requested. I will refer to these directories collectively as your `TEXINPUTS` path. All of the files which make up ASTEX should be put into a directory which is in your `TEXINPUTS` path. There is no reason that each person using the system should have their own copy of these files, so they should be placed in a central directory to which everyone who may use it has access (e.g. `/usr/local/lib/tex/macros`).

Other than putting these files in an accessible directory, there is not much you *must* do to set up. As you will see below, you should have a format file to run ASTEX (although a default format is used when no format file is specified). Several sample formats are provided for your use. However, in order to get full use out of this package, you will want to modify these samples or create your own format file. Section 4 will provide you with the information you need to create your own format file.

3 Using ASTEX

ASTEX should be used as a LATEX style file. That is, the first line of your document should be a `\documentstyle` command (or in newer versions of LATEX, `\documentclass`) declaring `astex` as the style. (See Figure 3.1.) This is not sufficient, however, because ASTEX is not as specific a request as is `book.sty` or `article.sty`, the usual LATEX style files with only one use and a predetermined format. It is expected that you will create an ASTEX format file for your own use, which specifies the appearance of the output and also what sort of document (textbook or articles) you are producing. (Creating a format file should not be difficult, and it only needs to be done once.) After the `\documentstyle` statement, you indicate the name of the format file you are using with the command `\asformat`.

Even within the same format, you may want to modify the output temporarily. For example, you might want to produce a draft version in which the names of all of the subsidiary files are printed on the document (see `\showfilename`), or you might want a grid printed over figures to aid in the positioning of labels (see `\showgrid`). This is done via the use of *flags*, which are commands placed at the beginning of the document to alter the output. Since the action a flag takes might be altered by your choice of format, flags should be placed just after the `\asformat` command.

Finally, the actual text of your document will be contained in an `document` environment. That it, between the commands `\begin{document}` and `\end{document}`. So, in summary, a TEXable ASTEXfile looks like Figure 3.1.

Then, assuming that ASTEX has been installed properly, all you need to do to

```
\documentstyle{astex}

\asformat{MY}  %  or some other format

\showfilename  %  \
\verbose       %   +- optional flags such as these should be placed
\printodd      %  /   here, between \asformat and \begin{document}

\begin{document}

% The main body of your text (the chapter, or list of articles)
% goes here.

\end{document}
```

Figure 3.1: A generic ASTEX file

process the file is to run LATEXon it. (On Unix systems this is done merely by typing `latex filename`.)

Note: If you are using the command `\amscompatability` in your format file in order to be able to use commands and symbols from the American Mathematical Society (AMS), you will need to have AMSLATEX installed and process the document with that in place of LATEX.

If TEX detects an error in a document which it is processing, it will halt and show you the commands which it stopped on. Often, it will provide a useful comment as well. It is worthwhile reading the information provided when TEX halts to try to figure out what is wrong. For example, if it halts when trying to load a font, perhaps you don't have that font or you typed its name wrong. Sometimes the problem is the command preceeding the one on which it halts. Sometimes it is even farther away. It can be difficult to debug a TEX file, and ASTEX does not really make it any easier, but trying is the only way to learn.

If LATEX does not have a problem processing your document, but you cannot print or preview the document, the problem might be the postscript commands which ASTEX uses. Read the section on postscript in order to see whether this might be the case and how you can avoid the problem.

4 Your Format File

The format file is what determines the layout and appearance of your document. Although there are sample format files provided for your use, in order to be able to tailor your document to your needs as well as the specifications of your publisher, you will need to create your own format file.

After the `\documentstyle` (or `\documentclass`) statement in your file, specify the format using the command `\asformat`. For example, `\asformat{MY}` specifies the format `MY`. You must then have a file named `as-formatMY.tex`, the format file, containing the information described below. A *blank* format file is provided, under the name `as-formatBLANK.tex`. When you are first setting up your format file it will probably be useful to begin with this blank, because it will point out all of your options and make sure that you set all of the necessary variables. In addition, sample formats are provided for your use.

The format actually comes in three parts. First, the format file itself: `as-formatSOMETHING.tex`. This file will specify a font file, (that will be automatically loaded). Finally, the defs file is loaded. This file, called `as-defsSOMETHING.tex` (where `SOMETHING` is the same as in the format above) is loaded after all ASTEX definitions are loaded. It is a location for you to put your own definitions, or modify ASTEX definitions. (The defs file is optional – you will not get an error message if the file is not present.)

Important Note: If you make your own definitions or macros, you should put them in your defs file and **not** modify the file `astex.sty`. That way, if I make a new version, fix bugs, or add features to ASTEX, I can merely distribute a new version of `astex.sty` and everything will still work.

4.1 How To Set Values in Your Format File

There are four ways which you will be expected to set values within your format, `as-fonts`, and defs files. These instructions and the blank files should make it clear which method you must use for which variables.

TEX Definition

The usual TEX command for defining a command is `\def`. To use this command to define `\infobox` to print three lines of text stating my name and address, I would say:

```
\def\infobox{Alex Kasman\\
Boston University\\
Boston, MA 02215}
```

Although this is all you would need to know in order to be able to define the commands which you must set in your format, the command is much more versatile than just that. Please feel free to read the file `astex.sty` and the other accompanying TEX files to see how I use this command or read the TEXBook for more information.

Using `\let` to Select a Sample

The TEX command `\let` is used to set one macro equal to another. For example, if `\cow` is a command which will print the Declaration of Independence, then the statement `\let\horse=\cow` defines `\horse` to be the same. Thus, the command `\horse` now also prints the Declaration of Independence.

Note: The equal sign (`=`) is optional. `\let\horse\cow` does exactly the same thing.

The reason this will be useful in your format file is that in many cases I have provided a variety of sample definitions for specific commands. In these cases, I have lettered the samples `a`, `b`, `c`.... So, for instance, I have provided several samples for the command `\bibtype` which determines the appearance of bibliography references. The samples are called `\bibtypea`, `\bibtypeb`, etc. To select `\bibtypeb` to be the definition of `\bibtype` then, I would include the line

```
\let\bibtype\bibtypeb
```

Note: There is no reason to expect that the first command must look the same as the second command other than the letter at the end. In fact, one might wish to set the appearance of bibliography references in the bibliography itself to be different than in the text of the article. In this case you would set `\bibitemtype` to be one of `\bibtypea`, `\bibtypeb`, etc.

Then, if you are not satisfied with any of the samples provided, all you need do is use the command `\def` to make a new definition for the command in question. For example, you need not use `\let` to set `\bibtype` to be one of the samples; instead you could define it to be some other type. However, you do need to know *something* about programming TEX in order to be able to do this. A good way to learn would be to look at the definitions of the samples which I provided.

Setting Dimensions

Some variables in TEX are dimensions, which is to say that they store a linear distance such as 1 inch or 10.35 picas. You may specify a dimension in just about any reasonable unit by merely writing the name of the dimension followed by the distance you wish to set. For example, I can set the dimension `\labspace` (which determines the space around a label in a figure) with the line

```
\labspace 1in
```

or

```
\labspace 10.35pc
```

However, regardless of the unit which you specify, TEX will store the dimension in *points*. (One point is approximately $\frac{1}{72}$ of an inch.)

Skips and Stretchy Glue: Actually, it is a little more complicated than that since *some* of the dimensions which you must set in ASTEX are actually *skips*. This means that in addition to a linear distance, there is also information about how much it can stretch or shrink. For more information, you should look in the TEXBook. However, it is easy to show you how this is done. You merely add the words “plus” and “minus” to indicate how much stretchability and shrinkability there is (respectively). For example, the space around a figure in a `stayfig` or `floatfig` environment is determined by the skip `\intextsep`. If you want to set this to be 20 points, with a little bit of shrink and stretch, you say:

```
\intextsep 20pt plus 5pt minus 3pt
```

If you find that your pages are stretching too much at certain skips, edit the skip by lowering the value after the “plus”. (Note: Lowering all of the “plus” values by the same factor is equivalent to doing nothing since stretching is merely determined by the relative “plus” values of each skip on the page.) Similarly, if a skip seems to be shrunk too much in many instances, lower the “minus” part of its value. (You can lower all “minus” values and affect the output.)

Specifying Fonts

Different versions of LATEX have different ways of specifying fonts. In your `as-fonts` file, you can specify your own fonts, the math fonts or the text fonts using whatever commands are appropriate for your version of LATEX. However, the ASTEX fonts (such as those used for section headings (`\secfont`), labels in figures (`\labfont`), and so on, should be specified with the command `\nameasfont` (as described below).

4.2 Font Selection

In addition to the usual text and math fonts which any TEXfile must specify, there are many ASTEXfonts. These fonts are used, for example, in section headings, chapter headings, labels in figures, the table of contents, etc. These fonts will be set using the command `\nameasfont` in a separate file, the `as-font` file. Some font files are supplied for your use, however, it is likely that you will want to modify them or create your own file to suit the specifications of your publisher or your own tastes.

Specifying a Font File

The font file which you wish to use is specified in your format file using the command `\asfonts`. Suppose you wish to use the font file named `as-fontFANCY.tex`. Then you include the line

```
\asfonts{FANCY}
```

in your format file. If no line of this sort is detected, then a default font file is loaded.

The font files which are supplied for your use are:

- bleh
- blah

Editting or Creating a Font File

The ASTEXfonts which must be specified in the `as-fonts` file are not families of fonts, as are the text and math fonts. For example, when a font is specified to be the font used in section headings, only one size and shape of the font is needed. What needs to be specified, then, when creating a font with the command `\nameasfont`, is the name by which you will call the font, the name of the font which you wish to use, and the size you wish the font to be printed. (These will be discussed further below.) After this has been specified, the command will do some calculations to determine two things: the "baselineskip" (the vertical distance between the bottom of two letters in this font if they are on successive lines) and the size of any math characters which will be printed when using this font. (Again, I will discuss these in greater detail below.) However, it may be that you wish to specify these to be something other than what `\nameasfont` chooses. Thus, two optional arguments appear in the definition of this command to allow this. A third optional argument allows you to specify commands to be performed everytime this font is called. As an example of how this third optional argument might be useful, consider the case in which the font which you are naming is a bold font. In that case, you would probably want to include the command `\boldmath` in the third optional argument. That way, if you are using this font and some math is imbedded in it, the math will be printed in bold as well.

Although you may use this command to define fonts for your own use, there are certain fonts which ASTEX will use, and you should certainly define all of these. There is a "blank" font file (called `as-fontsBLANK.tex`) provided with ASTEX in which you can find a list of all of the fonts you should define along with a description of which ASTEX commands use each one.

`\nameasfont` (❖) is used to specify ASTEX fonts in the `as-fonts` file. It creates a command which: calls a font, sets the baselineskip, and will switch to an appropriately sized math font. The syntax is

```
\nameasfont{CMDNAME}{FONTNAME}{SIZE}(BLSKIP)[MATHSIZE]<EXTRA-CMNDS>
```

where the last three arguments are optional. (Any of the optional arguments may be left out, but those that are present must be in the given order.) Here, `CMDNAME` is the name that will call the font, `FONTNAME` is the name of the font (in the naming scheme of your system), `SIZE` is a number representing the height of the font in points (the units are always points and should not be specified by "`pt`"), `BLSKIP` is the baselineskip which will be used along with this font, `MATHSIZE` is a the name of a command (such as `small`, `large` or `huge`) which leads to a math font of a size appropriate for imbedding in text written in `FONTNAME` at size `SIZE` points, and finally `EXTRA-CMNDS` are extra commands which will be executed whenever `\CMDNAME` is called. The default value for `BLSKIP` is 1.2 times `SIZE` points. The default value for `MATHSIZE` is the the standard LATEX sizes (`tiny`, `scriptsize`, …, `Huge`) for which the height of an "x" is closest to the font you selected. The default value for `EXTRA-CMNDS` is simply `\relax`.

For example, the standard name on my system for the Helvetica font which I use for section headings is `rphvr`. Furthermore, the name for the font used in section headings is `\secfont`. (This name is part of ASTEX, as you can learn by looking in the blank font file.) Thus, the command

```
\nameasfont{secfont}{rphvr}{14}
```

will set the font used in section headings to be Helvetica at 14 points. In addition, this command will determine that any math which occurs in a section heading will be typeset in `\LARGE`, since that is the LATEX size for which the height of a printed "x" is closest to the height in Helvetica at 14pt. (In general, this will be the best choice of a math font. However, there are some instances in which you may want to specify a different math font, such as in the case that the font you are defining is an all capitals font – in which case you would certainly want the math font to be much smaller.) The baselineskip will be set to 16.8 points ($= 1.2 \times 14$ points), so that the vertical distance between the bottom of lines in the section heading (if there are more than one) will be 16.8 points. If, however, I wanted the baselineskip to be 20 points, the math to be typeset in `\huge` and the math in section headings to be bold (i.e., printed using `\boldmath`), I would use

```
\nameasfont{secfont}{rphvr}{14}(16.8)[huge]<\boldmath>
```

Note: The fonts which you request must be installed in your system (in the appropriate size) in order for you to use this command. For example, the above command would certainly not work if your system does not have the Helvetica font or if it is not named `rphvr`. It should be noted that at present, two different types of fonts are frequently used with TEX: postscript fonts (such as Helvetica) which can be called at any size, and computer modern fonts (those standardly associated with TEX) which can only be loaded at specific sizes.

It would be a waste of memory and effort to use the command `\nameasfont` to load a font which is already being used for another purpose. For example, if your text is printed in Computer Modern roman at 10 points, then you already have Computer Modern bold at 10pt – specified by the commands `\normalsize\bf`. The font used for printing the folio (the page number at the top of the page) is called `\foliofont`, and you might very well want to use Computer Modern bold at 10 point for this font. Using the syntax of `\nameasfont` above would be wasteful, since it would load the information about the font under a new name even though it already has all of the information about that font. So, an alternate use of `\nameasfont` is available for specifying the font by commands which already exist. This alternate version of the command is accessed by putting an asterisk (*) immediately after the command. In this form, the command takes only two arguments.

`\nameasfont*` (❖) takes two arguments, the first being the name of the ASTEX font being named and the second being a set of pre-existing commands which result in the desired appearance of the text. For instance,

```
\nameasfont*{foliofont}{\normalsize\bf}
```

sets the font for printing the folio to be the bold, normalsized shape of the main text font of the document.

The commands above are ASTEX commands for setting the fonts specific to this package. If you wish to alter your math or text fonts, you should use the LaTeX packages which exist for this purpose. For example, the package `times.sty` can be used to convert the text font to Times-Roman. It is important that these packages be loaded at the *beginning* of the `as-fonts` file, so that math sizes can be determined appropriately.

`\asfontnotify` (❖) is a flag which causes a note to be printed on the screen and in the `.log` file every time an "as-font" is called. That is, once this flag is encountered, any font command `\FONT` which was defined with one of the two forms of `\nameasfont` will notify you that `FONT` has been called.

4.3 Page Layout

The following dimensions must be set to determine the basic layout of the page:

`\paperht` (⟼) is the height of the paper. Note, this is not necessarily the same as the height of the paper on which you are printing. Many books are printed on paper of height 10 inches, for example, whereas most printers use paper which is 11 inches high.

`\paperwd` (⟼) is the width of the paper. (See `\paperht`).

`\fullpgwid` (⟼) is the width of the region on which text will be printed on the page. It *includes* the amount which section headings might hang out to the left (as they do in these instructions).

`\cutpgwid` (⟼) is the width of the region on which text will be printed *excluding* the part on which headings will hang out to the left. It should therefore be less than or equal to `\fullpgwid`. If it is equal to `\fullpgwid`, then the headings will not be able to hang out to the left, and the command `\putinmargin` will not have room to put objects.

`\guttersize` (⟼) is a *new* dimension added in version 3.0. For some books, the publisher specifies the amount of space they want between the "fold" in the book (the gutter) and the text closest to it. So, we now provide this variable. If it is not set (or is equal to zero) then the gutter size is chosen automatically so as to center the text (including hanging section heads) on the page. However, if the variable `\guttersize` is specified in your format file then the image is shifted so that the gutter has exactly the specified dimension.

`\topmargin` (⟼) is the amount of space between the top of the paper and the top of the running head.

`\headheight` (⟼) is the height of the running heads.

`\headsep` (⟼) then is the distance from the running head to the top of the text.

`\textheight` (|↔|) is the height of the text (not counting the running head or the space around it.

So, in particular, the sum of `\topmargin`, `\headheight`, `\headsep` and `\textheight` should be less than `\paperht`, and the difference will be the margin at the bottom of the page!

Running Heads

The page number and text appearing at the top of each page are called the *running heads*. The page number is called the *folio* and the text is called the *mark*. There are two dimensions associated to the running heads, `\foliohang` and `\fromfolio`.

`\foliohang` (|↔|) is the dimension which determines how far out the folio hangs. If `\foliohang` is set to zero points, the folio will hang out only as far as the full page width. A greater value will cause it to hang out even more while a negative value will cause it to be indented.

`\fromfolio` (|↔|) is the amount of space added between the folio and the mark. It is not noticeable at all if the mark and folio are pushed apart (see `\runheadsout`). It has the affect of shifting the point around which the mark is centered if you choose to center th marks (see `\runheadscenter`). This dimension is *most* important in the case of `\runheadsin`, because in this case the distance between the mark and the folio is exactly `\fromfolio`.

If, for some reason, you wish to shift the right running head relative to the left one, you can use the dimension `\foliohangcrrct`.

`\foliohangcrrct` (|↔|) is an amount which the right running head is shifted relative to the left running head. By altering both `\foliohangcrrct` and `\foliohang` you can achieve any desired positioning of the running heads.

There are three common ways in which to position the marks in the running heads. The mark is either centered, pushed to the side opposite the folio or positioned a fixed distance away from the folio. You may choose one of these using the following three commands:

`\runheadsin` (❖) will push the mark *towards* the folio. Since space determined by `\fromfolio` is added between the mark and the folio, they will then be exactly this distance apart. This is the default.

`\runheadsout` (❖) will push the mark to the opposite side of the page from the folio. In this case, the value of `\fromfolio` makes no difference.

`\runheadscenter` (❖) will center the mark at the top of the page. The dimension `\fromfolio` can be used to shift the point around which it is centered

The folio is generated by the command `\thefolio`. Thus, you can change its appearance by changing this command. The font should not be set by this command.

`\thefolio` (❖) is the command which generates the folio (or page number) in the running head. By default, it is defined to be `\colorize{\thepage}`. In other words, it will be the page number (as determined by the LaTeX parameter `\thepage`), printed in color (if you are color separating your document).

There are two fonts used in the running head:

`\foliofont` (✡) is, obviously, the font used to print the folio, or page number, at the top of the page.

`\markfont` (✡) is the font used to print the mark, or text, at the top of each page in the running head.

`\uppercasemarks` (❖) is a flag which will *force* the left and right marks to be uppercase. It will work when the marks are set by the commands `\chapter`, `\section` or `\article`, but not if you set the marks by using some other command. If you want some letters to remain lowercase you can specify that by using the usual TEX `\lowercase` command.

4.4 Parameters for Sectioning Commands

There are three kinds of sectioning commands in ASTEX: `\section`, `\subsection` and `\subsubsection`. You will be using these commands regardless of whether you are doing a textbook or an article. The syntax remains basically the same as in LATEX.

`\section` (❖) (and also `\subsection` and `\subsubsection` which are exactly the same) takes one argument: the title of the section. In addition, there is an optional argument in square brackets in the case that you wish to send something other than the title of the section to the table of contents (`.toc`) file. For example, if the section title is too long for the table of contents, you could put a shorter version of the title in the square brackets. Here are some examples:

```
\section{This is the Title of the Section}
\subsection[Longer Subsection Title]{As You Can See, This is a Longer
Subsection Title...but It Won't Be this Long in the Table of Contents}
```

The big difference between sectioning commands in ASTEX and LATEX is merely the additional control you have over the appearance of the headings.

For each of the sectioning commands (`\section`, `\subsection` and `\subsubsection`) there is a font to specify and two dimensions (the skip before and after the heading) which need to be set in your format file. In addition, a “type” must be selected which determines how the heading will be displayed.

Sectioning Types

Associated to each sectioning command is a command which indicates what special things should be done to indicate the new section. (For example, notice that at each section in this document the title is pulled out to the left and a grey bar appears under it. Alternatively, at the subsection above, although the title is pulled to the left, no bar appears.) These commands are called `\sectype`, `\subsectype` and `\subsubsectype` respectively. You should define these commands in your format file in order to get the look which you want for your documents. There already exist different examples for you to choose from. They are the default value of `\sectype` and also the commands `\sectypea`, `\sectypeb`, `\sectypec`, `\sectyped`, `\sectypee` and `\sectypef`. If none of these satisfy you, you may certainly define your own.

To set `\sectype`, for example, to be one of the sample values, merely use the LATEX command `\let`:

```
\let\sectype\sectypec
```

13

1 Section Type A has Graphics

Every subspace $W \in Gr_1$ can be derived from a line bundle over a singular rational curve *[1]*. As will be explained below, it can also be written as the closure in H of the set of polynomials in z satisfying a finite number of differential conditions at a finite number of points divided by a polynomial q *[2]*. By forming the grassmannian of the linear space of such conditions, one is able to construct finite dimensional grassmannians dual to those in Gr_1.

• 2 Section Type B has a Bullet

Let $d(l,\lambda)$ denote the linear functional on the space $\mathbb{C}[z]$ which takes $f(z)$ to $f^{(l)}(\lambda)$, the l^{th} derivative evaluated at λ. Then let $\mathcal{C}$ be the infinite dimensional vector space over $\mathbb{C}$ generated by $d(l,\lambda)$ for all $l \in \mathbb{N}$ and all $\lambda \in \mathbb{C}$. Let $\mathcal{C}(\lambda) \subset \mathcal{C}$ for $\lambda \in \mathbb{C}$ be the subspace spanned by $d(l,\lambda)$ for all $l \in \mathbb{N}$. Stratify these subsets into $\mathcal{C}(l,\lambda) \subset \mathcal{C}(\lambda)$ which is the subspace spanned by $d(\alpha,\lambda)$ for $0 \le \alpha \le l-1$. (For convenience, we define $\mathcal{C}(0,\lambda)$ to be the empty set.)

Let C be any M dimensional subspace of $\mathcal{C}$. Here I will recall the mapping which associates a point of Gr_1 to C *[2]*. Let V_C be the vector space of polynomials in z which satisfy the condition $c(f) = 0$ for each $c \in C$. Finally, picking a monic polynomial, $q(z)$, of degree M the Hilbert closure of the set $q^{-1}V_C$ in H is a point $W \in Gr_1$.

3 Section Type C is Not Indented

This suggests the following definition. Let $Gr(\mathcal{C})$ be the set of finite dimensional linear subspaces of $\mathcal{C}$. Then letting

$$Gr_1^* = \left\{ (C,q) \in Gr(\mathcal{C}) \times \mathbb{C}[z] \,\middle|\, \dim C = M \text{ and } q = z^M + \sum_{i=0}^{M-1} c_i z^i \right\}$$

allows us to associate a point $W \in Gr_1$ to each $W^* = (C,q) \in Gr_1^*$ by the *dual mapping* described above. Although this mapping is onto (since every element of Gr_1 can be expressed in this way) it is not injective. Note, for example, that $(\{d(0,0)\},z)$ and $(\{d(0,0),d(1,0)\},z^2)$ both get sent to the vacuum solution, $H_+ \in Gr_1$.

4 The Default Section Type

Every subspace $W \in Gr_1$ can be derived from a line bundle over a singular rational curve *[1]*. As will be explained below, it can also be written as the closure in H of the set of polynomials in z satisfying a finite number of differential conditions at a finite number of points divided by a polynomial q *[2]*. By forming the grassmannian of the linear space of such conditions, one is able to construct finite dimensional grassmannians dual to those in Gr_1.

Figure 4.2: This figure demonstrates sample types a, b and c as well as the default value.

13

<u>1 Section Type D is Underlined</u>

Every subspace $W \in Gr_1$ can be derived from a line bundle over a singular rational curve *[1]*. As will be explained below, it can also be written as the closure in H of the set of polynomials in z satisfying a finite number of differential conditions at a finite number of points divided by a polynomial q *[2]*. By forming the grassmannian of the linear space of such conditions, one is able to construct finite dimensional grassmannians dual to those in Gr_1.

2 Section Type E is Centered

Let $d(l,\lambda)$ denote the linear functional on the space $\mathbb{C}[z]$ which takes $f(z)$ to $f^{(l)}(\lambda)$, the l^{th} derivative evaluated at λ. Then let $\mathcal{C}$ be the infinite dimensional vector space over $\mathbb{C}$ generated by $d(l,\lambda)$ for all $l \in \mathbb{N}$ and all $\lambda \in \mathbb{C}$. Let $\mathcal{C}(\lambda) \subset \mathcal{C}$ for $\lambda \in \mathbb{C}$ be the subspace spanned by $d(l,\lambda)$ for all $l \in \mathbb{N}$. Stratify these subsets into $\mathcal{C}(l,\lambda) \subset \mathcal{C}(\lambda)$ which is the subspace spanned by $d(\alpha,\lambda)$ for $0 \le \alpha \le l-1$. (For convenience, we define $\mathcal{C}(0,\lambda)$ to be the empty set.)

Let C be any M dimensional subspace of $\mathcal{C}$. Here I will recall the mapping which associates a point of Gr_1 to C *[2]*. Let V_C be the vector space of polynomials in z which satisfy the condition $c(f) = 0$ for each $c \in C$. Finally, picking a monic polynomial, $q(z)$, of degree M the Hilbert closure of the set $q^{-1}V_C$ in H is a point $W \in Gr_1$.

3 Section Type F is Indented like a Paragraph. This suggests the following definition. Let $Gr(\mathcal{C})$ be the set of finite dimensional linear subspaces of $\mathcal{C}$. Then letting

$$Gr_1^* = \left\{ (C,q) \in Gr(\mathcal{C}) \times \mathbb{C}[z] \,\middle|\, \dim C = M \text{ and } q = z^M + \sum_{i=0}^{M-1} c_i z^i \right\}$$

allows us to associate a point $W \in Gr_1$ to each $W^* = (C,q) \in Gr_1^*$ by the *dual mapping* described above. Although this mapping is onto (since every element of Gr_1 can be expressed in this way) it is not injective. Note, for example, that $(\{d(0,0)\},z)$ and $(\{d(0,0),d(1,0)\},z^2)$ both get sent to the vacuum solution, $H_+ \in Gr_1$.

4 The Default Section Type Every subspace $W \in Gr_1$ can be derived from a line bundle over a singular rational curve *[1]*. As will be explained below, it can also be written as the closure in H of the set of polynomials in z satisfying a finite number of differential conditions at a finite number of points divided by a polynomial q *[2]*. By forming the grassmannian of the linear space of such conditions, one is able to construct finite dimensional grassmannians dual to those in Gr_1.

Figure 4.3: This figure demonstrates sample types d, e and f as well as the default value.

will set the section type to be of type c. Similarly, one may set `\subsectype` with `\let`, but note that it should be set to one of `\sectypea`–`\sectypef` (i.e., there are not separate samples for `\subsection` and `\subsubsection`.

The different sample section types are illustrated in Figures 4.2 and 4.3. It should be noted that type A requires the graphics commands in order to load the shaded bar (it is, in fact, the type which I am using in this paper). Type F and the default section type after it are shown with a negative value of `\post@secskip` in order to have the following paragraph begin on the same line. This dimension is one of the parameters determining the spacing around headings.

Spacing around Sectioning Commands

`\secnumdim` (⟼⟶⟥) is the horizontal space (in all three sectioning commands) between the section number and the beginning of the title. By default, it is set to 1em.

`\pre@secskip` (⟼⟶⟥) is the space before a section command. Since, in fact, it is a "skip" rather than simply a dimension, you can include "plus" and "minus" values for stretch and shrinkability. Furthermore, whether the dimension specified is positive or negative is a "code" to indicate something unexpected. The sign of this dimension determines whether the first paragraph after the section heading is indented or not. That is, if you set

```
\pre@secskip 10pt plus 4pt minus 2pt
```

then the next paragraph after the section heading *will* indent. However, setting

```
\pre@secskip -10pt plus -4pt minus -2pt
```

will leave exactly the same amount of space before the heading, but will not indent the following paragraph.

`\post@secskip` (⟼) is the space *after* a section heading. It too is a skip and can therefore have stretch and shrinkability. Furthermore, the sign of the dimension specified will determine whether the following paragraph will begin on a new line or not. (To see how this looks, consider Figures 4.2 and 4.3. The only difference between the final section heading on each page is the sign of this dimension.) When `\post@secskip` is negative, it is the horizontal distance between the end of the section heading and the start of the following paragraph.

`\secfont` (✡) is the font used for section headings.

The commands used in the case of `\subsection` and `\subsubsection` headings are analogous, and are called `\pre@subsec`, `\post@subsec`, and `\subsecfont`, `\pre@subsubsec`, `\post@subsubsec` and `\subsubsecfont`.

Capitalizing Section Headings

It might be in your specifications that section headings are always capitalized. Rather than actually typing them this way (and possibly typing them differently for the table of contents) this can be "automated" using the command `\section@uc`. Merely set this command to be `\uppercase` in your format file and the section headings will automatically be in uppercase letters.

```
\let\section@uc\uppercase
```

Similarly, the commands `\subsection@uc` and `\subsubsection@uc` can be used to force the other sectioning commmands to print their headings in uppercase letters.

A similar new feature in version 3.0 is the command `\title@uc` which will allow chapter titles to be forced into uppercase. In particular, if you include the command `\uppercasetitle` in your format file or equivalently type:

```
\let\title@uc\uppercase
```

then the chapter title will be forced into uppercase. (Note that the running heads will still be in both upper and lower-case letters. So, this provides a means for the chapter title to be in caps on the first page but in upper and lower case letters in the running heads.

Note: This will force *all* of the letters in a heading, including the ones in math mode, to be uppercase. Since you will probably not want every letter to be converted to uppercase, you will want to also use the command `\lowercase` to specify that certain characters not be uppercase. For example, suppose that you have set `\subsection@uc` to be `\uppercase`, and you want to write the expression $R = x^2 + y^2$ in the heading, then you would type

```
\subsection{The Equation of a Circle in the form
$R=\lowercase{x}^2+\lowercase{y}^2$}
```

4.5 The defs File

If your format name is, for example, `MY`, then ASTEX will attempt to load a file called `as-defsMY.tex` after reading the format file. No error message will occur if this file is not present. If you are writing any of your own commands which are format specific, they should go into this defs file. It can even be considered as an extension of the format file. The only real difference between this file and the format file are that this file is read last, whereas the font file, and some other things are read after the format file.

5 Articles or Textbook?

One advantage of ASTEX over the variety of packages with my macros that existed before is that it combines *all* of the commands together in one system. The sections below which describe "commands for articles" or "commands for textbooks" are really only offering suggestions. You can use any of these commands in either medium. However, there are certain things which nearly always are true in one or the other. The commands `\articledefaults` and `\textbookdefaults` (one of which should probably be placed at the beginning of your format file) set up: the counters, the auto-page routine (by issue in the case of articles and by chapter in the textbook case), and the table of contents (not to print sections and subsections in the articles case). However, even after you have called one of these commands, you can use any of the commands below in either a book or an article.

6 Commands for Typesetting Articles

6.1 File Structure for Articles

The basic file structure for typesetting a book or journal made up of many articles is as follows. There is one file which inputs each article. That file looks like this:

```
\documentstyle{astex}

\asformat{THEFORMAT} % your format name here

\showtextbox         % \
\showgrid            % - flags are optional, but when they are used,
                     %   they should go here.

\volumenum{V}        %
\issuenum{3}         %  These commands are used for journals.
\pubdate{July, 1994} %


\begin{document}

\tableofcontents     % This command is can go at the beginning, the
     % end, or nowhere.  The page number will
     % automatically be reset after the table of
     % contents.

\article{FILENAME}{AUTHORS}{TITLE}
\article{FILENAME}{AUTHORS}{TITLE}
\article{FILENAME}{AUTHORS}{TITLE}
.
.
.
```

```
\article{FILENAME}{AUTHORS}{TITLE}
\article{FILENAME}{AUTHORS}{TITLE}
\article{FILENAME}{AUTHORS}{TITLE}

\end{document}
```

Then, you need a separate file for each article, which gets loaded with the `\article` command.

`\article` (❖) is the command used for loading an article file in a collection of articles or journals. In general, it takes three arguments as shown

```
\article{FILENAME}{AUTHORS}{TITLE}
```

and it will then load the file `FILENAME.tex` and put the authors and title in the running head at the top of the page and in the table of contents. If, as is sometimes the case, you want the title to appear *differently* in the running head and in the table of contents, then you can use an optional argument as shown

```
\article{FILENAME}{AUTHORS}[RUN HEAD TITLE]{TABLE OF CONTENTS TITLE}
```

6.2 Commands for the beginning of an article

`\title` (❖) Takes only one argument and that is the title of the paper. I expect there will be various styles, but right now there is just the one! It centers the title and prints it in a big font.

`\author` (❖) Takes one argument, which is the author name or authors. It then takes arbitrarily many other arguments, in squar brackets, which are the address of the author. If there are two or more authors with different addresses, then merely use consecutive `\author` commands.

For example, the commands

```
\author{John Adams and Benjamin Franklin}[Continental
Congress][Philadelphia, PA]

\author{Alex Kasman}[Department of Mathematics][Boston
 University][Boston, MA][USA]
```

produce the output:

John Adams and Benjamin Franklin
Continental Congress
Philadelphia, PA

Alex Kasman
Department of Mathematics
Boston University
Boston, MA
USA

`\abstract` (❖) Takes one argument, which is the abstract of the paper (usually one or two paragraphs). There are various types, determined by the command `\abstracttype`. To set the type, include a line such as

```
\let\abstracttype=\abstracttypea
```

The types which are provided are `\abstracttypea`–`\abstracttyped`.

Examples of various versions of `\abstracttype` are presented in Figure 6.2.

1

Abstract Type A:

Abstract: Most people are probably more familiar with classical physics, in which particles are represented as *points* with mass, speed, position, etc. One expects that the laws of physics could be represented mathematically (this has been the philosophy of physics since Newton) and the way it is done is by choosing variables that represent the position, speed, mass, etc... of the particle and then writing equations which tell you how they relate.

These equations turn out to be *differential* equations, which means equations which involve taking the derivatives of these quantities. For example, one very powerful technique is that of the Hamiltonian.

Abstract Type B:

Most people are probably more familiar with classical physics, in which particles are represented as *points* with mass, speed, position, etc. One expects that the laws of physics could be represented mathematically (this has been the philosophy of physics since Newton) and the way it is done is by choosing variables that represent the position, speed, mass, etc... of the particle and then writing equations which tell you how they relate.

These equations turn out to be *differential* equations, which means equations which involve taking the derivatives of these quantities. For example, one very powerful technique is that of the Hamiltonian.

Abstract Type C:

Abstract: Most people are probably more familiar with classical physics, in which particles are represented as *points* with mass, speed, position, etc. One expects that the laws of physics could be represented mathematically (this has been the philosophy of physics since Newton) and the way it is done is by choosing variables that represent the position, speed, mass, etc... of the particle and then writing equations which tell you how they relate.

These equations turn out to be *differential* equations, which means equations which involve taking the derivatives of these quantities. For example, one very powerful technique is that of the Hamiltonian.

Abstract Type D:

Most people are probably more familiar with classical physics, in which particles are represented as *points* with mass, speed, position, etc. One expects that the laws of physics could be represented mathematically (this has been the philosophy of physics since Newton) and the way it is done is by choosing variables that represent the position, speed, mass, etc... of the particle and then writing equations which tell you how they relate.

These equations turn out to be *differential* equations, which means equations which involve taking the derivatives of these quantities. For example, one very powerful technique is that of the Hamiltonian.

Figure 6.4: Examples of different abstract types

6.3 Bibliography

The bibliography should come at the end of the paper in the `bibliography` environment using the `\bibitem` command as follows:

`bibliography` (`\begin`-`\end`) is an environment containing nothing other than `\bibitem` commands. So, it would be used like

```
\begin{bibliography}
\bibitem[AK]{A. Kasman, ‘‘A Concise History of Onions’’, {\it The
International Journal of Onions\/}, Vol. 3. No. 2, 1987}
.
.
.
\bibitem[Z]{J. Zorn, ‘‘A Paper’’, {\it A Journal\/}, No. 3, 1999}
\end{bibliography}
```

`\bibitem` (❖) is the command which is used in the bibliography. It takes two arguments, the first in square brackets is the label which you can refer to with the `\bibref` command, and the other is the text to be printed for the reference. Although the examples above show a reasonable usage of this command, there is no reason that the second argument must be in the form of these examples. You may write *anything* which will generate the reference you want.

In general, the number of the reference will be displayed in the same format as a reference number in the text. That is, it will displayed using the `\bibtype`. If, for some reason, you wish the numbers to appear differently in the bibliography than they do in a bibliography reference in the text, you can set `\bibitemtype` independently, just as you would set `\bibtype`.

Note: Due to the command `\bibref` (See below) a comma (`,`), square brackets (`[` or `]`) or two consecutive dashes (`--`) cannot be used in the label. Also, note that the labels generated by the `\bibitem` command are stored separately from the labels generated by other commands. Consequently, it is possible for you to use the same label for a bibliography entry and another object such as a picture or claim.

The `\bibref` command should be used in the text to refer to a bibliography entry by its label.

`\bibref` (❖) takes one argument in square brackets, which is a list of labels for bibliography entries separated by commas or double-dashes. For example,

```
\bibref[AK,JoBBt--M,Z]
```

is the way one would refer to references `AK` and `Z` as well as all references between `JoBBt` and `M`. The appearance of the reference is determined by `\bibtype`. Several sample `\bibtype`s are provided for your use. As usual, you set `\bibtype` to any of these samples using a line such as

```
\let\bibtype\bibtypea
```

in your format file. Here are examples of the sample bibtypes:

- bibtypea: *[1, 3–4]*
- bibtypeb: $^{1,\ 3–4}$
- bibtypec: [**1, 3–4**]
- bibtyped: 13.
- bibtypee: (13)
- bibtypef: **13**

Note: A warning will be printed on the screen and in the log file when a bibliography reference is not recognized. As with standard LaTeX labels, this generally means that the file must be LaTeXed again in order for the reference to be found. If such a warning appears repeatedly, it is an indication that either the label or the reference are written incorrectly.

In general, the `\bibtype` also determines how the numbers in the bibliography itself will be displayed. However, if you wish to use a different bibtype in the bibliography, set `\bibitemtype` to the desired definition.

Note: I assume that a comma and a space are generally used to separate the printed form of different arguments in a `\bibref` command. That is, the printed form will usually be [1, 23, 35] rather than [1,23,35] or [1/23/35]. However, if you do want one of these alternative forms, you should change the definition of `\bibseps`. By default, this is ", ", however changing it to "," or "/" (for example) would result in the forms above.

By default, the commands above will always number the bibliography references and refer to them by number in the text. In some cases, people prefer to specify strings of arbitrary characters. Using the flag `\exactrefs`, you can specify that this should be done either in an individual article, an entire book or in an entire format.

`\exactrefs` (❖) is a flag which causes `\bibitem` and `\bibref` to print the argument which is normally a label, exactly as it is with no interpretation. That is, once the flag has been called, the command `\bibref[AK,Z and Q]` will print exactly the string contained in the square brackets (using the appropriate `\bibtype`). For example, the command above would produce *[AK,Z and Q]* under `\bibtypea` or $^{\text{AK,Z and Q}}$ under `\bibtypeb`. However, it will do this regardless of whether there are any bibliography entries with those names. You can type anything you want, but you no longer get an error statement if you make a mistake.

Place `\exactrefs` at the beginning of an article to have it affect that article only. Put it at the beginning of a list of articles to affect the entire list or put it in your format file to have it apply to all files printed in your format.

6.4 Claims

Claims are things such as theorems, corollaries, etc. There are two ways to call them: either as a statement (which does not involve a proof) or in an environment including a proof. Below, I will use `claim` as an example name for my claim, but it would apply equally to any other claim. For example, when typing a corollary, merely use `\begin{corollary}` in place of `\begin{claim}` and use `\statecorollary` in place of `\stateclaim`.

Note: These claim commands "overwrite" the LaTeX commands in the sense that the command `\theorem` is now the ASTEX command described below rather than the one defined in LaTeX.

Stating a Claim without Proof

`\stateclaim` (❖) Generates the statement of a claim with no proof following. It takes the statement of the claim as one argument, and has one optional argument which preceeds that. The optional argument is either a label by which the claim can be referred to, in square brackets, *or* the name or number of the claim in angular brackets. If no argument in angular brackets is used, the claim is automatically numbered in sequence with other claims. If the argument in angular brackets is a single dash ("-") then only the word "Claim" is printed, with no following number or name.

Examples of the use of `\stateclaim` are:

```
\stateclaim{There is no largest prime number.}
```

which produces the output

Claim 6.1: *There is no largest prime number.*

but cannot be referred to in the text since I have no explicit way of knowing what number it is.

Alternatively, one can use

```
\stateclaim[LgPrime]{There is no largest prime number.}
```

which produces the output

Claim 6.2: *There is no largest prime number.*

(essentially the same as the one above) but now I can refer to it as `Claim~\ref{LgPrime}`, which would look like "Claim 6.2".

Also, I can specify the name or number of the claim explicitly using angular brackets:

```
\stateclaim<A>{There is no largest prime number.}
```

which looks like

Claim A: *There is no largest prime number.*

I cannot refer to this using a `\ref` command...but I don't have to because I know exactly what it is called: A!

On rare occasions, one might want a claim with no name or number. In this case, put a single dash in the angular brackets:

```
\stateclaim<->{There is no largest prime number.}
```

and your output will look like

Claim: *There is no largest prime number.*

Of course, there are a variety of claim types provided for your use. You may set `\claimtype` to any of `\claimtypea` through `\claimtypee` to alter the output of `\stateclaim`.

`\claimtype` (❖) determines the appearance of a claim. Five samples are provided: `\claimtypea` – `\claimtypee`. They are set using the command `\let` as is common within this package.

Stating a Claim with Proof

When a proof is to be provided as well, then the claim should be in the form of an environment:

`claim` (`\begin`-`\end`) is the environmental form of the command `\stateclaim`. It takes all of the same arguments as `\stateclaim` (the square brackets for labels, angular brackets for name or number followed by the claim in braces), but then adds an end of proof box at the `\end` command.

For example:

```
\begin{claim}<D>{There are infinitely many prime numbers}

Suppose there were only a finite number.  Then take the product....
...which contradicts the assumption.
\end{claim}
```

would produce the output:

Claim D: *There are infinitely many prime numbers*

Proof: Suppose there were only a finite number. Then take the product....
...which contradicts the assumption.

The word "Proof" is put in automatically. If you want something other than the word "proof" to appear, use the command `\proofhead` to specify the phrase you want. For example:

```
\begin{claim}<D>{There are infinitely many prime numbers}

\proofhead{Outline of Proof} Suppose there were only a finite....\\
...which contradicts the assumption.
\end{claim}
```

Claim D: *There are infinitely many prime numbers*

Outline of Proof: Suppose there were only a finite...
...which contradicts the assumption.

The same `\claimtype` which determines the appearance of `\stateclaim` determines the appearance of the environment `claim`. The part before the word "Proof" will look the same, except for type "E", in which a line appears in the environmental form which was not present in the statement.

Of course, there are different forms of the end of proof box (which is usually a box...hence the name). You can specify that `\proofbox` be any of `\proofboxa` through `\proofboxe` using the `\let` command. The different forms look like

- `\proofboxa` looks like:
- `\proofboxb` looks like:
- `\proofboxc` looks like:
- `\proofboxd` looks like:
- Finally, `\proofboxe` is the long horizontal line demonstrated in the examples above. It works well with `\claimtypee` which is also a horizontal line.

`\noproofbox` (❖) is a command which will cause the proof box on the next environmental claim to be left off. It can be called anywhere prior to the `\end` statement which would produce the box.

`\newclaim` (❖) is a command for creating new claim statements and environments. This has already been done for `claim`, `theorem`, `corollary`, and `proposition`. However, if you want to add new claim commands, for example if you want a claim command called "conjecture", merely add the line

```
\newclaim{conjecture}
```

to your format file (or simply in the TEX file where you need it).

Using Different Claimtypes

It is possible to use a different `\claimtype` for each type of claim. For example, your theorems could be formatted differently from your corollaries. To do this, merely use `\let` or `\def` to set the command `\WHATEVERct` to be the appropriate type, where `WHATEVER` is `theorem` or `corollary`. For example, if you want lemmas to be set using `\claimtypec`, simply include the line:

```
\let\lemmact\claimtypec
```

All other claims will still be determined by the value of `\claimtype`.

You may take advantage of this to use the claim routine for things which aren't even claims. For example, you could now define commands to be used for remarks or definitions which are (behind the scenes) done with the claim routine.

It may be that you don't want to use the fonts which are associated with the claim routine (which are `\claimheadfont` and `\claimbodyfont`). In this case, you will want to define your own claim type since all of the supplied claim types use these fonts. Look at the definition of `\claimtypea`, `\claimtypeb`, etc. to get an idea of how these work.

Finally, you may not want an end of proof box to appear at the end of some claim type (especially if it is really a definition or something else which is not actually a claim). In this case, you merely need to redefine the command `\endWHATEVER`. For example, if you want lemmas to end with a bit of vertical space, but no proof box, you may use the definition

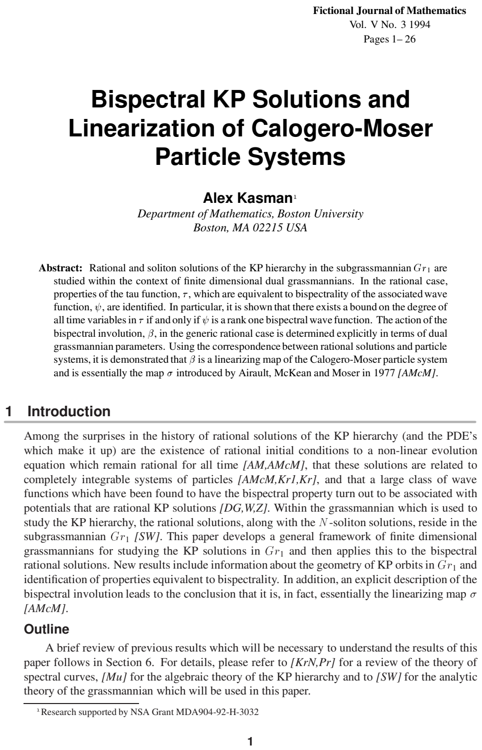
```
\def\endlemma{\bigskip\par}
```

Note, in addition, that you can build your own claim "environment" if the actual claim environment does not suit your purpose. That is, if you start with `\stateclaim`, then indicate the beginning of the proof with `\proof`, and end with `\endproof`, it will look just like the claim environment. In addition, you can change the appearance of the word "proof" by altering the definition of `\proof`.

6.5 InfoBox

By infobox, I mean the box which appears on the first page of an article with information about the journal. It will be placed on an article automatically when the article is loaded with the `\article` command, and you have specified a position for the box to appear using the command `\putinfobox`. Some examples of an infobox appear in Figure 6.5.

`\putinfobox` (❖) is a command which should be placed in your format file to indicate the position of the infobox. The possible arguments are `tl, t, tr, bl, b, br` for the top-left, top,..., bottom, and bottom-right of the page.


Fictional Journal of Mathematics
Vol. V No. 3 1994
Pages 1–26

Bispectral KP Solutions and Linearization of Calogero-Moser Particle Systems

Alex Kasman[1]
Department of Mathematics, Boston University
Boston, MA 02215 USA

Abstract: Rational and soliton solutions of the KP hierarchy in the subgrassmannian Gr_1 are studied within the context of finite dimensional dual grassmannians. In the rational case, properties of the tau function, τ, which are equivalent to bispectrality of the associated wave function, ψ, are identified. In particular, it is shown that there exists a bound on the degree of all time variables in τ if and only if ψ is a rank one bispectral wave function. The action of the bispectral involution, β, in the generic rational case is determined explicitly in terms of dual grassmannian parameters. Using the correspondence between rational solutions and particle systems, it is demonstrated that β is a linearizing map of the Calogero-Moser particle system and is essentially the map σ introduced by Airault, McKean and Moser in 1977 *[AMcM]*.

1 Introduction

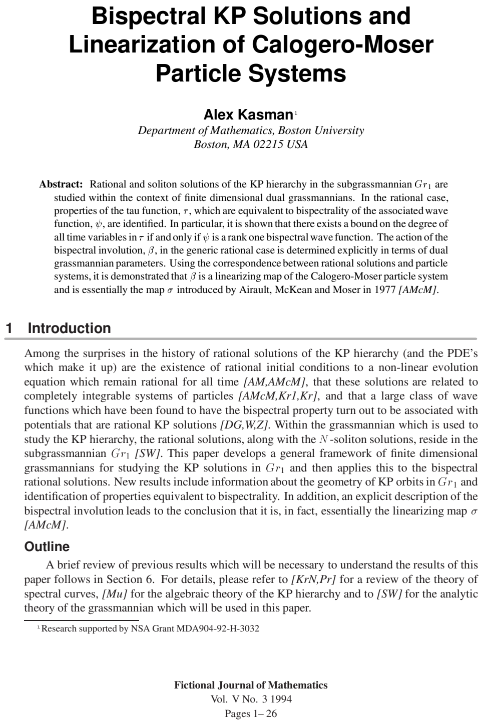
Among the surprises in the history of rational solutions of the KP hierarchy (and the PDE's which make it up) are the existence of rational initial conditions to a non-linear evolution equation which remain rational for all time *[AM,AMcM]*, that these solutions are related to completely integrable systems of particles *[AMcM,Kr1,Kr]*, and that a large class of wave functions which have been found to have the bispectral property turn out to be associated with potentials that are rational KP solutions *[DG,W,Z]*. Within the grassmannian which is used to study the KP hierarchy, the rational solutions, along with the N-soliton solutions, reside in the subgrassmannian Gr_1 *[SW]*. This paper develops a general framework of finite dimensional grassmannians for studying the KP solutions in Gr_1 and then applies this to the bispectral rational solutions. New results include information about the geometry of KP orbits in Gr_1 and identification of properties equivalent to bispectrality. In addition, an explicit description of the bispectral involution leads to the conclusion that it is, in fact, essentially the linearizing map σ *[AMcM]*.

Outline

A brief review of previous results which will be necessary to understand the results of this paper follows in Section 6. For details, please refer to *[KrN,Pr]* for a review of the theory of spectral curves, *[Mu]* for the algebraic theory of the KP hierarchy and to *[SW]* for the analytic theory of the grassmannian which will be used in this paper.

[1] Research supported by NSA Grant MDA904-92-H-3032

1


Bispectral KP Solutions and Linearization of Calogero-Moser Particle Systems

Alex Kasman[1]
Department of Mathematics, Boston University
Boston, MA 02215 USA

Abstract: Rational and soliton solutions of the KP hierarchy in the subgrassmannian Gr_1 are studied within the context of finite dimensional dual grassmannians. In the rational case, properties of the tau function, τ, which are equivalent to bispectrality of the associated wave function, ψ, are identified. In particular, it is shown that there exists a bound on the degree of all time variables in τ if and only if ψ is a rank one bispectral wave function. The action of the bispectral involution, β, in the generic rational case is determined explicitly in terms of dual grassmannian parameters. Using the correspondence between rational solutions and particle systems, it is demonstrated that β is a linearizing map of the Calogero-Moser particle system and is essentially the map σ introduced by Airault, McKean and Moser in 1977 *[AMcM]*.

1 Introduction

Among the surprises in the history of rational solutions of the KP hierarchy (and the PDE's which make it up) are the existence of rational initial conditions to a non-linear evolution equation which remain rational for all time *[AM,AMcM]*, that these solutions are related to completely integrable systems of particles *[AMcM,Kr1,Kr]*, and that a large class of wave functions which have been found to have the bispectral property turn out to be associated with potentials that are rational KP solutions *[DG,W,Z]*. Within the grassmannian which is used to study the KP hierarchy, the rational solutions, along with the N-soliton solutions, reside in the subgrassmannian Gr_1 *[SW]*. This paper develops a general framework of finite dimensional grassmannians for studying the KP solutions in Gr_1 and then applies this to the bispectral rational solutions. New results include information about the geometry of KP orbits in Gr_1 and identification of properties equivalent to bispectrality. In addition, an explicit description of the bispectral involution leads to the conclusion that it is, in fact, essentially the linearizing map σ *[AMcM]*.

Outline

A brief review of previous results which will be necessary to understand the results of this paper follows in Section 6. For details, please refer to *[KrN,Pr]* for a review of the theory of spectral curves, *[Mu]* for the algebraic theory of the KP hierarchy and to *[SW]* for the analytic theory of the grassmannian which will be used in this paper.

[1] Research supported by NSA Grant MDA904-92-H-3032

Fictional Journal of Mathematics
Vol. V No. 3 1994
Pages 1–26

Figure 6.5: Examples of the infobox in the top right corner and bottom center.

When the position specified is the bottom of the page, by default, no page number is printed on the page. (It is expected that you will make it part of the infobox itself). When the position specified is on the top of the page, the page number will be centered at the bottom of the page. Either of these can be changed by altering the definition of `\ps@firstpage`.

`\infobox` (❖) is the command which determines the contents of the infobox. You define it in your format file using whatever commands you wish. It should be the lines which you want to appear in the box separated by the line break command `\\`. It is recommended that you use the commands `\thepage`, `\thelastpage`, `\thejournal`, `\theissue`, `\thevolume` and `\thepubdate` along with `\\` for beginning a new line. The default definition of `\infobox` is:

```
\def\infobox{\small\bf\thejournal \\
\small Vol. \thevolume\ No. \theissue\ \thepubdate\\
\small Pages \thepage -- \thelastpage}
```

which is probably very close to what you would want. (It is illustrated in Figure 6.5.)

Note: If you are using an infobox for a book of articles, rather than a journal, you can replace the definition above with one which would be more appropriate. For example, the first line would be the book name, the second line the publication information and the last line would be page numbers as above.

`\issuenum` (❖) is a command that defines the variable `\theissue` containing the current issue number of the journal. Its use is illustrated above in the sample file structure.

That is, prior to the `\begin{document}` you may use this command to specify the issue number. The variable `\theissue` is not only used for printing the issue number in the infobox but *also* for setting the correct page number relative to the previous issue!

Similar commands, `\volumenum`, `\pubdate`, and `\journalname` exist for setting the other variables often used in the infobox.

`\infospace` (⟼) is the vertical space between the infobox and the main textbox. That is, if the infobox is at the top of the page, it is the space between it and the first object on the page. Alternatively, if the infobox is at the bottom, then this is the space between the top of the infobox and the last item on the page.

More Infoboxes?

You say one info box isn't enough? Well, I'll tell you what – I'll throw in another one *free*, but it is a *little* one. Just as you define `\infobox` using `\def` to be a series of short lines broken by the command \\, you may define `\littleinfobox` to be another one. The little infobox will show up in the top right or bottom right corner of the page (opposite to the position of the large infobox.) In actuality, you can make it as large as you want. However, unlike the other infobox, no extra room on the page is left for the little infobox. So, the problem if you do not make it small enough is that it is likely to "crash" into other things and look crowded. It can handle one or two lines, and is sufficient for copyright or pricing information which frequently appears in journals.

6.6 Submissions from Authors

You can give a copy of `astex.sty` to authors who will be contributing to your journal or book. Since the `DEFAULT` format does not require any additional files, they can use that file alone in order to process their own article with ASTEX. (Alternatively, you could supply them with your own format file.) A short appendix at the end of these instructions is intended for the use of these authors (although they are not independent of the rest of the instructions). It would probably be useful for you to write your own "instructions for authors" which includes information on how to get a copy of ASTEX and how to use it. In addition, two short sample articles are contained in an appendix for quick reference.

7 Commands for Typesetting a Textbook

The textbook commands are macros for use in an undergraduate textbook. In particular, it helps in the handling of end of section exercises and solution manuals. In addition, it helps you work with separate TEX files for each chapter by organizing an index, table of contents and automatic referencing between chapters.

7.1 An Outline of the File Structure of a Textbook

The main point here is that there is not *one* file which (when run through LATEX) produces the entire book. Each chapter is in its own file. In addition, the preface, table of contents, index and other portions are also in separate files. There are two reasons that I have chosen to do it this way:

- ASTEX uses up a lot of memory in order to do all of the things which it does. There probably would not be enough memory to run an entire book at once.
- running very large files can be very frustrating, because they take a long time and you *don't* want to find an error in the last chapter and then have to run the entire file again.

Consequently, ASTEX contains commands, hacks and routines that link the files. For example, the command `\gref` allows you to cross reference between files. In addition, your format file will contain a *chapter list*. The chapter list (created with the command `\nextchap`) is an ordered list of the files which make up the book and the directories in which they are contained. This serves several functions, including: automatically setting the page number of the first page in a chapter to the correct value as determined by the last page number of the preceeding chapter, generating a table of contents and generating an index.

It is recommended that each file in the chapter list be placed in a separate directory. (That means a separate directory for each chapter.) Below, I will generally use the naming scheme in which the directory `CHAP1` contains a chapter file called `chap1.tex` and so on. This is merely an example; you can use any name for the chapter files or directories that you wish. However, I cannot over-emphasize the importance of attempting to use a consistent naming scheme. This makes it simpler for you to find files, but in addition, allows the possiblity of writing a computer program to perform complicated tasks on the files.

The chapter files themselves will input other files. For example, you will probably use the command `\secinput` to input the sections as individual files, (although you don't have to). The exercises, though, as well as the figures will each be in a separate file. The command `\exercises` or `\finput` will appear in the chapter file to load exercises of figures, respectively. Again, a consistent file naming scheme should be used. My personal preference is to name figures, question files and exercise lists so that the filename immediately tells you the location of the object in the book. For example, `3-2ex4.tex` might be the name of exercise 4 in section 3.2 and `fig5p2.tex` might be figure 2 in chapter 5.

7.2 Setting Up Your Format File For Textbook Production

You must put several things in your format file in order to take advantage of ASTEX's textbook production commands. First you must specify a *home directory* by defining the variable `\homedir`. Each chapter (and other portions of the book) will be contained in separate directories under this home directory. For example, my formt file contains the definition

```
\def\homedir{/usr2/alexk}
```

Then, you must indicate the location and order of each of the chapters relative to this home directory. This will be used, for example, to collect the table of contents or index of each portion of the book and combine them into one file. This is done with the `\nextchap` command. A sequence of these commands, with two arguments indicating the name and location of the files, is all that is needed. The command `\nextchap` takes two arguments: the first being the name of the directory (relative to the home directory) and the second being the name of the file (with the suffix `.tex` left off). If my chapter files were `/usr2/alexk/CHAP1/chap1.tex` and so on, my format file would contain the lines

```
\nextchap{FRONTMATTER}{preface}
\nextchap{FRONTMATTER}{to-the-student}
\nextchap{CHAP1}{chap1}
\nextchap{CHAP2}{chap2}
.
.
```

```
.
\nextchap{CHAP11}{chap11}
\nextchap{ENDSTUFF}{appendixA}
\nextchap{ENDSTUFF}{appendixB}
\nextchap{ENDSTUFF}{index}
```

where here I am also hypothesizing the existence of other portions of the book besides the chapters.

Note: Remember to update this *chapter list* if any of the information changes (for example, if you add a new chapter).

One function served by the chapter list is that it automatically sets the page number of the files in the list. It does this by checking *only* the name of the file, not the directory. That is, if I have a file called `chap2.tex` in some directory other than `CHAP2`, it will still automatically start on the page after chapter 1. **So, you should not have another file with the same name as some file in the chapter list,** unless you also want it to start at the same page number. See information on `\autopagebychapter` for more information.

Headings

There are two types of headings which are likely to be used in a textbook that you should define. One is the heading for *chapters*. The command you must define in your format file is `\chaphead`. This command, which takes no arguments, will be used to generate the chapter heading. It is like `\sectype` in that it is something you define to control the display of a sectioning command (the `\chapter` command, in particular). It may take up a whole page, or it might only be one line long. Commands that you might want to use in its definition include `\arabic{chapter}` and `\numberword{chapter}` which give the current chapter number in arabic numerals or spelled out as an English word respectively. Also, you should definitely use `\thetitle` which is the title of the chapter. You may want to use `\thesummary` which will attempt to load a summary file of the chapter. (That is, if you want a summary to appear on the first page of every chapter of the book, include the command `\thesummary` in your definition of `\chaphead` and create a file containing the summary called `CHAPNAME-summary.tex` in each directory, where `CHAPNAME` is the name of the chapter file.) As with `\sectype`, there is a default definition along with several alternatives for you to try or to study in order to create your own definition. The alternate definitions are `\chapheada` through `\chapheade`. To use one, merely include the line

```
\let\chaphead\chapheada
```

for example, in your format file. The different sample "chapheads" are shown in Figure 7.6. The final sample, `\chapheade` is not illustrated in Figure 7.6, but looks quite a bit like the cover page of these instructions. Note that `\chapheade` is not very versatile. In particular, the black background obscures the crop marks and registration marks, and so it would not be good for producing photoready film or color separations. (But it sure looks nice.) The sample chapter headings which I provide use the fonts `\chapfont` and `\chaptitlfont`. So, if you use one of my samples, you will probably want to define these to suit your tastes.

In addition, you must define headings for the exercises. Most books have at least two types of exercise headings. One would say something like "Exercises for Section 3.2"