Contribution Title

Name of First Author and Name of Second Author

Abstract Each chapter should be preceded by an abstract (no more than 200 words) that gives the name of the module and summarizes the purpose for which it can be used. The following is a plausible description of a module from MRST:
The multiscale restricted-smoothing basis (MsRSB) method is the current state-of-the-art within multiscale methods. MsRSB is very robust and versatile can either be used as an approximate coarse-scale solver having mass-conservative subscale resolution, or as an iterative fine-scale solver that will provide mass-conservative solutions for any given tolerance. The performance of the method has been demonstrated on incompressible 2-phase flow, compressible 2 and 3-phase black oil models, as well as compositional models. It has also been demonstrated that the method can utilize combinations of multiple prolongation operators, e.g., corresponding to coarse grids with different resolutions, adapting to geological features, adapting to wells, or moving displacement fronts. This chapter explains the basic ideas of the MsRSB method, including methods to construct coarse partitions, prolongation, and restriction operators, reduction of the fine-scale flow equations to a coarse-scale system, and formulation as part of a two-level iterative solver. We outline the key functions in the module and show various examples of how the method can be used as an iterative solver for incompressible and compressible flow on 2D rectilinear grids, unstructured grids, and 3D stratigraphic grids.
The abstract will also appear *online* at `www.SpringerLink.com` and be available with unrestricted access. This allows unregistered users to read the abstract as a teaser for the complete chapter.

Name of First Author
Name, Address of Institute, e-mail: `name@email.address`

Name of Second Author
Name, Address of Institute e-mail: `name@email.address`

1 Organizing Your Content into Sections

Use the template *authorsample.tex* together with the document class `svmult.cls` to style the various elements of your chapter content.

1.1 Subsection Heading

Instead of simply listing headings of different levels, we recommend to let every heading be followed by at least a short passage of text. Further on, please use the LaTeX automatism for all your cross-referencess and citations. That is, use `\ref{<mylabel>}` for cross-references and `\cite{<mylabel>}` for bibliographic references. Try to make your reference- and citation labels as unique as possible to avoid compilation conflicts with other chapters in the book, e.g., by adding the name of your module to each label `\label{sec:msrsb-mylabel}`, and so on.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

1.1.1 Subsubsection Heading

The same rule applies here: always let a heading be followed by a short text.

Paragraph Heading

The same rule applies here: always let a heading be followed by a short text.

Subparagraph Heading

The same rule applies here: always let a heading be followed by at least a short passage of text.

Run-in Heading Boldface Version Some arbitrary text that follows after the run-in heading to make a sentence that covers more than a single line.

Run-in Heading Boldface and Italic Version A collection of words comes here to make the text following the heading spread out over more than a single line.

Run-in Heading Displayed Version
And this should be the last sectioning command.

1.2 Emphasized Paragraphs

If you want to emphasize complete paragraphs of texts we recommend to use the newly defined class option `graybox` and the newly defined environment `svgraybox`. This will produce a 15 percent screened box 'behind' your text.

The `svmult` class also offers other environments to emphasize certain paragraphs

```
\begin{trailer}{Trailer Head}  ...   \end{trailer}
\begin{question}{Questions}    ...   \end{question}
\begin{important}{Important}   ...   \end{important}
\begin{warning}{Attention}     ...   \end{warning}
\begin{tips}{Tips}             ...   \end{tips}
\begin{overview}{Overview}     ...   \end{overview}
\begin{backgroundinformation}{Background Information}
  ...
\end{backgroundinformation}
```

We strongly suggest that you avoid using any of these. In particular, we request that you **do not use** the environment for computer code:

```
\begin{programcode}{Program Code}
    \begin{verbatim}...\end{verbatim}
\end{programcode}
```

but instead use the setup defined in `mrstbook.sty`, which is discussed in more detail in Section 5 on page 9.

2 Lists

There are three different types of list environments: numbered lists, unnumbered lists, and definition lists.

Numbered Lists
For typesetting numbered lists we recommend to use the `enumerate` environment – it will automatically rendered in line with the preferred layout.

1. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
    a. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
    b. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.

2. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.

Unnumbered Lists
For unnumbered list we recommend to use the `itemize` environment – it will automatically be rendered in line with the preferred layout.

- Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development, cf. Table 1.
  - Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
  - Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
- Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.

Definition Lists
If you want to list definitions or the like, we recommend to use the enhanced `description` environment – it will automatically rendered in line with the preferred layout.

Type 1	That addresses central themes pertainng to migration, health, and disease. In Section 1, Wilson discusses the role of human migration in infectious disease distributions and patterns.
Type 2	That addresses central themes pertainng to migration, health, and disease. In Section 1.1 on page 2, Wilson discusses the role of human migration in infectious disease distributions and patterns.

3 Mathematical Formulas

Use the standard `equation` and `equation*` environments to typeset your equations in numbered form, e.g.,

$$a \times b = c, \tag{1}$$

or in nonnumbered form

$$a \times b = c,$$

using proper punctuation at the end of each equation in either case. For multiline equations we recommend to use the `align`, environment from the `amsmath` package

$$\begin{aligned} \left|\nabla U^{\mu}_{\alpha}(y)\right| &\le \frac{1}{d-\alpha}\int \left|\nabla \frac{1}{|\xi - y|^{d-\alpha}}\right| d\mu(\xi) = \int \frac{1}{|\xi - y|^{d-\alpha+1}}\, d\mu(\xi) \\ &= (d-\alpha+1)\int\limits_{d(y)}^{\infty} \frac{\mu(B(y,r))}{r^{d-\alpha+2}}\, dr \le (d-\alpha+1)\int\limits_{d(y)}^{\infty} \frac{r^{d-\alpha}}{r^{d-\alpha+2}}\, dr. \end{aligned} \tag{2}$$

You can also typeset this equation with the `multline` environment

$$\left|\nabla U_\alpha^\mu(y)\right| \le \frac{1}{d-\alpha}\int\left|\nabla\frac{1}{|\xi-y|^{d-\alpha}}\right|\,d\mu(\xi) = \int\frac{1}{|\xi-y|^{d-\alpha+1}}\,d\mu(\xi) \\ = (d-\alpha+1)\int\limits_{d(y)}^{\infty}\frac{\mu(B(y,r))}{r^{d-\alpha+2}}\,dr \le (d-\alpha+1)\int\limits_{d(y)}^{\infty}\frac{r^{d-\alpha}}{r^{d-\alpha+2}}\,dr, \quad (3)$$

or using `split` inside an `equation` environment

$$\begin{split}\left|\nabla U_\alpha^\mu(y)\right| &\le \frac{1}{d-\alpha}\int\left|\nabla\frac{1}{|\xi-y|^{d-\alpha}}\right|\,d\mu(\xi) = \int\frac{1}{|\xi-y|^{d-\alpha+1}}\,d\mu(\xi) \\ &= (d-\alpha+1)\int\limits_{d(y)}^{\infty}\frac{\mu(B(y,r))}{r^{d-\alpha+2}}\,dr \le (d-\alpha+1)\int\limits_{d(y)}^{\infty}\frac{r^{d-\alpha}}{r^{d-\alpha+2}}\,dr.\end{split} \quad (4)$$

For consistent notation throughout the book, we suggest that you use `\vec{}` for vectors in physcial space and `\tens{}` for tensors, i.e.,

$$\nabla\cdot\mathbf{v} = q, \qquad \mathbf{v} = \begin{cases} -\mathsf{K}\nabla p, & \text{for single-phase flow,} \\ -\mathsf{K}\lambda(S)\nabla p, & \text{for multiphase flow.}\end{cases} \quad (5)$$

For discrete vectors and matrices, one should rather use `\vect{}` and `\mat{}`,

$$\mathtt{div}(\boldsymbol{v}) = \boldsymbol{q}, \qquad \boldsymbol{v} = -\boldsymbol{K}\mathtt{grad}(\boldsymbol{p}). \quad (6)$$

Here, we have also used the discrete operators `\ddiv` and `\dgrad` defined in the `mrstbook` class.

4 Figures and Tables

Tables should, to the extent possible, be constructed without vertical lines and with as few horizontal lines as possible inside the tabular. The tabular should preferably extend across the whole line width. You can ensure this by either using paragraph columns with a specified width, or by altering the column separation. Likewise, if need be, you can set the space between rows by redefining:

```
\setlength{\tabcolsep}{5pt}       % default value is 6pt
\renewcommand{\arraystretch}{1.5} % default value is 1.0
```

An alternative way to adjust the rule spacing is to add `\noalign{\smallskip}` before or after the `\hline` and `\cline{i-j}` commands, as done in Table 1.

Use `\includegraphics` from the `graphicx` package to include figures. To simplify the processing of the full book and make the result as good as possible:

1. Place all your graphics files in a separate directory called `figs/` (or, if necessary, in subdirectories thereof).
2. Restrict your use of graphics formats to png, pdf, and encapsulated postscript.
3. For plots exported from MATLAB/MRST:
    - figures that primary contain lines, text, bar plots, pie charts, etc., should be exported to a vector graphics format like PDF or encapsulated PostScript.
    - plots containing 2D or 3D graphics constructed with MRST commands like `plotGrid` and `plotCellData` are best exported to PNG.
    - the `export_fig` package from MATLAB File Exchange is a useful tool that simplifies exporting plots.
4. Each figure should ideally include a single graphics file using a consistent and easy identifiable naming scheme like `ausamp-fig-01.pdf`, `ausampFig01.png`, and so on. In particular, if you make complicated figures that combine multiple graphics files and/or use complex Tikz/PGF/pgfplot commands (as in Figure 1), we request that you precompile and include these figures as separate files. The LaTeX `standalone` environment is very useful to this end.
5. Try to crop away unnecessary whitespace. Examples of useful tools to this end include image manipulation programs like `gimp` or command-line tools like `mogrify -trim` and `pdfcrop` on linux.
6. Try to avoid enforcing strict placement of figures. What you produce is *not* a camera-ready document and using constructions like `\begin{figure}[h]` (and even stricter versions) leaves LaTeX little room to optimize pagebreaks and float placements, and makes the typesetters' job more difficult.

By default, the figure caption should be placed under the figure. However, if the figure only occupies a fraction of the text width, you can consider using the `sidecaption`, as shown in Figure 1, to place the caption on the left side of the pager. Beware that the result may not look very visually appealing and that this declaration only works if the width of the included graphics is less than 7.8 cm. The caption text will be set raggedright if the width of the caption is less than 3.4 cm.

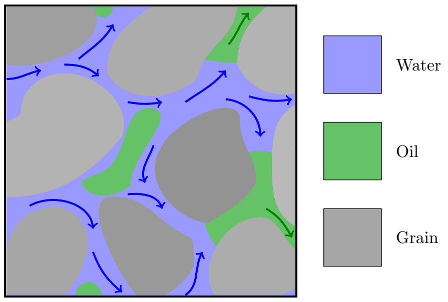
Two figures can also be arranged side-by-side inside a single `figure` environment by using the commands `\leftfigure` and `\rightfigure`, which place the corresponding plots inside two boxes that each covers approximately 46% of the current text width, and flash the two to the left and right, respectively, as shown in Figures 2 and 3. The code for this is as follows:

Table 1 Please write your table caption here

Classes	Subclass	Length	Action Mechanism
Translation	mRNA^a	22 (19–25)	Translation repression, mRNA cleavage
Translation	mRNA cleavage	21	mRNA cleavage
Translation	mRNA	21–22	mRNA cleavage
Translation	mRNA	24–26	Histone and DNA Modification

^{a} Table footnote (with superscript)

Fig. 1 If the width of the figure is less than 7.8 cm, use the `sidecaption` command to flush the caption on the left side of the page. If the figure is positioned at the top of the page, align the sidecaption with the top of the figure. To achieve this, you simply need to use the optional argument `[t]` with the `sidecaption` command.



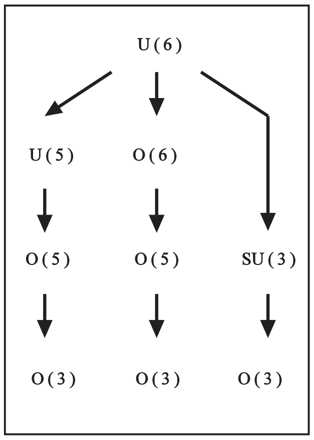


Fig. 2 This is the caption of the figure shown to the left.

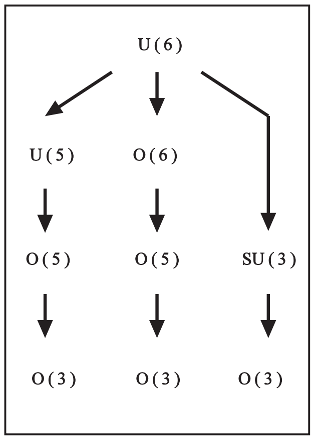


Fig. 3 This is the caption of the figure shown to the right.

```
\begin{figure}
  \begin{minipage}{\textwidth}
    % Options are: c=center, l=flush left, r=flush right, s=spread
    \leftfigure[l]{\includegraphics[width=.35\textwidth]{..}}\hfill%
    \rightfigure[r]{\includegraphics[width=.35\textwidth]{.}}
  \end{minipage}
  \twocaptionwidth{.35\textwidth}{.35\textwidth}
  \leftcaption{This is the caption..} \label{fig:left}
  \rightcaption{This is the caption ..}\label{fig:right}
\end{figure}
```

Notice that the captions are specified *outside* of the `minipage` environment, as suggested by Springer's Reference Guide. However, it is also possible to place the captions *inside* the minipage, stack multiple rows of figures, and use different dimensions for the minipage environments. As with the use of `\sidecaption`, these commands should nevertheless be used with some caution to ensure nice-looking typography. In a similar way, one can use the `\subfigures` command to subnumber multiple captions alphabetically within a single figure-environment, as shown in Figure 4a and 4b. Another alternative is to use a single caption that includes the description of each of the two subfigures like in Figure 5. Which of these two options one should use to create subfigures is left to each author to decide.

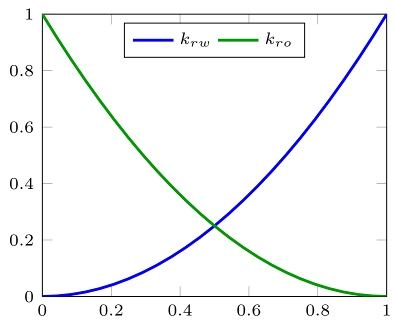


Fig. 4a Caption for the first subfigure.

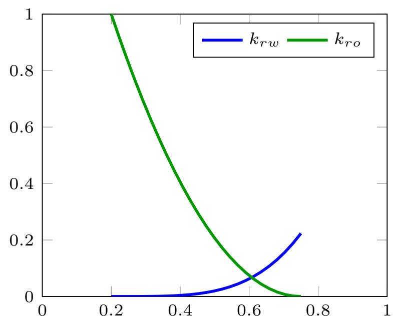


Fig. 4b Caption for the second subfigure.

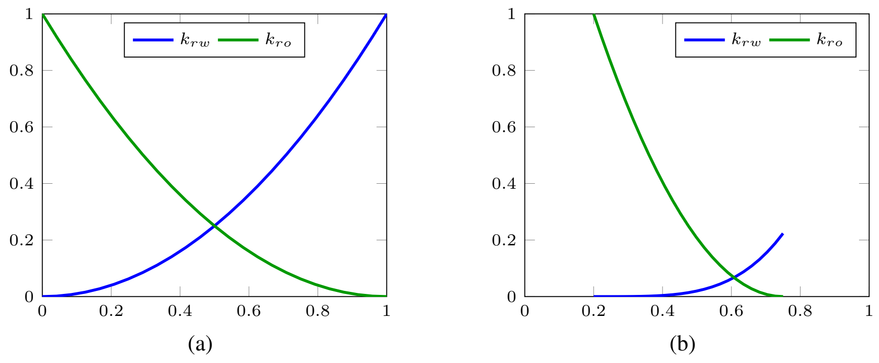


Fig. 5 Figure caption describing two subfigures. (a) The left subfigure. (b) The right subfigure.

If you copy figures, tables, or text passages from other works, you must obtain *permission* from the copyright holder (usually the original publisher). Please enclose the signed permission with the manuscript. The sources must be acknowledged either in the captions, as footnotes or in a separate section of the book.

5 Program Codes

The main purpose is to teach the reader new models and numerical methods that go beyond those discussed in the original MRST book [2]. To this end, it is particularly important that you, as a author, also focus on presenting your implementation in way so that the reader not only can use your code to solve the problems you have designed it for, but also can modify and extend your code to fit his/her own purposes.

Notice that when you submit a chapter for review and possible inclusion in this book, you must also provide a full, functional code for the whole module and all examples discussed in the text. This implies, in particular, that your code can only be based on functionality that is already available in a public version of MRST or functionality described by another chapter of this book. In the latter case, you also need to provide this code as part of your submission. During the peer review, our reviewers should be able to recreate the results of all examples discussed in your chapter. They will also perform a review of how the module is organized and how it is documented. They will, however, not be asked to perform an extensive code review.

5.1 Creating a Module

As a general rule, each chapter of this book should describe a specific add-on module to MRST, or in other words, the code you present should be organized as an MRST module. A module in MRST is strictly speaking a collection of functions, object declarations, and example scripts located in a folder. Each module needs to have a unique name and reside in a different folder than other modules. Our only requirements are that the code is well tested and documented in a format that does not deviate too far from that used elsewhere in MRST, uses a clear naming convention that avoids potential clashes with other parts of MRST, and contains a few tutorial examples that outline the main functionality and explain the most common syntax. The code also needs to contain a clear specification of copyright and the license under which it can be used (the GNU General Public License). In addition, we also require that your code does not use functionality from MATLAB's many toolboxes, which potential users may not have access to.

Although you are free to organize the code as you like, best practices suggest that you organize the module using a few standard subfolders

`examples`: Should contain the worked examples that outline the key functionality of the module, illustrate how you wish the code should be used, and present the strengths, but also potential weaknesses, of your code. If your examples require data, these must also be included, preferably in a separate `data` folder.

`models`: If your code is based on the AD-OO framework from MRST, your model classes should be found here.

`utils`: Additional functionality you expect that the user will need to call. Any *internal* functionality you *do not* expect that any user will call directly, should preferably be placed in subfolders of `utils` that are clearly marked as such, e.g., called `internal` or the special `private` folder, which lets MALAB limit the scope of these functions.

Other standard folder names include `data`, `src`, `fluids`, and `properties`. Each folder should preferably also contain a file `Contents.m` that outlines its content. Our reviewers will be asked to evaluate to what extent potential contributions satisfy these guidelines.

To the extent possible, your code should follow the coding standards that have been weakly imposed on the rest of MRST:

- Use `camelCase` to name functions, scripts, and variables. Use `CamelCase` to name objects.
- Ensure that your functions have unique names so that they do not unintentionally overload other functionality from MATLAB or MRST. If you *insist* on overloading function names, be warned that this will only work properly if the user loads your module *after* the other module that defines the same name.
- Use standard names for common quantities like `G` for the grid, `rock` for structs containing petrophysical data, `hT` or `T` for transmissibility, `bc` for boundary conditions, `state` for reservoir states, `W` for well structures, and so on.
- Ensure that your code has a certain degree of backward compatibility. In particular, try to avoid using very recent functionality so that your module only works with the most recent releases of MATLAB (and MRST).
- Document all functions (and objects) you expect that the user will call from his/her own scripts. For functions, describe the input and output data and what the function does, and provide references to related functions that offer similar functionality.

5.2 Tutorial Examples

Good tutorials are essential if you want other users of MRST to pick up and use your code. The intention is that this book should follow the important scientific principle of complete reproducibility, which in our setting means that the reader of the book should be able to use your code to reproduce the essential content of *all* examples presented in each chapter[1]. We thus require that all numerical examples are supported by complete source code written using code sections or live scripts:

[1] In saying this, we understand that discrepancies may appear as a result of using different MATLAB or MRST releases, in particular when it comes to plotting, and that a certain manual postprocessing is inevitable to create publication-quality figures

- Most modules in MRST present their tutorials in the form of MATLAB scripts with code sections. Code sections, also known as code cells or cell mode, contain contiguous lines of code you want the reader/user to evaluate as a group in a MATLAB script, beginning with two comment characters (`%%`). Typically, a code cell will perform a specific computation, make a plot, etc., and have a header and a comment above the code that explains in sufficient detail what the cell does and/or discusses its result.
- MATLAB live scripts and live functions (*.mlx) are interactive documents that combine MATLAB code with formatted text, equations, and images in a single environment called the Live Editor. In addition, live scripts store and display output alongside the code that creates it. This functionality is relatively new in MATLAB (live scripts in versions R2016a and above, and live functions in versions R2018a and above), and has so far not been used in any public MRST modules. We nevertheless welcome its use herein, as long as it helps the reader understand your tutorials. Notice, however that using live scripts/functions will limit backward compatibility, and it may therefore be advantageous to also include standard MATLAB scripts using code sections.

In designing your tutorials, you should consider the following:

1. Make sure that each tutorial loads all the necessary modules from MRST, including your current module, and that it does not specify paths that are specific to your own computer.
2. Section the example into natural entities and make sure that you present the reader with plots of quantities he/she may want to see. Typical examples: if you load or create a simulation model, the user will typically want to see the grid, petrophysical parameter, well positions, fluid properties, etc.
3. If the tutorial requires extensive setup or preprocessing that is not essential to the main narrative, you should factor these parts out to a separate function/script. If the setup is very time consuming, you should consider providing a `*.mat` file with the results, so that the user can skip this part when studying your example, but also have the ability to go back and redo the computations leading up to the contents of the `*.mat` file if necessary.
4. Good tutorials should run within a few minutes at most. If this is not possible, you should warn the user and provide some kind of feedback that measures progress, e.g., by reporting iterations/time steps, plotting partial results, or displaying a MATLAB `waitbar`.
5. If your tutorial relies on one or more data sets, make sure that these are available and come with a license that allows the reader to access and use them.
6. If your tutorial relies on stochastic data and exhibits significant variations with these, you should enable the user to specify the random seed (`rng(seed)`) you used to obtain the results presented in the book.
7. In many cases it is tempting to make parameter-dependent tutorials, in which the user needs to specify one or more input parameters to choose among different behavior and/or setups. Warning the user about this in at the top of the tutorial is generally a good idea, but is unfortunately not always sufficient, since users

of MRST often attempt to run tutorials before reading the details. As a general rule, you should therefore either supply reasonable defaults choices and/or make sure that the code checks that the input parameters have been set, e.g., using the `exist` function.

5.3 Presenting Code Excerpts

The chapters of this book are expected to follow a similar type of presentation as in the original MRST book [2]; that is, the presentation should intermingle the discussion of methods and examples with excerpts from MATLAB and MRST codes. For this purpose, you should use functionality from the `listings` package, which has been set up in `mrstbook.sty` to provide semantic syntax highlighting as in the MATLAB editor:

1. For inline code excerpts, you can use the `\mcode` command, which will give you a different font and color highlighting that reflects the MATLAB syntax: `twophaseJacobian(G,state,rock,fluid,'pn1',pv1,..)`. Variables and names of functions and scripts can also be distinguished from the rest of the text by using the `\verb` and `\texttt` commands, but these do not give semantic highlighting. In some rare cases, it may be necessary to use the underlying `\lstinline| |` command, primarily if the code you wish to display contains comment characters: `% this is an inline comment`.
2. Displayed code excerpts can be included with the `lstlisting` environment. As an example, the following commands

```
\begin{lstlisting}
% Radially symmetric grid graded towards the origin
P = [];
for r = exp(-3.5:.25:0),
   [x,y,z] = cylinder(r,16); P = [P [x(1,:); y(1,:)]];
end
P = unique([P'; 0 0],'rows');
\end{lstlisting}
```

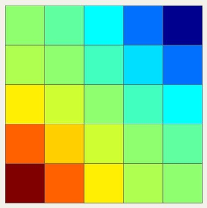
   will produce the output

```
% Radially symmetric grid graded towards the origin
P = [];
for r = exp(-3.5:.25:0),
   [x,y,z] = cylinder(r,16); P = [P [x(1,:); y(1,:)]];
end
P = unique([P'; 0 0],'rows');
```

3. In some cases, it may be useful to include a figure inside the code listing

```
p = initVariablesADI(zeros(nc,1));
q = zeros(nc, 1);              % source term
q(1) = 1; q(nc) = -1;          % -> quarter five-spot

eq    = div(grad(p))+q;        % equation
eq(1) = eq(1) + p(1);          % make solution unique
p     = -eq.jac{1}\eq.val;     % solve equation
```



for which we use Tikz in combination with a listing within a minipage nested within a node

```
\begin{tikzpicture}
  \node[anchor=south west] at (0,0){%
    \begin{minipage}{.99\linewidth}
      \begin{lstlisting}
p = initVariablesADI(zeros(nc,1));
:
      \end{lstlisting}%
    \end{minipage}
  };
  \node[anchor=south east] at (11.2,.4) {%
    \includegraphics[height=7.5em]{figs/solvePoisson-p}};
\end{tikzpicture}
```

You may have to adjust the position of the node containing the picture and possibly also explicitly enforce appropriate line breaks before and after this `tikzpicture` environment.

4. Last, but not least, it may be necessary to include output from MATLAB. For this, we propose that you use the `Verbatim` environment, which is what we have used for all text boxes herein. This can easily be configured in many different ways. We recommend, in particular, that you use leftline and a different text color distinguish the computer output from other verbatim displays

```
fluid =
    properties: @(varargin)properties(opt,varargin{:})
    saturation: @(x,varargin)x.s
       relperm: @(s,varargin)relperm(s,opt,varargin{:})
```

5. If you need to display input data, we suggest that you use the `Verbatim` environment with a frame

```
SWOF
--   SWAT            KRW                   KRO              PCOW
   0.1200000000                    0    1.000000000000    0
   0.1210000000   0.000000011363636    1.000000000000    0
   0.1400000000   0.000000227272727    0.997000000000    0
   0.1700000000   0.000000568181818    0.980000000000    0
   0.2400000000   0.000001363636364    0.700000000000    0
   0.3199999990   0.000002272727261    0.350000004375    0
   0.3700000000   0.000002840909091    0.200000000000    0
   0.4200000000   0.000003409090909    0.090000000000    0
```

```
  0.5200000000   0.000004545454545   0.021000000000    0
  0.5700000000   0.000005113636364   0.010000000000    0
  0.6200000000   0.000005681818182   0.001000000000    0
  0.7200000000   0.000006818181818   0.000100000000    0
  0.8200000000   0.000007954545455   0.000000000000    0
  1.0000000000   0.000010000000000                0    0
/
```

5.4 Standard Index and Index with MRST Commands

The book will have two 2-column index lists at the end: one for general keyword and one for MRST (and MATLAB) commands. Entries into the first index are included through the usual `\index{My new keyword}` command. In addition, we provides a new command `\moduleindex{modulename}` that will collect references to MRST modules as subentries of a keyword called "Modules". Entries into the second index are entered through the `\mrstindex{myMrstCommandName}` command. To generate the two index lists, you must run two passes of `makeindex`

```
   makeindex -s styles/mrstsvind.ist myfilename
   makeindex -s styles/mrstsvind.ist myfilename.mii -o myfilename.mio
```

Acknowledgements If you want to include acknowledgments of assistance and the like at the end of an individual chapter please use the `acknowledgement` environment – it will automatically be rendered in line with the preferred layout.

Appendix

When placed at the end of a chapter or contribution (as opposed to at the end of the book), the numbering of tables, figures, and equations in the appendix section continues on from that in the main text. Hence please *do not* use the `appendix` command when writing an appendix at the end of your chapter or contribution. If there is only one the appendix is designated “Appendix”, or “Appendix 1”, or “Appendix 2”, etc. if there is more than one.

$$a \times b = c \tag{7}$$

References

References may be *cited* in the text using the standard `\cite` command, or any of the extension from the `natbib` package: `\citet` gives a citation with author names included (Aarnes et al. [1]), whereas `\citep` gives only a number like `\cite` ([1]). All authors are expected to provide references as a BibTeX file, and thus you do not need to think of formatting or ordering of the references. This will be done by the editors. However, please make sure that you include as much relevant information as possible and follow the following rules:

- Separate all authors by "and"; this will automatically be changed to appropriate forms
- If you use initials only in names, please write "Smith, W. G." and not "Smith, W.G." or "W.G. Smith". Both the latter will be interpreted as W. Smith.
- Put brackets around names and other words in titles that need to be always capitalized, like {*K*}*rylov subspace method*. Do not put brackets around the full title.
- Try to include DOI numbers whenever possible, but do not include this as a full url. Do not include other urls to the full online document unless there is no DOI available.
- ISBN and ISSN numbers are not necessary.
- If possible, try to abbreviate journal names using standard abbreviations.
- **Most important:** to avoid potential name clashes with BibTeX files from other chapters, please make sure that you use unique names for your references, e.g., by adding the name of your module to the citation key like `msrsb:mycitationkey`

The list below shows four examples of citations as they will appear in a chapter, following the conventions from the `abbrvnat` bibstyle:

[1] J. E. Aarnes, T. Gimse, and K.-A. Lie. An introduction to the numerics of flow in porous media using Matlab. In G. Hasle, K.-A. Lie, and E. Quak, editors, *Geometrical Modeling, Numerical Simulation and Optimisation: Industrial Mathematics at SINTEF*, pages 265–306. Springer–Verlag, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-68783-2_9.

[2] K.-A. Lie. *An introduction to reservoir simulation using MATLAB/GNU Octave: User guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge University Press, Cambridge, 2019.

[3] O. Møyner and K.-A. Lie. A multiscale restriction-smoothed basis method for high contrast porous media represented on unstructured grids. *J. Comput. Phys.*, 304:46–71, 2016. doi:10.1016/j.jcp.2015.10.010.

[4] O. Møyner and H. A. Tchelepi. A multiscale restriction-smoothed basis method for compositional models. In *SPE Reservoir Simulation Conference*, 2017. doi:10.2118/182679-MS.

Index

Usage of MRST Functions