The cpssp package*

Wolfgang Skala
Wolfgang.Skala@sbg.ac.at

2009/06/06

1 Introduction

The cpssp package allows you to draw a two-dimensional representation of a protein's secondary structure in LaTeX. Besides, it is possible to graphically **c**ompare **p**rotein **s**econdary **s**tructure **p**redictions (hence the name of the package). One can both compare predictions from a single program for several protein sequences (which have been aligned using any appropriate algorithm) and predictions from several programs for a single protein sequence.

The following steps are necessary to obtain such a representation: First, input files for the cpssp program have to be prepared (§ 2). Second, these input files are run through the cpssp program, which generates files that can be read by LaTeX (§ 3). Third, these files are included in a LaTeX document which uses the cpssp package, thus providing the necessary commands (§ 4).

A detailed description of these steps follows, and examples use sequences and predicted secondary structures for collagenases G (ColG) and H (ColH) from *Clostridium histolyticum* and for collagenase T (ColT) from *C. tetani*.

2 Preparation of the Input Files

cpssp is a Python 2.6 script and should run on any platform where Python is available. Start the program by either typing `python cpssp` ⟨*options*⟩ or by making sure that the script is executable (e.g., `chmod u+x cpssp`) and then directly starting the file (`cpssp` ⟨*options*⟩).

cpssp needs one or two input files, one containing the sequences and the other containing the secondary structure predictions. If you want to compare the secondary structure predictions from several programs of the same sequence, you only need the latter, whereas both are needed if you plan to compare the secondary structures predicted by a single program, but for two or more different sequences.

The input files must be in FASTA format. Thus, each sequence or structure starts with a header line whose first character is > followed by a description of the sequence. Note that this description is used as a label in the graphical output (the description lines from the sequence file if several sequences are compared, and the description lines from the structure file if several prediction programs are compared), so it is advisable to keep it short (in our example below, the sequences

*This document corresponds to cpssp v1.0, dated 2009/06/06.

have been named after the organism they are from, and the structures have been named after the program which predicted them).

The FASTA file containing the sequences may be exported from any multiple sequence alignment program you like. Gaps in the sequences may be indicated by dashes (-) or periods (.); besides, the standard one letter abbreviations for amino acids (ARNDCQEGHILKMFPSTWYV) are allowed. In the structure FASTA file, the following letters are allowed: B (β-bridge), C (coil), E (β-strand), G (3_{10}-helix), H (α-helix), I (π-helix), S (bend), and T (β-turn); in addition, a coil is indicated by a space ( ) or a dash (-) in the output of some prediction programs. All these secondary structure elements are supported; although most algorithms predict only coil, α-helices and β-strands, the remaining elements occur in the output of secondary structure assignment programs such as STRIDE. `cpssp` has been tested with the output from STRIDE [1] and the following prediction algorithms: PSIPRED [2], Jpred [3], CDM [4], SSpro [5], and SABLE [6].

The order of the sequences or structures in the FASTA files determines their order in the graphical output. Note that if using both sequence and structure file, the order must be the same in these files. Furthermore, each sequence must be as long as the corresponding structure and all sequences or structures must have the same length (gaps included), or the input files can't be processed properly.

In our first example, ClustalX [7] was used to align the catalytic domains of three clostridial collagenases (ColG, ColH and ColT) [8]. The aligned sequences were exported from the resulting multiple sequence alignment by using JalView [9]. The corresponding FASTA file is `colsequences.fasta`. The FASTA file containing the structures is `colstructures.fasta`; it was made by obtaining the predictions of the three sequences from PSIPRED followed by copying and pasting the prediction lines and adding headers.

Our second example will compare the secondary structure of ColG predicted by five different programs (the ones mentioned above). Therefore, the same approach as for the structure file from the first example was made, yielding the file `colpredictions.fasta`.

The third example makes use of another possibility provided by `cpssp`, namely to print a single secondary structure prediction for a single protein. The PSIPRED prediction for ColT was therefore copied into the file `coltprediction.fasta` as described above.

3 Running the `cpssp` Program

The `cpssp` program is a Python script and should run on every system where a Python implementation is available. Basically, it reads the input file(s) and decides whether to compare structures predicted by the same algorithm for different proteins or structures predicted by different algorithms for the same protein. Then, gaps appearing in each sequence at the same position are removed from the sequences. The structures read from the file get a common syntax (i.e., all spaces and dashes indicating a coil are converted to a C) and are gapped so that they possess the same gaps as their respective sequences. In a final step, structures are broke into lines and are translated to LaTeX commands which are written to one or several output files.

The `cpssp` command supports the following options:

- `-h` or `--help`: Prints a short description of the available parameters.

- `-v` or `--version`: Prints the program version.
- `-s` or `--sequence-file`: The name of the input file containing the sequence data. This option must be omitted if structures predicted by different programs for a single sequence are to be compared.
- `-u` or `--structure-file`: This is the only mandatory option, as the name of the input file containing the structure data must be given in each case.
- `-o` or `--output-file`: Depending on the size of the graphical output (see below), `cpssp` splits the output into several separate files. This is a useful feature if you want to include your comparison in a paper and have to deal with a limited page size (but can be turned off nevertheless as described below). The default output filename is `cpsspresultX.tex`, where `X` is the part that is consecutively numbered. However, by specifying the `-o` option and its argument, you may change the filename before the number. For example, `-o tpp2results` would give `tpp2results0.tex`, `tpp2results1.tex` and so on.
- `-w` or `--image-width`: A number that sets the approximate width w of the image in cm (the default value is 15). Note that the total width is somewhat larger if you decide to include the number of the residues at the end of each line in the final image.
- `-t` or `--image-height`: A number that sets the maximal height of each image in cm (the default value is 20). `cpssp` uses an algorithm where a new output file is started if the total height exceeds the value given by this option, thereby breaking a comparison into several images. While this feature is useful within the constraints of printing paper, is can be turned off by specifying a maximal height of zero (`-t 0`). Note that the height specified is not reached in any case; this is due to the distances of lines and blocks (see below).
- `-i` or `--line-indent`: A number that specifies the space that is left blank at the beginning of each line in cm (default is 2.5). This space is reserved for the labels that tell the reader which protein is represented by the secondary structure shown and for the numbers that indicate the residue at the start of each line.
- `-r` or `--residues-per-line`: The number of amino acid residues that should fit on a single line (default is 50). The width r of a single residue is calculated from this number (n), from the indention i at the beginning of each line and from the line width w by $r = \frac{w-i}{n}$. Therefore, r is 0.25 cm for the default values.
- `-l` or `--line-distance`: A number that gives the distance between lines (default is .5). In an image comparing secondary structure predictions, each prediction resides on its own line; if the length of the comparison exceeds line width minus the indention, is it broken into several blocks which have a distance given by
- `-b` or `--block-distance` (default is 1).

In our first example, the program was invoked by

```
cpssp -s ColSequences.fasta -u ColStructures.fasta
  -w 14 -i 1.5 -t 20 -o Collagenases
```

to produce a total of 2 images comparing the collagenases. For the second example, the command was

```
cpssp -u ColGPredictions.fasta -w 14 -i 1.5 -t 10
  -l .25 -b .5 -o ColGPredictions
```

to produce a total of 3 images comparing the predicted structures for ColG from *C. histolyticum*, and for the third example,

```
cpssp -u ColTPrediction.fasta -w 14 -i 0 -r 56 -t 20
  -b .25 -o ColTPrediction
```

produced a single output file.

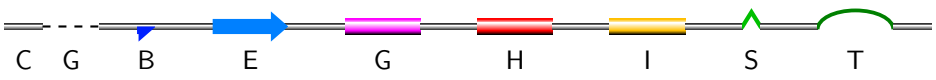
4 Usage of the cpssp Package

4.1 Package Options

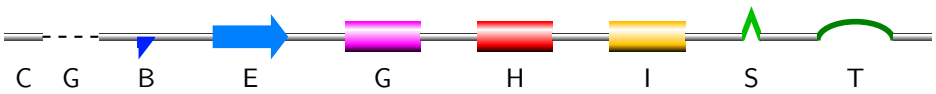
The image itself is plotted in LaTeX using the Ti*k*Z and PGF packages [10]. In order to use the output from `cpssp` in your LaTeX document, you have to load the corresponding package by `\usepackage{cpssp}` in the preamble. The following options are known to the package; their values are set by using the ⟨*key*⟩=⟨*value*⟩ syntax.

`style` governs the general appearance of the representation. Currently four styles are defined:

- `small` is the default value and gives an image as in fig. 1. Note that the default lengths in the `cpssp` program have been optimized for this style, so you may need to specify different values when using one of the styles below (as was done in the examples).

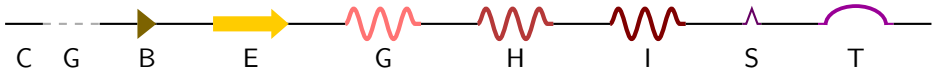


- `large` is similar to the `small` style, but the secondary structure elements are about twice as large (see fig. 3). This style is useful for short sequences.



- `graylines` is the style to use for black and white images when space runs out (see fig. 2).

- `pdb` is similar to the output of the RCSB Protein Data Bank (under "Sequence / Structure Details"; see fig. 4).



`labels` can either be `true` or `false` and determines if the secondary structure names should be shown at the beginning of each line. It may be appropriate to turn off the labels if a single secondary structure of a single protein is shown.

`numbers` is also a bool option and displays or hides the numbers at to left and right of each line which indicate the first an last residue in this line, respectively.

`labelformat` controls the formatting of the structure labels; its default definition is `\sffamily\scriptsize`. The last control sequence may take a single mandatory argument, so `labelformat=\textit` is valid.

`numberformat` formats the numbers that indicate the first and last residue on each line, default is `\sffamily\tiny`. The last control sequence may again take a single mandatory argument.

`threehelixname` and `pihelixname` hold macros that print the name of the 3_{10}-helix and the π-helix, respectively, above the corresponding secondary structure segments when using the `graylines` style. These macros are `\tiny$3_{10}$` and `\tiny$\pi$` by default, respectively.

`\cpsspformat` These options can be changed at any place within the document by changing their values within the argument of `\cpsspformat{`⟨*options*⟩`}`.

4.2 Including `cpssp` Files

The output LaTeX files from `cpssp` may be included either directly in the text or inside a float environment such as `figure` by using one of the following:

`\cpsspinput`
- `\cpsspinput[`⟨*options*⟩`]{`⟨*filename*⟩`}`
  This command takes a single mandatory argument, the name of the `cpssp` output file without the `.tex` extension. The optional argument takes the same options as the package and allows to change the format of the representation locally.

`cpsspimage`
- `\begin{cpsspimage}[`⟨*options*⟩`]{`⟨*filename*⟩`}`
  ...
  `\end{cpsspimage}`
  This is the environment version of `\cpsspinput` with the same optional and mandatory argument. The environment allows you to use additional graphical elements (e.g., comments and arrows) in the secondary structure image by using standard Ti*k*Z syntax (this was done in the small images beneath the style descriptions where additional letters indicate the secondary structure element which is represented by each segment).

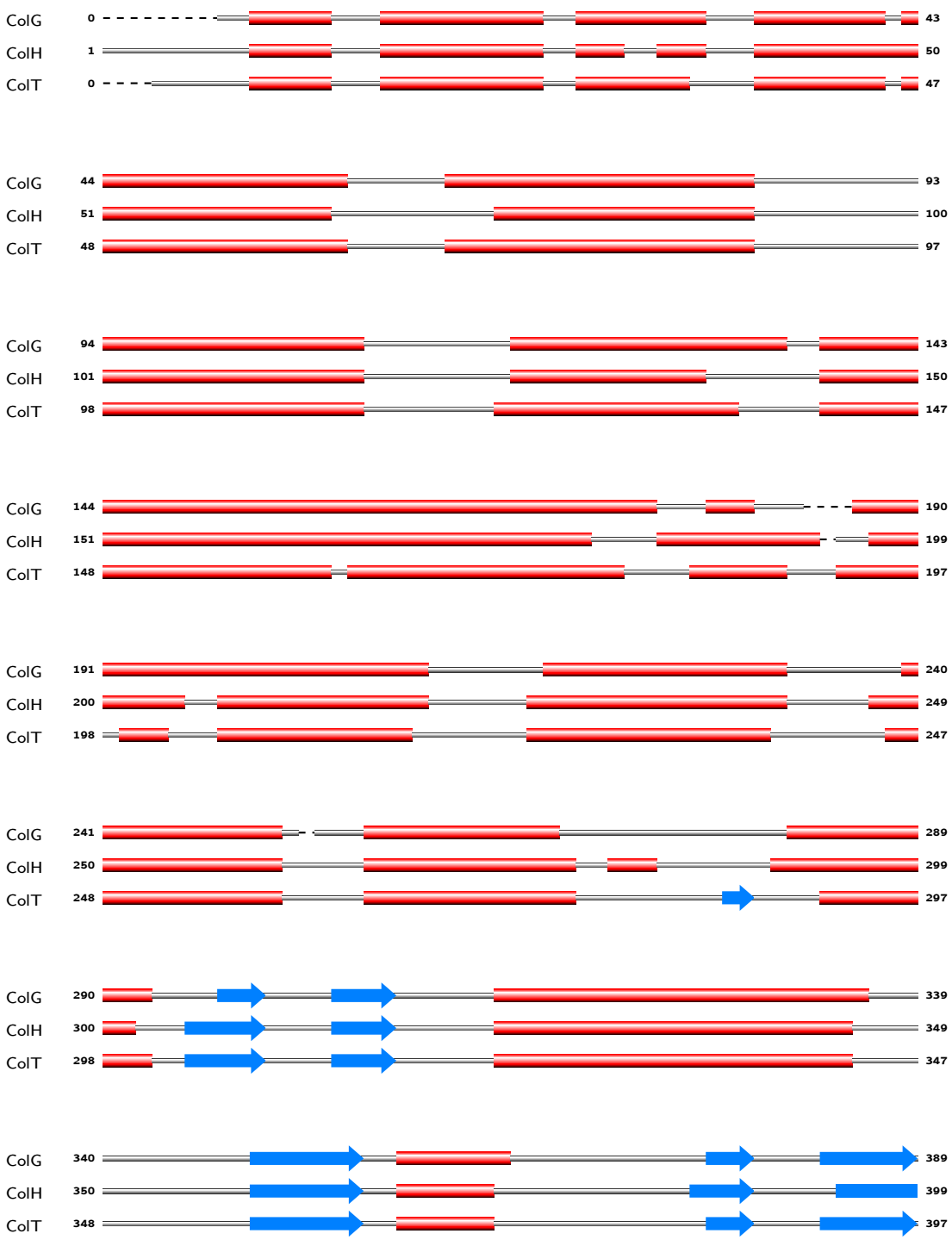


Figure 1: A secondary structure comparison from `Collagenases0.tex` using the `small` style and the default format for both residue numbers and labels.

Figure 2: A secondary structure comparison from `ColGPredictions1.tex` using the `graylines` style. Residue numbers are turned off by the option `numbers=false`. The label format is changed to `\sffamily\tiny\textit`.

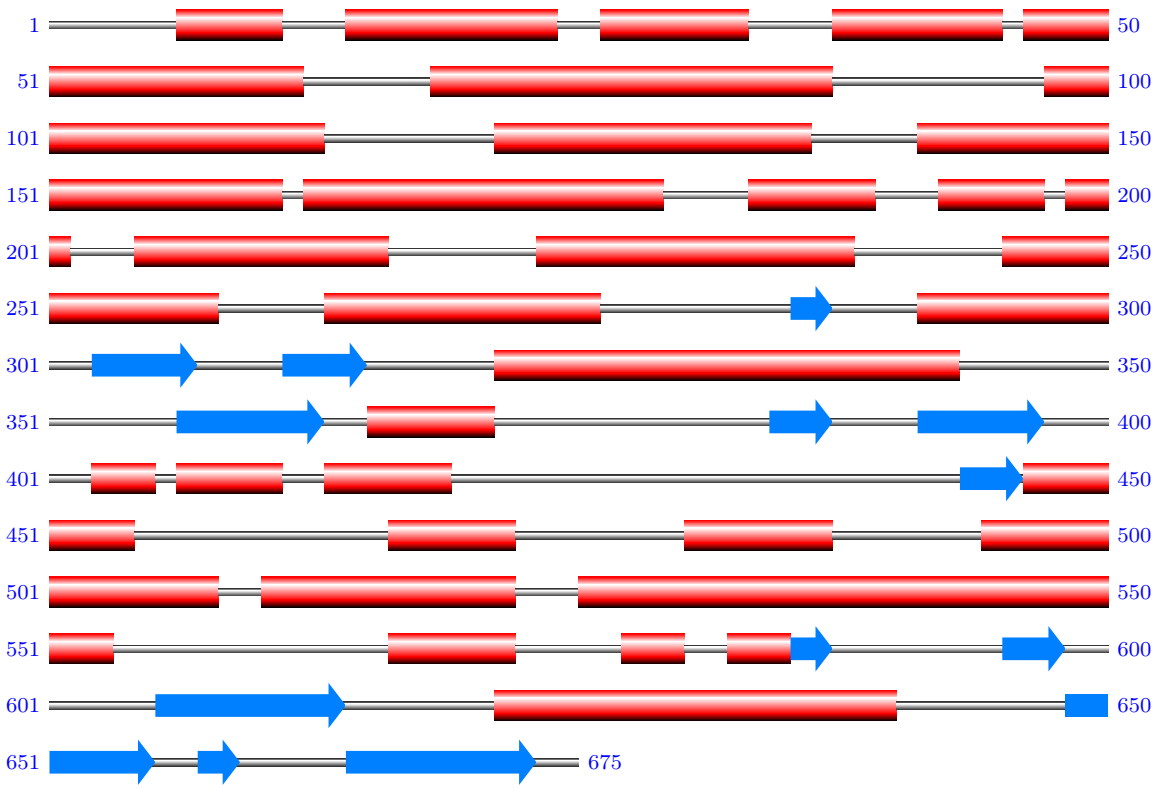


Figure 3: A secondary structure prediction from `ColTPrediction0.tex` using the `large` style. Labels are hidden by `labels=off`. The number format is `\footnotesize\color{blue}`.

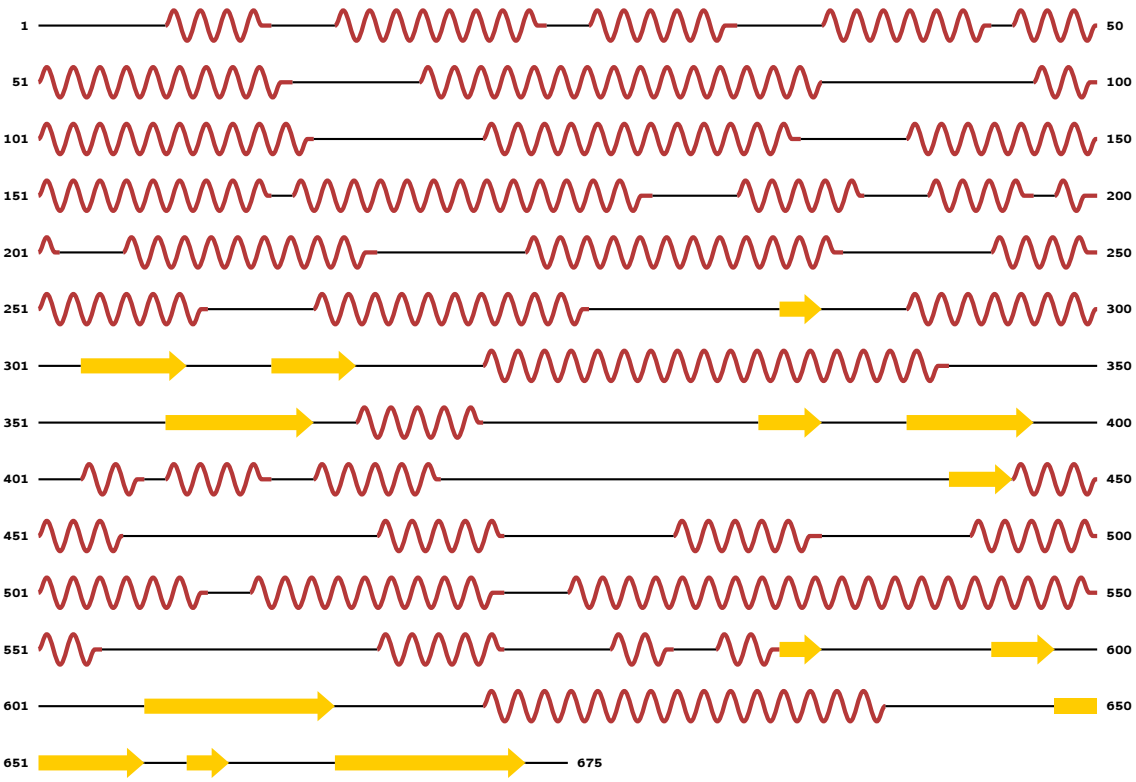


Figure 4: A secondary structure prediction from `ColTPrediction0.tex` using the `pdb` style. Labels are hidden by `labels=off`.

References

[1] Frishman, D. & Argos, P. Knowledge-based protein secondary structure assignment. *Proteins* **23**, 566–579 (1995).

[2] McGuffin, L. J., Bryson, K. & Jones, D. T. The PSIPRED protein structure prediction server. *Bioinformatics* **16**, 404–405 (2000).

[3] Cole, C., Barber, J. D. & Barton, G. J. The Jpred 3 secondary structure prediction server. *Nucleic Acids Res.* **36**, W197–W201 (2008).

[4] Cheng, H., Sen, T. Z. *et al.* Consensus Data Mining (CDM) Protein Secondary Structure Prediction Server: combining GOR V and Fragment Database Mining (FDM). *Bioinformatics* **23**, 2628–2630 (2007).

[5] Pollastri, G., Przybylski, D. *et al.* Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins* **47**, 228–235 (2002).

[6] Adamczak, R., Porollo, A. & Meller, J. Combining prediction of secondary structure and solvent accessibility in proteins. *Proteins* **59**, 467–475 (2005).

[7] Larkin, M. A., Blackshields, G. *et al.* Clustal W and Clustal X version 2.0. *Bioinformatics* **23**, 2947–2948 (2007).

[8] Ducka, P., Eckhard, U. *et al.* A universal strategy for high-yield production of soluble and functional clostridial collagenases in *E. coli*. *Appl. Microbiol. Biotechnol.* in print (2009).

[9] Waterhouse, A. M., Procter, J. B. *et al.* Jalview Version 2–a multiple sequence alignment editor and analysis workbench. *Bioinformatics* **25**, 1189–1191 (2009).

[10] Tantau, T. The Ti*k*Z and pgf Packages (2008).
`http://tug.ctan.org/tex-archive/graphics/pgf/base/`

5 Implementation

5.1 Package Options

The packages ifthen and calc are used for conditionals and calculation of some coordinates, respectively, while kvoptions handles the options specified when the package is loaded. These options and their default values are described in § 4.

```
\RequirePackage{ifthen,calc}
\RequirePackage{kvoptions}
  \SetupKeyvalOptions{family=Cps, prefix=Cps@}
  \DeclareStringOption[small]{style}
  \DeclareBoolOption[true]{labels}
  \DeclareBoolOption[true]{numbers}
  \DeclareStringOption[\sffamily\scriptsize]{labelformat}
  \DeclareStringOption[\sffamily\tiny]{numberformat}
  \DeclareStringOption[\tiny$3_{10}$]{threehelixname}
  \DeclareStringOption[\tiny$\pi$]{pihelixname}
\ProcessKeyvalOptions*
```

5.2 Command Options

The options which can be set in the optional arguments of `\cpsspinput` and the `cpsspimage` environment and in the mandatory argument of `\cpsspformat` are the same as when loading the package and are set up with commands provided by the keyval package (which is not loaded explicitly because kvoptions takes care of that).

```
\define@key{cpssp}{style}{\cpssp@style{#1}}
\define@key{cpssp}{numbers}{%
  \ifthenelse{\equal{#1}{true}}%
    {\setboolean{Cps@numbers}{true}}%
    {\setboolean{Cps@numbers}{false}}%
}
\define@key{cpssp}{labels}{%
  \ifthenelse{\equal{#1}{true}}%
    {\setboolean{Cps@labels}{true}}%
    {\setboolean{Cps@labels}{false}}%
}
\define@key{cpssp}{labelformat}%
  {\renewcommand\cpssp@labelformat{#1}}
\define@key{cpssp}{numberformat}%
  {\renewcommand\cpssp@numberformat{#1}}
\define@key{cpssp}{threehelixname}%
  {\renewcommand\cpssp@threehelix{#1}}
\define@key{cpssp}{pihelixname}%
  {\renewcommand\cpssp@pihelix{#1}}
```

5.3 Ti*k*Z Configuration

The Ti*k*Z libraries `positioning` and `decorations.pathmorphing` are needed for additional positioning options for the `\node` command and for drawing of the helices in `pdb` style, respectively.

```
\RequirePackage{tikz}
```

```
\usetikzlibrary{positioning}
\usetikzlibrary{decorations.pathmorphing}
```

The package defines some additional colors which govern the appearance of the secondary structure elements in the `small`/`large` (the first nine) and `pdb` style (the last nine). In each color, the last letter indicates the structure element in which the color is used. You may change these colors by the commands provided by the xcolor package anywhere within the document. Note that the shades of gray used in the `graylines` style can not be changed.

```
\colorlet    {CpsspColorX}        {black}
\providecolor{CpsspColorB}   {rgb}{0,.125,1}
\colorlet    {CpsspColorC}        {black!80}
\providecolor{CpsspColorE}   {rgb}{0,.5,1}
\providecolor{CpsspColorG}   {rgb}{1,0,1}
\providecolor{CpsspColorH}   {rgb}{1,0,0}
\providecolor{CpsspColorI}   {rgb}{1,.75,0}
\providecolor{CpsspColorS}   {rgb}{0,.75,0}
\providecolor{CpsspColorT}   {rgb}{0,.5,0}
\colorlet    {CpsspColorPdbX}     {black!33}
\providecolor{CpsspColorPdbB}{rgb}{.49,.39,0}
\colorlet    {CpsspColorPdbC}     {black}
\providecolor{CpsspColorPdbE}{rgb}{1,.8,0}
\providecolor{CpsspColorPdbG}{rgb}{1,.45,.45}
\providecolor{CpsspColorPdbH}{rgb}{.71,.22,.22}
\providecolor{CpsspColorPdbI}{rgb}{.5,0,0}
\providecolor{CpsspColorPdbS}{rgb}{.4,0,.39}
\providecolor{CpsspColorPdbT}{rgb}{.6,0,.59}
```

5.4 User Interface, Part 1

The following commands are found in the output of the `cpssp` program, but may also be used anywhere within a `tikzpicture` environment.

`\cpsspGap` `\cpsspBridge` `\cpsspCoil` `\cpsspSheet` `\cpsspSheetT` `\cpsspThreeTenHelix` `\cpsspAlphaHelix` `\cpsspPiHelix` `\cpsspBend` `\cpsspTurn`

These nine macros each take three mandatory arguments, the first specifying the y coordinate and the second and third specifying the start and end x coordinates, respectively (all coordinates should be given in centimeters). Each macro basically expands to another control sequence which contains the Ti*k*Z drawing commands (see below). The names clearly indicate which secondary structure element is produced. `\cpsspSheetT` draws the N-terminus of a β-sheet whithout the C-terminal arrowhead; this is useful for β-sheets that continue past the end of a line.

```
\newcommand\cpsspGap{\cpssp@makeX}
\newcommand\cpsspBridge{\cpssp@makeB}
\newcommand\cpsspCoil{\cpssp@makeC}
\newcommand\cpsspSheet{\cpssp@makeE}
\newcommand\cpsspSheetT{\cpssp@makeET}
\newcommand\cpsspThreeTenHelix{\cpssp@makeG}
\newcommand\cpsspAlphaHelix{\cpssp@makeH}
\newcommand\cpsspPiHelix{\cpssp@makeI}
\newcommand\cpsspBend{\cpssp@makeS}
\newcommand\cpsspTurn{\cpssp@makeT}
```

`\cpsspLabel` The `\cpsspLabel` macro first checks if the names of the structures should be printed at the beginning of each line and then either calls the appropriate internal

macro or does nothing. The first mandatory argument takes the y coordinate, the second the label text.

```
\newcommand\cpsspLabel[2]{%
  \ifthenelse{\boolean{Cps@labels}}{\cpssp@makeLabel{#1}{#2}}{}%
}
```

\cpsspStartRes
\cpsspStartRes

These macros, which print the number of the first and the last residue in the line, first check if the numbers should appear at all and then either call the internal command or do nothing. The three arguments specify y coordinate, x coordinate and label text, respectively.

```
\newcommand\cpsspStartRes[3]{%
  \ifthenelse{\boolean{Cps@numbers}}{\cpssp@makeStartRes{#1}{#2}{#3}}{}%
}
\newcommand\cpsspEndRes[3]{%
  \ifthenelse{\boolean{Cps@numbers}}{\cpssp@makeEndRes{#1}{#2}{#3}}{}%
}
```

5.5 Styles

\cpssp@makeX
\cpssp@makeB
\cpssp@makeC
\cpssp@makeE
\cpssp@makeET
\cpssp@makeG
\cpssp@makeH
\cpssp@makeI
\cpssp@makeS
\cpssp@makeT
\cpssp@makeLabel
\cpssp@makeStartRes
\cpssp@makeEndRes

First we define some empty internal macros which will be redefined when a particular style is chosen. These are the macros that are called by the user-level control sequences described above.

```
\newcommand\cpssp@makeX{}
\newcommand\cpssp@makeB{}
\newcommand\cpssp@makeC{}
\newcommand\cpssp@makeE{}
\newcommand\cpssp@makeET{}
\newcommand\cpssp@makeG{}
\newcommand\cpssp@makeH{}
\newcommand\cpssp@makeI{}
\newcommand\cpssp@makeS{}
\newcommand\cpssp@makeT{}
\newcommand\cpssp@makeLabel{}
\newcommand\cpssp@makeStartRes{}
\newcommand\cpssp@makeEndRes{}
```

\cpssp@labelformat
\cpssp@numberformat

The two macros below are used for formatting the structure labels and first/last residue numbers. As can be seen from the code, whenever they are used, the argument that follows is surrounded by braces. This allows one to use control sequences with a single mandatory argument at the end of the definition of both `\cpssp@labelformat` and `\cpssp@numberformat`.

```
\newcommand\cpssp@labelformat{}
\newcommand\cpssp@numberformat{}
```

\cpssp@threehelix
\cpssp@pihelix

These two commands contain the labels that are printed above 3_{10}- and π-helices in the `graylines` style. They are set by the corresponding options.

```
\newcommand\cpssp@threehelix{}
\newcommand\cpssp@pihelix{}
```

\cpsstyle@small

The `small` style is shown in fig. 1. The internal drawing commands described above are redefined so that they expand to TikZ drawing commands. Note that coordinates are calculated by using the syntax from the calc package.

```
\newcommand\cpsstyle@small{%
  \renewcommand\cpssp@makeX[3]{%
    \draw[dashed, thick, color=CpsspColorX]%
      (##2, ##1+.2) -- (##3, ##1+.2);%
  }%
  \renewcommand\cpssp@makeB[3]{%
    \cpssp@makeC{##1}{##2}{##3}%
    \fill[color=CpsspColorB]%
      (##2, ##1+.1) -- (##3-.23, ##1+.1) -- (##3-.23, ##1) --%
      (##3, ##1+.2) -- (##2, ##1+.2);%
  }%
  \renewcommand\cpssp@makeC[3]{%
    \shade[top color=CpsspColorC, bottom color=white]%
      (##2, ##1+.2375) rectangle (##3, ##1+.21875);%
    \shade[top color=white, bottom color=CpsspColorC]%
      (##2, ##1+.21875) rectangle (##3, ##1+.1625);%
  }%
  \renewcommand\cpssp@makeE[3]{%
    \cpssp@makeC{##1}{##3-.1}{##3}%
    \fill[color=CpsspColorE]%
      (##2, ##1+.1) -- (##3-.23, ##1+.1) -- (##3-.23, ##1) --%
      (##3, ##1+.2) -- (##3-.23, ##1+.4) -- (##3-.23, ##1+.3)--%
      (##2, ##1+.3);%
  }%
  \renewcommand\cpssp@makeET[3]{%
    \fill[color=CpsspColorE]%
      (##2, ##1+.1) rectangle (##3, ##1+.3);%
  }%
  \renewcommand\cpssp@makeG[3]{%
    \shade[top color=CpsspColorG, bottom color=white]%
      (##2, ##1+.3) rectangle (##3, ##1+.25);%
    \shade[top color=white, bottom color=CpsspColorG]%
      (##2, ##1+.25) rectangle (##3, ##1+.15);%
    \shade[top color=CpsspColorG, bottom color=black]%
      (##2, ##1+.15) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeH[3]{%
    \shade[top color=CpsspColorH, bottom color=white]%
      (##2, ##1+.3) rectangle (##3, ##1+.25);%
    \shade[top color=white, bottom color=CpsspColorH]%
      (##2, ##1+.25) rectangle (##3, ##1+.15);%
    \shade[top color=CpsspColorH, bottom color=black]%
      (##2, ##1+.15) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeI[3]{%
    \shade[top color=CpsspColorI, bottom color=white]%
      (##2, ##1+.3) rectangle (##3, ##1+.25);%
    \shade[top color=white, bottom color=CpsspColorI]%
      (##2, ##1+.25) rectangle (##3, ##1+.15);%
    \shade[top color=CpsspColorI, bottom color=black]%
      (##2, ##1+.15) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeS[3]{%
    \begin{scope}%
```

```
      \clip (##2, ##1) rectangle (##3, ##1+1);%
      \draw[ultra thick, color=CpsspColorS, line cap=round]%
        (##2, ##1+.19) -- ({(##2+##3)*\real{.5}}, ##1+.39) --%
        (##3, ##1+.19);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeT[3]{%
    \begin{scope}%
      \clip (##2, ##1) rectangle (##3, ##1+1);%
      \draw[line width=.05cm, color=CpsspColorT]%
        (##2, ##1+.1625) arc (180:0:{(##3-##2)*\real{.5}} and .25);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeLabel[2]{%
    \node at (0, ##1) [inner sep=0pt, above right=.075 and 0]%
      {\cpssp@labelformat{##2}};%
  }%
  \renewcommand\cpssp@makeStartRes[3]{%
    \node at (##2, ##1+.2) [left]%
      {\cpssp@numberformat{##3}};%
  }%
  \renewcommand\cpssp@makeEndRes[3]{%
    \node at (##2, ##1+.2) [right]%
      {\cpssp@numberformat{##3}};%
  }%
}
```

`\cpsstyle@large` The `large` style is shown in fig. 3.

```
\newcommand\cpsstyle@large{%
  \renewcommand\cpssp@makeX[3]{%
    \draw[dashed, thick, color=CpsspColorX]%
      (##2, ##1+.3) -- (##3, ##1+.3);%
  }%
  \renewcommand\cpssp@makeB[3]{%
    \cpssp@makeC{##1}{##2}{##3}%
    \fill[color=CpsspColorB]%
      (##2, ##1+.15) -- (##3-.23, ##1+.15) -- (##3-.23, ##1) --%
      (##3, ##1+.3) -- (##2, ##1+.3);%
  }%
  \renewcommand\cpssp@makeC[3]{%
    \shade[top color=CpsspColorC, bottom color=white]%
      (##2, ##1+.35) rectangle (##3, ##1+.325);%
    \shade[top color=white, bottom color=CpsspColorC]%
      (##2, ##1+.325) rectangle (##3, ##1+.25);%
  }%
  \renewcommand\cpssp@makeE[3]{%
    \cpssp@makeC{##1}{##3-.1}{##3}%
    \fill[color=CpsspColorE]%
      (##2, ##1+.15) -- (##3-.23, ##1+.15) -- (##3-.23, ##1) --%
      (##3, ##1+.3) -- (##3-.23, ##1+.6) -- (##3-.23, ##1+.45) --%
      (##2, ##1+.45);%
  }%
  \renewcommand\cpssp@makeET[3]{%
    \fill[color=CpsspColorE]%
```

```
      (##2, ##1+.15) rectangle (##3, ##1+.45);%
  }%
  \renewcommand\cpssp@makeG[3]{%
    \shade[top color=CpsspColorG, bottom color=white]%
      (##2, ##1+.5) rectangle (##3, ##1+.4);%
    \shade[top color=white, bottom color=CpsspColorG]%
      (##2, ##1+.4) rectangle (##3, ##1+.2);%
    \shade[top color=CpsspColorG, bottom color=black] %
      (##2, ##1+.2) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeH[3]{%
    \shade[top color=CpsspColorH, bottom color=white]%
      (##2, ##1+.5) rectangle (##3, ##1+.4);%
    \shade[top color=white, bottom color=CpsspColorH]%
      (##2, ##1+.4) rectangle (##3, ##1+.2);%
    \shade[top color=CpsspColorH, bottom color=black] %
      (##2, ##1+.2) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeI[3]{%
    \shade[top color=CpsspColorI, bottom color=white]%
      (##2, ##1+.5) rectangle (##3, ##1+.4);%
    \shade[top color=white, bottom color=CpsspColorI]%
      (##2, ##1+.4) rectangle (##3, ##1+.2);%
    \shade[top color=CpsspColorI, bottom color=black] %
      (##2, ##1+.2) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeS[3]{%
    \begin{scope}%
      \clip (##2, ##1) rectangle (##3, ##1+1);%
      \draw[line width=.075cm, color=CpsspColorS, line cap=round]%
        (##2, ##1+.275) -- ({(##2+##3)*\real{.5}}, ##1+.6) --%
        (##3, ##1+.275);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeT[3]{%
    \begin{scope}%
      \clip (##2, ##1) rectangle (##3, ##1+1);%
      \draw[line width=.075cm, color=CpsspColorT]%
        (##2, ##1+.25) arc (180:0:{(##3-##2)*\real{.5}} and .25);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeLabel[2]{%
    \node at (0, ##1+.3) [inner sep=0pt, right]%
      {\cpssp@labelformat{##2}};%
  }%
  \renewcommand\cpssp@makeStartRes[3]{%
    \node at (##2, ##1+.3) [left]%
      {\cpssp@numberformat{##3}};%
  }%
  \renewcommand\cpssp@makeEndRes[3]{%
    \node at (##2, ##1+.3) [right]%
      {\cpssp@numberformat{##3}};%
  }%
}
```

\cpsstyle@graylines The graylines style is shown in fig. 2.

```
\newcommand\cpsstyle@graylines{
  \renewcommand\cpssp@makeX[3]{%
    \draw[dashed, thick, color=black!33]%
      (##2, ##1+.05) -- (##3, ##1+.05);%
  }%
  \renewcommand\cpssp@makeB[3]{%
    \filldraw[very thin, color=black!25, draw=black]%
      (##2, ##1+.0125) -- (##3-.075, ##1+.0125) --%
      (##3-.075, ##1-.0375) -- (##3, ##1+.05) --%
      (##2, ##1+.05) -- (##2, ##1+.0125);%
  }%
  \renewcommand\cpssp@makeC[3]{%
    \draw[thick, color=black]%
      (##2, ##1+.05) -- (##3, ##1+.05);%
  }%
  \renewcommand\cpssp@makeE[3]{%
    \filldraw[very thin, color=black!25, draw=black]%
      (##2, ##1+.0125) -- (##3-.075, ##1+.0125) --%
      (##3-.075, ##1-.0375) -- (##3, ##1+.05) --%
      (##3-.075, ##1+.1375) -- (##3-.075, ##1+.0875) --%
      (##2, ##1+.0875) -- (##2, ##1+.0125);%
  }%
  \renewcommand\cpssp@makeET[3]{%
    \filldraw[very thin, color=black!25, draw=black]%
      (##2, ##1+.0125) rectangle (##3, ##1+.0875);%
  }%
  \renewcommand\cpssp@makeG[3]{%
    \filldraw[thin, color=black!75, draw=black] %
      (##2, ##1) rectangle (##3, ##1+.1);%
    \node at ({(##2+##3)*\real{.5}}, ##1) [anchor=south]%
      {\cpssp@threehelix};%
  }%
  \renewcommand\cpssp@makeH[3]{%
    \filldraw[thin, color=black!75, draw=black] %
      (##2, ##1) rectangle (##3, ##1+.1);%
  }%
  \renewcommand\cpssp@makeI[3]{%
    \filldraw[thin, color=black!75, draw=black] %
      (##2, ##1) rectangle (##3, ##1+.1);%
    \node at ({(##2+##3)*\real{.5}}, ##1) [anchor=south]%
      {\cpssp@pihelix};%
  }%
  \renewcommand\cpssp@makeS[3]{%
    \begin{scope}%
      \clip (##2, ##1) rectangle (##3, ##1+1);%
      \draw[thick, color=black!75, line cap=round]%
        (##2, ##1+.05) -- ({(##2+##3)*\real{.5}}, ##1+.2) --%
        (##3, ##1+.05);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeT[3]{%
    \begin{scope}%
      \clip (##2, ##1) rectangle (##3, ##1+1);%
```

```
      \draw[thick, color=black!75]%
        (##2, ##1+.035) arc (180:0:{(##3-##2)*\real{.5}} and .2);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeLabel[2]{%
    \node at (0, ##1+.05) [inner sep=0pt,right]%
      {\cpssp@labelformat{##2}};%
  }%
  \renewcommand\cpssp@makeStartRes[3]{%
    \node at (##2, ##1+.05) [left]%
      {\cpssp@numberformat{##3}};%
  }%
  \renewcommand\cpssp@makeEndRes[3]{%
    \node at (##2, ##1+.05) [right]%
      {\cpssp@numberformat{##3}};%
  }%
}
```

\cpsstyle@pdb The pdb style is shown in fig. 4.

```
\newcommand\cpsstyle@pdb{
  \renewcommand\cpssp@makeX[3]{%
    \draw[dashed, thick, color=CpsspColorPdbX]%
      (##2, ##1+.2) -- (##3, ##1+.2);%
  }%
  \renewcommand\cpssp@makeB[3]{%
    \cpssp@makeC{##1}{##3-.1}{##3}%
    \fill[color=CpsspColorPdbB]%
      (##2, ##1) -- (##3, ##1+.2) -- (##2, ##1+.4);
  }%
  \renewcommand\cpssp@makeC[3]{%
    \draw[thick, color=CpsspColorPdbC]%
      (##2, ##1+.2) -- (##3,##1+.2);%
  }%
  \renewcommand\cpssp@makeE[3]{%
    \cpssp@makeC{##1}{##3-.1}{##3}%
    \fill[color=CpsspColorPdbE]%
      (##2, ##1+.1) -- (##3-.23, ##1+.1) -- (##3-.23, ##1) --%
      (##3, ##1+.2) -- (##3-.23, ##1+.4) -- (##3-.23, ##1+.3)--%
      (##2, ##1+.3);%
  }%
  \renewcommand\cpssp@makeET[3]{%
    \fill[color=CpsspColorPdbE]%
      (##2, ##1+.1) rectangle (##3, ##1+.3);%
  }%
  \renewcommand\cpssp@makeG[3]{%
    \draw[ultra thick, color=CpsspColorPdbG, decorate,%
      decoration={snake, amplitude=.2cm}]%
      (##2, ##1+.2) -- (##3, ##1+.2);%
  }%
  \renewcommand\cpssp@makeH[3]{%
    \draw[ultra thick, color=CpsspColorPdbH, decorate,%
      decoration={snake, amplitude=.2cm}]%
      (##2, ##1+.2) -- (##3, ##1+.2);%
  }%
```

```
  \renewcommand\cpssp@makeI[3]{%
    \draw[ultra thick, color=CpsspColorPdbI, decorate,%
      decoration={snake, amplitude=.2cm}]%
      (##2, ##1+.2) -- (##3, ##1+.2);%
  }%
  \renewcommand\cpssp@makeS[3]{%
    \begin{scope}%
      \draw[thick, color=CpsspColorPdbS]
        (##2, ##1+.2) -- (##2+.05, ##1+.2) --%
        ({(##2+##3)*\real{.5}}, ##1+.4) -- (##3-.05, ##1+.2) --%
        (##3, ##1+.2);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeT[3]{%
    \begin{scope}[color=CpsspColorPdbT]%
      \draw[thick]%
        (##2, ##1+.2) -- (##2+.1, ##1+.2);%
      \draw[line width=.05cm]%
        (##2+.1,##1+.185) arc (180:0:{(##3-##2-.2)*\real{.5}} and .25);%
      \draw[thick]%
        (##3-.1, ##1+.2) -- (##3, ##1+.2);%
    \end{scope}%
  }%
  \renewcommand\cpssp@makeLabel[2]{%
    \node at (0, ##1) [inner sep=0pt, above right=.075 and 0]%
      {\cpssp@labelformat{##2}};%
  }%
  \renewcommand\cpssp@makeStartRes[3]{%
    \node at (##2, ##1+.2) [left]%
      {\cpssp@numberformat{##3}};%
  }%
  \renewcommand\cpssp@makeEndRes[3]{%
    \node at (##2, ##1+.2) [right]%
      {\cpssp@numberformat{##3}};%
  }%
}
```

`\cpssp@style` The `cpssp@style` macro is used to change the style internally. It is called with the style name as the single mandatory argument and first checks if this style is known (i.e., the macro `\cpsstyle@`⟨*style name*⟩ is defined) and executes this macro if possible; otherwise, a package warning is issued and the `small` style is used.

```
\newcommand\cpssp@style[1]{
  \@ifundefined{cpsstyle@#1}{%
    \PackageWarning{myclassicthesis}{Style ``\Cps@style'' undefined%
      \MessageBreak using ``small'' instead}%
    \cpsstyle@small%
  }{%
    \csname cpsstyle@#1\endcsname%
  }%
}
```

5.6 User Interface, Part 2

\cpsspformat Now the three main user commands are defined. \cpsspformat takes one mandatory argument which is a list of comma-separated options in keyval format; these options change the appearance of the cpssp output.

```
\newcommand\cpsspformat[1]{\setkeys{cpssp}{#1}}
```

When the package is loaded by \usepackage[⟨*options*⟩]{cpssp}, \cpsspformat is called and the options are set to the values specified in the optional argument of \usepackage.

```
\cpsspformat{%
  style=\Cps@style,%
  numberformat=\Cps@numberformat,%
  labelformat=\Cps@labelformat,%
  threehelixname=\Cps@threehelixname,%
  pihelixname=\Cps@pihelixname%
}
```

\cpsspinput \cpsspinput is basically a shorthand for the more powerful cpsspimage environment (see below). The optional argument takes the same option list as \cpsspformat, the mandatory argument takes the name of the input file without the .tex extension.

```
\newcommand\cpsspinput[2][]{\begin{cpsspimage}[#1]{#2}\end{cpsspimage}}
```

cpsspimage The cpsspimage environment takes the same arguments as the \cpsspinput commands. \begin{cpsspimage} first begins a tikzpicture environment and then locally changes the appearance of the secondary structure image which is finally loaded by \input. As tikzpicture remains open until \end{cpsspimage}, it is possible to add further graphical elements to the image by using standard Ti*k*Z commands.

```
\newenvironment{cpsspimage}[2][]%
  {\begin{tikzpicture}\cpsspformat{#1}\input{#2.tex}}%
  {\end{tikzpicture}}
```

Change History

v1.0

General: Initial version 1

Index

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

E