Octave for Engineers

Andreas Stahel
Bern University of Applied Sciences

OctConf, June 2013, Milano

Personal I

- Andreas Stahel

- Working at Bern University of Applied Sciences in Biel
  - Teaching:
    - Math at Bachelor level to mechanical and electrical engineers
    - Numerical Methods for the Master Program of Biomedical Engineering
  - Many industry projects in mathematical modeling
  - Research: mainly numerical methods applied to PDEs, lately on Novikov Veselov equations
- Web: `//staff.ti.bfh.ch/sha1/`
- E-mail: `Andreas.Stahel@bfh.ch`

Personal II

- *Octave* is used regularly for teaching, project work and research.
- For the last couple of years I run a class on how to use *Octave* for engineers.
- I started using *Octave* in 1993/94 and am addicted to it since then.
- *Octave* replaces `MATLAB` for many reasons: open source, great community support, platform independent, (legally) free.
- My professional life would be different without *Octave*!

Thank you guys

Goals of this talk

- Point out important aspects to consider when moving engineers closer to *Octave*.
- Help others to teach *Octave*.
- Show examples and provide documentation to the community.
- All these observations are based on personal experience.

Goals of the class on *Octave*

- The engineer recognizes situations when *Octave* might be useful.
- He/she has the skill to translate the idea/algorithm to useful, reliable *Octave* code.
- The student shall continue using *Octave* for his projects in school and also when he leaves school for work in industry or research.

Whom am I talking to?

One of the most important point for classes is to determine the key skills and interest of the audience.

For my class:

- Most students are electrical or mechanical engineers in the second or third year of the Bachelor program. A few computer scientists might show up.
- All students had some basic training in a classical programming language (C, C++, Java, ...)
- All students have some basic knowledge of Physics, Math, Electronics and Mechanics.
- The main interest of the students is Engineering, not Math.

The context of the class

It is important to keep the circumstances of the class in mind.

- The class meets once a week for 90 minutes, for 16 weeks.
- The students are expected to work on the topic outside of the class too, for approximately 20-30 hours.
- The students choose to attend this class, i.e. not mandatory.
- We have regular lecture rooms and a lab with a PC for each student, or some space to put his laptop.
- The typical class size is between 12 and 20 students.

From the Starting Point to the Goal I

Once we know the starting point and the goal we have to choose a path, such that (most) students are able to go all the way.

- Probably the most important skill is to know where to find **documentation and help**. Google is helpful, but often inefficient!
- To be able to use *Octave* one has to have **basic skills and knowledge** on programming with *Octave*, e.g. vectors, matrices, data types, control structures.
- For future Engineers some commands are considerably more important that others, e.g. graphics, data analysis, ...
- Teachers can not force students to learn, they can at best guide and help! Engineers are interested in **engineering applications** and thus we have to use these to the students advantage.

From the Starting Point to the Goal II

Based on the above reflexions the class is organized in two parts:

- During the first 7-8 weeks basic *Octave* skills and commands are presented.
  - Learn about the basics of *Octave* programming.
  - See a selection of typical commands and codes useful for engineers.
  - Learn to find and use the available documentation.
- The remaining weeks are used to examine a selection of real world engineering problems.
- The students are required to works on a project of their own choice and turn in a written report on their project.

Evaluation of student performance

The grading of the class is based on three aspects:

30% Presence in class.

30% A mid term quiz, checking the basic *Octave* commands and structures.

40% Evaluation of the report on the project.

Setup for one Session I

It is important that the students have an active role working with *Octave*. Thus each 90 minute session on basic skills is split in two parts:

- For the first part the instructor presents some topic and points out important or surprising aspects.
  - I use blackboard and beamer to present facts and run live demos to illustrate the most important aspects.
  - More details are provided in the lecture notes and the students are informed by E-mail what aspect will be presented in class.
  - Programming techniques similar to C, C++ receive very little attention, the special tricks of the trade of *Octave* are pointed out.

Setup for one Session II

- For the second part the students obtain a worksheet (on paper) with instruction on how to get to know the commands and structures presented in the first part.
  - The documentation in the lecture notes are used to complete the tasks asked for on the worksheet.
  - The instructor shall not talk to the class any more, but provide individual assistance.
  - Usually the students are informed about complete solutions to the task at hand by a reference to my web site.

The above approach relies on the students to work on their own:

- Each student will work at his own pace.

\+ Good students can work at their own pace and learn a lot.

– Non motivated students learn close to nothing.

Topics for Basic Skills

Obviously we have to choose the *Octave* topics to be considered *Basic Skills*

- Setup of *Octave* on the students system, editor, documentation
- Vectors, matrices, scripts and functions
- Data types, control structures, formated reading and writing
- Solving equations, linear systems and nonlinear equations
- Graphics, including export to PNG and PDF
- Differential equations, use C++ code
- Elementary image processing, (vectorization)

Examples for *Basic Skills* I

In the first session the goal is to assure that all students can start up *Octave*, and install it on their personal system.

- Login, starting *Octave*, an editor and a browser pointing to the standard *Octave* documentation.
- Installing *Octave* and packages on Win*, Linux, Mac, ...
- Pointers to my web site with the lecture notes, sample codes and data files.
  `http://staff.ti.bfh.ch/sha1/Octave.html`
- Use the commands `help`, `doc`
- Generate a trivial graph.

Examples for *Basic Skills* II

One session is devoted to vectors and matrices, scripts and functions.

- Creating vectors and matrices
- Vector operations, including the dot notation, e.g. `.*` or `*` ?
- Illustrate the power of vectorized code.
- Explain the difference between a function file and a script file.

Examples for *Basic Skills* III

One session is devoted to graphics.

- Generate a simple 2D graph.
- Titles, labels, text in graph, multiple graphs.
- Generate a PDF or PNG file for inclusion in LaTeX, LibreOffice, Word, ...
- Histograms, 3D meshes and surfaces, contour plots
- Vector fields

Have a look at the corresponding worksheet!

Engineering Applications

- One of the principal goals of the class is to make students recognize possible applications of *Octave*, and then he/she should be able to use *Octave*.
- To move closer to this goal we examine a few real world examples, either from Bachelor or Master thesis projects or from industry projects.
- The students (usually) recognize that they might run into similar questions and problems soon, thus they want to learn about the solutions.

Setup for one Session I

It is important that the students have an active role working with *Octave*. Thus each 90 minute session on basic skills is split in two parts:

- For the first part the instructor presents the question/problem to be examined and then tries to explain idea of the algorithm to be used. Short code segments with the key points are shown and the final results are displayed.
  - I use again blackboard, beamer and live demos.
  - More details are provided in the lecture notes.
  - For each session I pick one particular tool to be used.

Setup for one Session II

- For the second part the students obtain a worksheet (on paper) with instruction on how to get to work through the complete solution of the problem presented in the first part.
  - The documentation in the lecture notes are used to complete the tasks asked for on the worksheet.
  - Most code and data is provided on the web page. The students are invited to play with the codes.
  - The instructor shall not talk to the class any more, but provide individual assistance.

The above approach relies on the students to work on their own:

- Each student will work at his own pace.

\+ Good students can work at their own pace and learn a lot.

– Non motivated students learn close to nothing.

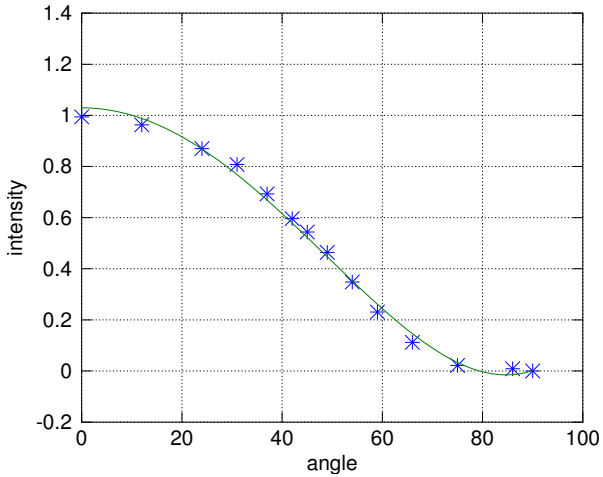
Selection of possible Applications

- Applications of linear regression, nonlinear regression
- Movement analysis of a watch caliber hitting the floor (generate a movie)
- Implementing an arctan(x) function on a micro controller (use `int16`, `uint16`, `int32`, `uint32`)
- Vibration analysis (Fourier)
- Analysis of damping of a vibrating cord (regression and calling an external program)
- Compute the magnetic field in a Helmholtz coil (numerical integration and vector fields)
- Combine two laser scans of an object to one 3D image
- Analyse the evolution of stock values (reading data from files)

Examples of Engineering Applications I

One double session is devoted to linear and nonlinear regression. Many years of consulting students show that there is an enormous lack of skill using this engineering tool.

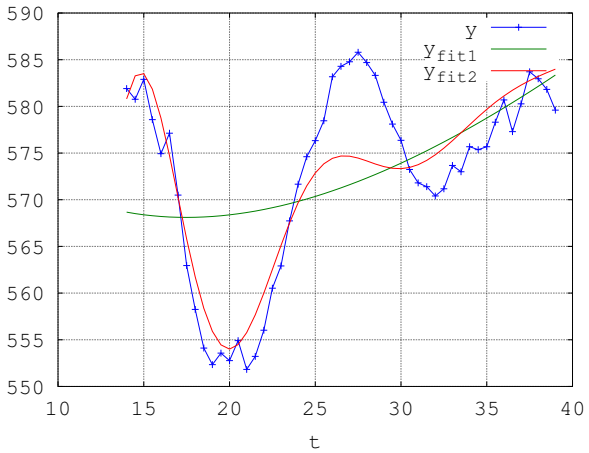
- The basics, building up the matrix notation.
- Choice of basis function, rescaling, high condition number.
- Use `LinearRegression()` from the `optim` package.

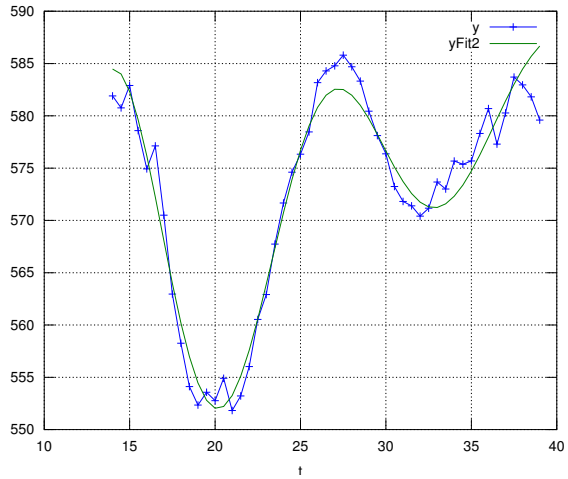


- Do **not** mention the more mathematical aspects, like QR factorization instead of $\mathbf{M}^T \cdot \mathbf{M}$.

Examples of Engineering Applications II

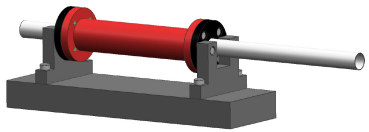
- Illustrate the problems of nonlinear regression, e.g. how to construct initial values. Use `leasqr()` from the `optim` package.

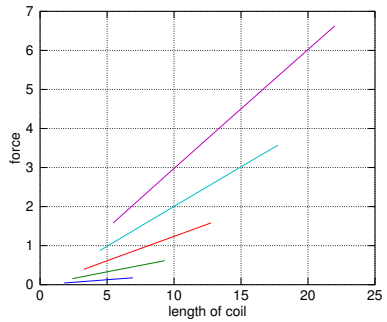


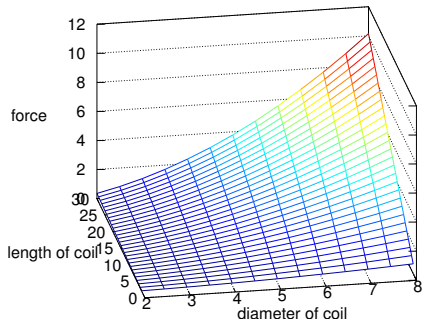


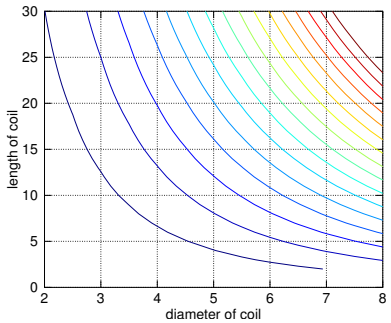
Examples of Engineering Applications III

- In the second part the students choose a topic from the lecture notes and examine the question and its solution.
  - One example is the force generate by magnetic coil. This is a liner regression with two independent variables.





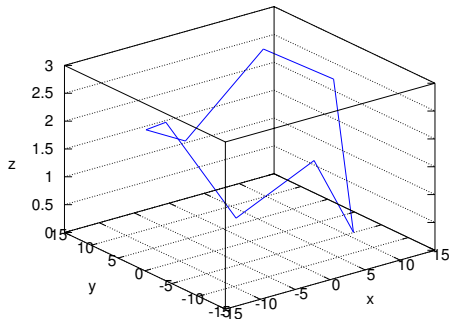


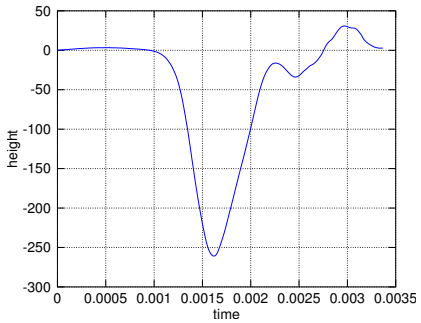


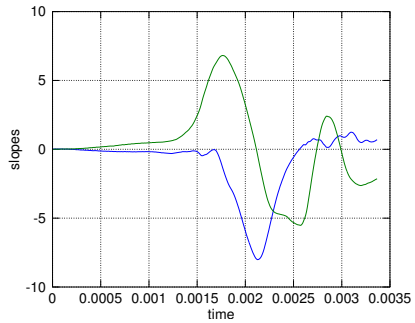
Examples of Engineering Applications IV

One session is devoted to the analysis of the movements of a watch caliber falling on the ground. Based on the measured data multiple animations are generated by an external program `mencoder`.

- Read the data from a file and visualize.
- Decompose the movement into height changes, rotations and deformations.
- Visualize the above with animations.

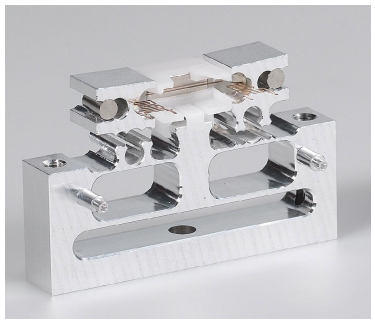


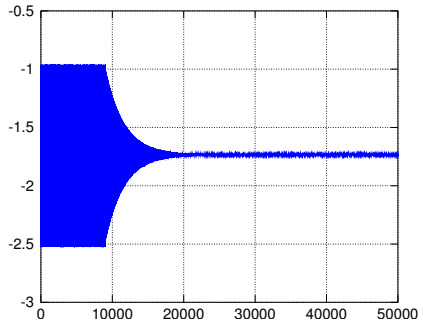


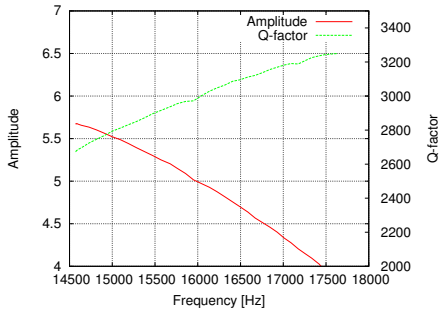


Examples of Engineering Applications V

One session is devoted to the analysis of the Q factor (damping) of a vibration cord based force sensor. Here I illustrate how to use `gnuplot` to generate special graphs.







How make *Octave* more Attractive for Engineers

Based on observations of and discussions with engineers and students the following points will make *Octave* even more attractive:

- GUI
- Make the transition from and to *Octave* as easy as possible
- Ease of installation
- Free, as in free beer
- Documentation
- Platform independent
- Free, as in free speech (open source)

This wish list is in decreasing order of importance of engineers[1].

[1] different from my ranking!

What can I give to the community?

All of the above would not be possible without the help of the great *Octave* community.

Thank you guys

It is only fair that I try to contribute too.

- Find the lecture notes for the above class on my web page `//staff.ti.bfh.ch/sha1/` in the *Octave* frame as the file `OctaveAtBFH.pdf`. Or use Google to search for this file.
- All data files and codes are available at `//staff.ti.bfh.ch/sha1/Labs/PWF/Codes/`
- On a few occasions I have contributed some code to *Octave* and its packages[2]

[2] The help and support you get from the community is amazing and beats any tech support from commercial companies I deal with!

That's all folks

Thank you for your attention