Akustische Simulationen mit Matlab und Python im Org-Mode

Johann-Markus Batke

Hochschule Emden/Leer, Constantiaplatz 4, 26723 Emden, Deutschland

Email: johann-markus.batke@hs-emden-leer.de

Zusammenfassung

Der Org-Mode des Texteditors Emacs stellt ein mächtiges Werkzeug dar, dass die Umsetzung des Konzepts „Literate Programming“ ermöglicht. Da der Org-Mode mittlerweile sehr populär geworden ist, existiert auch eine Portierung für den ebenfalls verbreiteten Editor Vi.

Literate Programming bedeutet technisch betrachtet das gemeinsame Bearbeiten von Programmtexten und dokumentenorientierten Texten in einer Datei. Werden gleiche Konzepte auch bei der Python-Software-Suite Jupyter verfolgt, ist der Org-Mode ungleich flexibler, da zum Beispiel Testvariablen zentral verwaltet werden können und nur wenige Einschränkungen für die verwendeten Programmiersprachen bestehen. So lassen sich Auswertungen z.B. in Matlab/Octave, Python, R, C++ und anderen Sprachen miteinander in einem Dokument verbinden, Daten können unter Programmtextfragmenten ausgetauscht werden.

Die Darstellung von Programmier- und Dokumentationsaufgaben wird in diesem Beitrag am Beispiel einer Toolbox für akustische Berechnungen dargestellt, die vollständig im Org-Mode implementiert ist. Verwendet werden für die Programmierung Matlab/Octave und Python, für die Dkumentation deie exportierbaren Formate HTML und LaTeX. Bespondere Beachtung kommt in der Darstellung dabei der Aufgabe des Testens der Implementierungen zu, die aus wissenschaftlicher Hinsicht sehr relevant ist.

Literate Programming

„Literate Programming“ bedeutet wörtlich übersetzt „literarisches Programmieren“ - Donald Knuth als Urheber dieses Paradigmas bezeichnet „Literate Programming“ auch als „strukturierte Dokumentation“ („structured documentation“) [1]. Technisch betrachtet bedeutet es das gemeinsame Bearbeiten von Programmtexten und dokumentenorientierten Texten in einer Datei [2].

Als Implementierung des Literate Programming stellt Knuth in [3] sein System `WEB` vor. Eine `WEB`-Datei enthält Dokumentation und Quelltext. Der Quelltext kann mit dem Befehl `tangle` in eine weitere Datei extrahiert werden, die Dokumentation mit dem Befehl `weave` in TeX-Code verwandelt werden.

Ein Variation dieser Idee präsentiert Thimbleby für die Programmiersprache C in Form der Anwendung `CWEB`, die Formatierung der Ausgabe wird über `troff` durchgeführt [4]. Als wesentliche Anwendungen werden hier bereits genannt

- (literarische) Kommentare auf jede Form der Literatur
- die mehrsprachige Kommentierung von Computer-Programmen
- die Kombination von formaler Spezifikation einer Software und Quelltext
- die informelle Beschreibung von Programmtexten
- die Annotation von interaktiven Verwendungen eines Programms.

Der Org-Mode des Texteditors Emacs erlaubt die komfortable Umsetzung des Paradigmas „Literate Programming“ und kann Weiterführungen der Ideen aus WEB und CWEB betrachtet werden. Es können im Prinzip beliebige Programmiersprachen verwaltet werden, die per `tangle` in eigene Dateien geschrieben werden können. Zudem steht eine Vielzahl von Export-Formaten (anstelle von `weave`) zur Verfügung, über die eine Programm-Dokumentation erstellt werden kann. In diesem Beitrag soll anhand der Akustik-Toolbox Booast gezeigt werden, wie die Programmiersprachen Matlab/Octave und Python gehandhabt werden können [5].

Software

Emacs

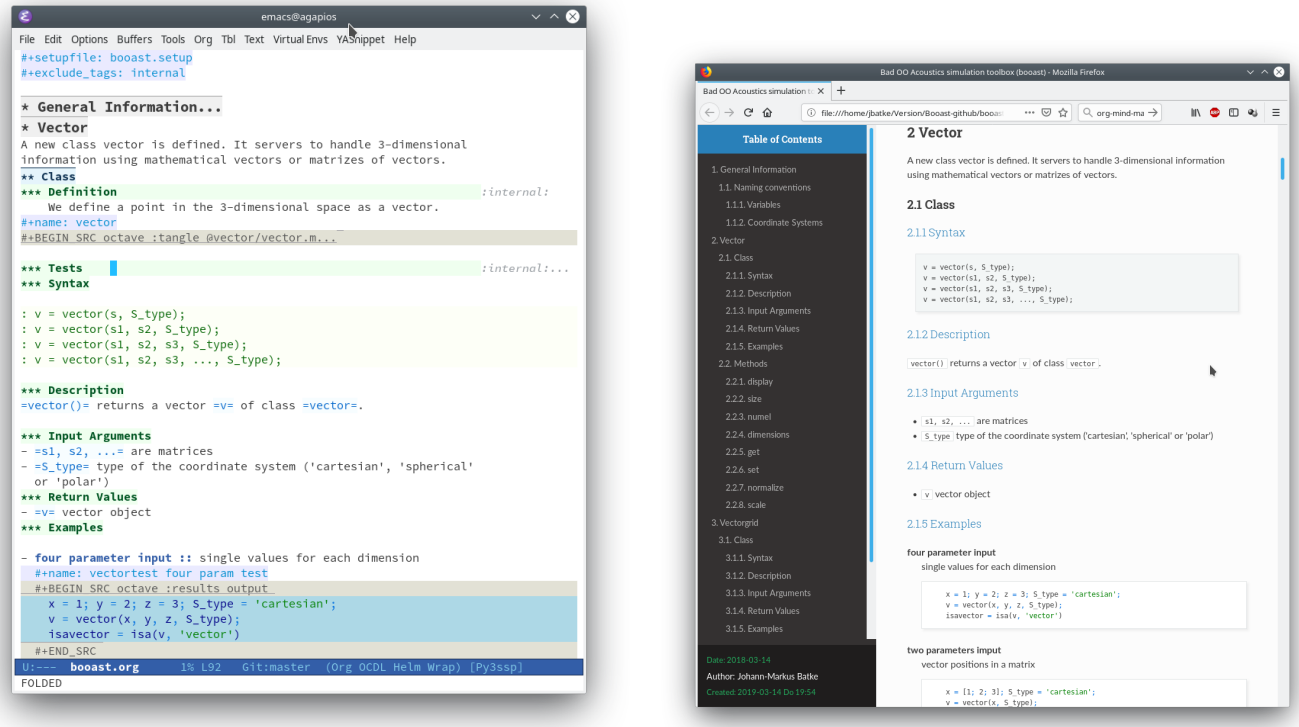
Die Ursprünge des Editors Emacs gehen auf das 1976 zurück. Das Akronym Emacs steht für „Editing MACroS“, eine Sammlung von Macros zur Bedienung des Texteditors TECO. Der Editor TECO war in der gleichnamigen Sprache TECO programmierbar, und die „Emacs“ wurden in eben dieser Sprache geschrieben [6]. Zur gleichen Zeit entstand ein ähnlicher Editor mit Namen „EINE“ (EINE is not EMACS) auf Basis der Programmiersprache Lisp (und konsequenterweise später die Weiterführung „ZWEI“ - ZWEI was EINE initially). Der heute noch gebräuchlihe GNU Emacs wurde von Richard Stallmann initial im Jahr 1985 in C implementiert, funktionale Erweiterungen werden seitdem wie in EINE und ZWEI per Emacs-Lisp ermöglicht. Solche Erweiterungen können meist einfach über das Paket-System des Emacs nachgeladen werden, sofern sie nicht ohnehin in der aufgrund ihrer langen Entwicklungsgeschichte reichhaltigen GNU-Emacs-Distribution enthalten sind [7].

Org-Mode

Eine wichtige solche Erweiterung des Emacs ist der 2003 entstandene Org-Mode. Der Org-Mode ist heute Teil der GNU-Emacs-Distribution und wird fortlaufend aktualisiert und erweitert [8].

Er wurde vom Informatiker Carsten Dominik mit dem Ziel programmiert, wissenschaftliche Ausarbeitungen besser organisieren zu können [9].

Der Org-Mode ist mittels Info-System des Emacs gut dokumentiert. Die aktuelle Version 9 bietet ausschließlich die elektronische Darstellung der Dokumentation, eine gedruckte Fassung liegt für Version 7 vor [10].



(a) Der Texteditor Emacs mit der Quelldatei booast.org. Der Export dieser Datei kann u.a. in das Format html erfolgen.

(b) Die HTML-Dokumentation der Toolbox booast wird durch das Stylefile-Paket Read-the-org im Erscheinungsbild verändert.

Abbildung 1: Die Programmfenster der Programme Emacs und Firefox.

Der Org-Mode ist ein Modus zur Bearbeitung von ASCII-Texten, ein minimalistisches Markup (im Konzept vergleichbar mit Markdown) ermöglicht eine strukturierte Dokumenteneingabe. Weiterhin können LaTeX-Fragemente eingebettet werden, was die Eingabe wissenschaftlicher Texte erleichtert. Abbildung 1a zeigt ein beispielhaftes Dokument in Org-Formatierung.

Exportfunktionen in verschiedenste Formate ermöglichen die Konvertierung des Textes in ein Anzeigeformat. Im Rahmen dieses Beitrags wird der Export in HTML verwendet. Die Formatierung der Dokumenteneingabe in Abbildung 1 zeigt ein Beispiel. Der Export des Dokuments hat so die Funktion wie der Schritt `weave` aus dem ursprünglichen Konzepts des Literate Programming.

Das Markup des Org-Mode geht weit über die Textformatierung hinaus. In diesem Kontext besonders interessant ist die Einbettung sog. Code-Blöcke [11]. Eine Octave-Fragment innerhalb einer org-Datei wird etwa wie folgt darstellt:

```
#+begin_src octave :tangle doppel.m
function y = doppel(x)
y = 2*x;
#+end_src
```

Über die Angabe `:tangle doppel.m` wird ganz im Sinne des Literate Programming festgelegt, dass dieses Quelltextfragment in eine den Matlab/Octave-Konventionen entsprechende Datei geschreiben wird.

Da nahezu beliebige Programmiersprachen dargestellt werden können, wird die Quelltextverwaltung im Org-Mode mit *Babel* betitelt. Durch Kombination der Export-Funktionen und Babel ist die Umsetzung des Paradigmas Literate Programming in komfortabler Weise möglich.

Vi

Für Nutzer des Texteditors vim existiert eine Portierung des Org-Mode [12]. Alternativ kann der GNU Emacs in seiner Bedienung den Bedienschemata des vim vollständig angepasst werden (extensible vi layer, evil) [13]. Insbesondere die Distribution spacemacs nutzt diese Möglichkeit [14].

Matlab und Octave

Matlab kann als Standard-Sprache für viele Simulationsaufgaben in der Industrie und Forschung betrachtet werden. Die Software wurde ursprünglich Ende der 1970er als freies Projekt gestartet, aber 1984 von der Firma Mathworks kommerzialisiert [15]. Mittlerweile existieren zahlreiche freie Alternativen, von denen besonders Octave als Code-kompatible Variante hervorzuheben ist, das seit 1992 in Vollzeit durch John Eaton entwickelt wird [16]. Beide Programme laufen unter allen üblichen Betriebssystemen (Windows, OSX, Linux).

Bei der Umsetzung von Objektorientierung in Form von Klassen werden in Matlab/Octave viele einzelne Dateien erzeugt. Die Verwaltung einer so entstandenen Klassenbibliothek erfordert einen gewissen Aufwand. Mittels Babel lassen sich wie im vorherigen Abschnitt illustriert solche Klassen komfortable in einer org-Datei verwalten und auch weitergeben, was ein praktischer Aspekt beim Austausch mit Dritten ist. Auch können Testprogramme mit eingebettet werden, ebenso wie für den Test erforderliche Testdaten. Damit sind wichtige Kriterien der reproduzierbaren Forschung technisch einfach

zu erfüllen.

Anwendungsbeispiel

Als Anwendungsbeispiel des Org-Mode im Zusammenspiel mit Octave dient hier der Test der Implementationen der sphärischen Besselfunktion, die eine typische spezielle Funktionen aus dem Bereich akustischer Berechnungen darstellt. Als Beispiel für die Testdatenerzeugung soll die Octave-Implementation mit einer Python-Variante verglichen werden.

Die Octave-Version der sphärischen Besselfunktion stammt aus der Toolbox booast [5]. Der Aufruf erfolgt in der Form

```
N_order = 3;
L_x = 100;
x_L = linspace(0, 10, L_x);
J_sph_L = sphericalbesselj(N_order, x_L);
```

Die Python-Version der sphärischen Besselfunktion aus der Bibliothek scipy sieht im Aufruf ähnlich aus [5], [17]:

```
import numpy as np
import scipy.special as sc
N_order = 3
L_x = 100
x_L = np.linspace(0, 10, L_x)
j_NL = sc.spherical_jn(N_order, x_L)
```

Beide Versionen sollten die gleichen Ergebnisse ergeben, was im Rahmen der Octave-Toolbox als Test vorgesehen ist. Der Vergleich beider Programmfragmente erfordert die Steuerung beider Implementierungen mit gleichen Parametern (hier wurde zur Vereinfachung ein fester Wertebereich für `x_L` von 0 bis 10 festgelegt). Der Org-Mode bietet die Möglichkeit, Eingangsparameter für Code-Blöcke zu verwalten. Die gemeinsamen Parameter werden in einer Org-Mode-Tabelle niedergelegt:

```
| Datei Octave Ergebnisse | octres.csv |
| Datei Python Ergebnisse | ipyres.csv |
| L_x                     |         50 |
| N_order                 |          3 |
```

Die Tabelle legt die Namen der csv-Dateien fest, die in eine Ergebnisse geschrieben werden. Weiterhin sind die Anzahl der zu berechnenden Punkte `L_x` und die Ordnung der Funktion `N_order` als Variable festgelegt.

Für die Berechnung in Octave kann nun folgendes Quelltextfragment angegeben werden:

```
#+header: :var S_filename = testparam[0,1]
#+header: :var N_order = testparam[3,1]
#+header: :var L_x = testparam[2,1]
#+begin_src octave
J_sph_L = sphericalbesselj(N_order, linspace(0, 10,
     L_x));
dlmwrite(S_filename, J_sph_L.');
#+end_src
```

Die übergebenen Parameter in der Zeile `#+header:` werden in eine entsprechende Octave-Darstellung übersetzt und in den Quelltext eingefügt.

Gleichermaßen kann in Python verfahren werden:

```
#+header: :var S_filename=testparam[1,1]
#+header: :var N_order=testparam[3,1]
#+header: :var L_x=testparam[2,1]
#+BEGIN_SRC ipython
j_NL = sc.spherical_jn(N_order, np.linspace(0, 10,
    L_x))
np.savetxt(S_filename, np.asarray(j_NL), delimiter=
    ",")
#+END_SRC
```

Das Ergebis der Berechnung zeigt Abbildung 2.

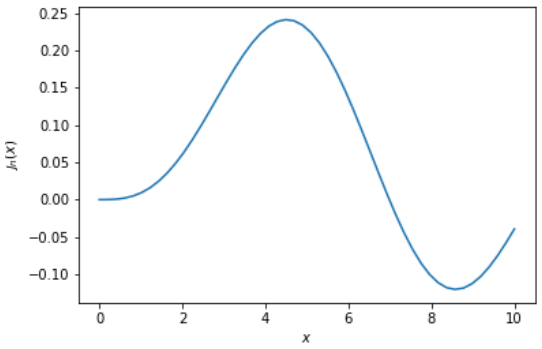


Abbildung 2: Die sphärische Besselfunktion, nach Parametern der gegebenen Tabelle berechnet durch Octave bzw. Python. Ziel ist es, Abweichung zwischen beiden Berechnungen zu ermitteln.

Beide Ergebnisse werden nun voneinander abgezogen, hier beispielsweise mithilfe eines Python-Code-Blocks:

```
#+header: :var S_octres = testparam[0,1]
#+header: :var S_ipyres = testparam[1,1]
#+BEGIN_SRC ipython
octdata = np.genfromtxt(S_octres, delimiter=",")
ipydata = np.genfromtxt(S_ipyres, delimiter=",")
plt.plot(octdata-ipydata)
plt.xlabel("$x$")
plt.ylabel("Differenz␣Octave/Python")
#+END_SRC
```

Die Differenz ist in Abbildung 3 dargestellt - in diesem Beispiel wird zufriedenstellende Übereinstimmung erzielt.

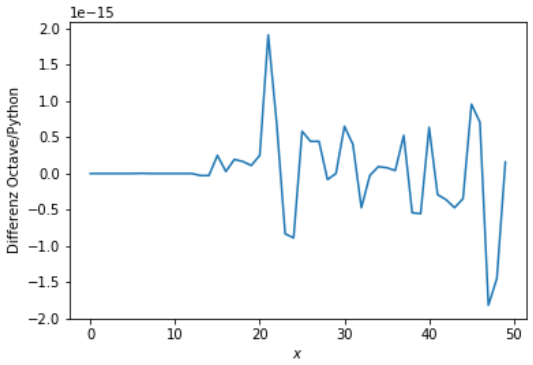


Abbildung 3: Differenz der Berechnungen in Octave und Python für die sphärische Besselfunktion.

Literatur

[1] D. E. Knuth, *The art of computer programming*. Reading, Mass: Addison-Wesley, 1997, ISBN: 0201896834.

[2] Wikipedia, Die freie Enzyklopädie, Hrsg. (2018). Literate Programming. Datum des Abrufs: 30. März 2019, 19:57 UTC, Adresse: https://de.wikipedia.org/w/index.php?title=Literate_programming&oldid=183704796.

[3] D. E. Knuth, „Literate Programming“, *The Computer Journal*, Jg. 27, Nr. 2, S. 97–111, Jan. 1984, ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97. eprint: http://oup.prod.sis.lan/comjnl/article-pdf/27/2/97/981657/270097.pdf. Adresse: https://dx.doi.org/10.1093/comjnl/27.2.97.

[4] H. Thimbleby, „Experiences of ‘Literate Programming’ using cweb (a variant of Knuth’s WEB)“, *The Computer Journal*, Jg. 29, Nr. 3, S. 201–211, Jan. 1986, ISSN: 0010-4620. DOI: 10.1093/comjnl/29.3.201. eprint: http://oup.prod.sis.lan/comjnl/article-pdf/29/3/201/1556530/290201.pdf. Adresse: https://dx.doi.org/10.1093/comjnl/29.3.201.

[5] J.-M. Batke. (2018). Bad Object Oriented Acoustics Simulation Toolbox, Adresse: https://github.com/badkey/booast.

[6] J. Zawinski. (8. März 1999). Emacs Timeline, Adresse: https://www.jwz.org/doc/emacs-timeline.html (besucht am 31. 3. 2019).

[7] Free Software Foundation, Hrsg. (2015). GNU Emacs, Adresse: https://www.gnu.org/software/emacs/ (besucht am 31. 3. 2019).

[8] C. Dominik u. a. (2003). Org mode for Emacs - You Life in Plain Text, Homepage, Adresse: https://orgmode.org/ (besucht am 31. 3. 2019).

[9] C. Dominik. (15. Juli 2008). Emacs Org-mode – a system for note-taking and project planning, Google-TechTalk, Adresse: https://www.youtube.com/watch?&v=oJTwQvgfgMM (besucht am 4. 2. 2012).

[10] ——, *The Org Mode 7 Reference Manual - Organize Your Life with GNU Emacs*. Network Theory Ltd., 2010, ISBN: 1906966087, 9781906966089.

[11] E. Schulte, D. Davison, T. Dye und C. Dominik, „A Multi-Language Computing Environment for Literate Programming and Reproducible Research“, *Jorunal of Statistical Software*, Jg. 46, Nr. 3, Jan. 2012.

[12] J. C. Ebersbach. (2019). vim-orgmode, Adresse: https://github.com/jceb/vim-orgmode (besucht am 31. 3. 2019).

[13] E. Kolev u. a. (2019). Evil, Adresse: https://github.com/emacs-evil/evil (besucht am 31. 3. 2019).

[14] S. Benner. (2019). spacemacs - a community driven distribution, Adresse: http://spacemacs.org (besucht am 31. 3. 2019).

[15] Wikipedia, Die freie Enzyklopädie, Hrsg. (18. Sep. 2018). Matlab. Version 181023482. Datum des Abrufs: 30. März 2019, 20:27 UTC, Adresse: https://de.wikipedia.org/w/index.php?title=Matlab&oldid=181023482.

[16] J. W. Eaton. (1992). GNU Octave, Homepage. Datum des Abrufs: 30. März 2019, 20:30 UTC, Adresse: https://www.gnu.org/software/octave/.

[17] SciPy community, Hrsg., *SciPy Reference Guide*, Version Release 1.2.1, 10. Feb. 2019. Adresse: https://docs.scipy.org/doc/scipy/scipy-ref-1.2.1.pdf.