NTNU

Norwegian University of
Science and Technology

Experiences from PhD studies

Camille Hamon

25 November 2016

Top three things I would have done differently

— Better programming practices
  - Coding is what takes most of our time.
  - We are not educated or trained properly to program.
  - There exist well-recognized programming practices but we don't know them.
  - Especially: Version control and test-driven development

— Get off the computer and dedicate more time to set my research in perspective (instead of digging into coding right away)
  - Where does my research fit in?
  - What exactly do I want to solve?
  - Talk with different people (fellow PhD students, industry, . . .) $\Rightarrow$ different people = different target groups = formulate your problem differently = different perspectives

— Sometimes, the best you can do is to call it a day.

How I worked

- My working habits as a PhD student:
  1. Spend some time on thinking
     - What the problem is
     - How to solve it
  2. Spend a lot of time in Matlab
- In real-life:
  - Problem formulation and definition of case studies take the larger share of a project time
  - Coding comes late in a project

Defining your problem

Very important to define:

- The scope using well-accepted terms
  - Including time horizon of interest
  - Processes of interest
  - Where it fits in in today's real-life practices
  - Be clear about your assumptions: today's assumptions are future work!
  - When writing papers, this helps reviewers know exactly what you are addressing, and what you are not (equally important)
- What others have done
- What the industry is doing
- The broader picture and the more detailed picture

Defining your problem - 2

Very important to define:

- The scope using well-accepted terms
- What others have done
  - Important for yourself = where do you stand? Where do you fit in among all other research?
  - Keep a running literature review and come back to it and update it
  - Gives you a map of the field
  - Helps you identify the dimensions of the problem
  - Possible collaboration with others
  - It takes time but having a solid knowledge of the literature is very valuable
- What the industry is doing
- The broader picture and the more detailed picture

Literature review - my workflow

- Zotero + one-click import from browser (metadata + PDF)
- When starting research on a topic:
  1. I create a new folder in Zotero
  2. On Google scholar: enter different combinations of keywords related to your topics
  3. Identify the important papers (based on number of citations or familiarity with some authors' work) and work your way up from there or down
     - Way up: what papers have cited this one? the «cited by» feature in Google scholar is very useful!
     - Way down: what papers were cited by this one?
  4. Identify the main people / research groups
  5. Identify the main dimensions of the problems
  6. Create a Latex document where I write my notes and compare how the papers address the different dimensions of the problem
  7. Iterative process going back and forth between steps 2-6
- Difficulty: Filter the noise and identify the important research works
- The more you read, the more efficient you become at this

Defining your problem - 3

Very important to define:

- The scope using well-accepted terms
- What others have done
- What the industry is doing: Is some form of your problem addressed today by the industry?
  - If it is, how do they do it? Why do we need to do differently? What assumptions do they make? What can we do better? What are the shortcomings of the current way of handling the problem?
  - If it is not, why don't they do it? What makes the problem worth investigating? What could convince them of studying this problem?
- The broader picture and the more detailed picture

Defining your problem - 4

Very important to define:

- The scope using well-accepted terms
- What others have done
- What the industry is doing: Is some form of your problem addressed today by the industry?
- The broader picture and the more detailed picture
  - What do you contribute to by solving the problem?
  - Both pictures are equally important
  - Don’t forget reality!
  - Be clear about your assumptions (assumptions = gap between reality and proposed method / tool)
  - Think about how to transition to your novel method / tool from today’s practices
    - Roadmap from today’s practices to implementing and using your method = related to identifying the gaps
    - Ex: Data needs, etc . . . (got a lot of questions on this in conferences) => related to your assumptions

Defining your problem when writing a paper

- Remember that you have been working on your problem full time
- Others, including reviewers, **have not**
- Reviewers are familiar with existing work in the field
- Important to clearly identify your contributions with respect to other research works
- Tip from my PhD supervisor: write the introduction in four parts
  1. Define what problem you investigate from the general picture (more RES, power systems closer to the their limits, . . .) to the particular problem you are looking at (interarea oscillations, market clearing, . . .)
  2. what others have done
  3. what others have **not** done = research gap
  4. what you do = contributions

Importance of discussing with others

- Time spent discussing your project with others (and others' projects) = high return on investment
- Important to adapt to whom you speak
  - Good training to try to convey your messages to different target groups
- Fellow PhD students and academics
- Industry

Importance of discussing with others

- Time spent discussing your project with others (and others' projects) = high return on investment
- Important to adapt to whom you speak
- Fellow PhD students and academics
  - Read each others' papers, present to each other, . . .
  - Working in the same field
    - Discuss the details
  - Working in different fields
    - Explain what you are doing
    - Bigger picture
    - Important when you are caught up in the details of your problem
  - Attend the Friday seminars ☺
- Industry

Importance of discussing with others

- Time spent discussing your project with others (and others' projects) = high return on investment
- Important to adapt to whom you speak
- Fellow PhD students and academics
- Industry
  - Industry is usually interested but time constrained
  - Need to reformulate your problem in industry terms
  - Again, adapt to your target group
  - Ask specific questions
  - Very beneficial for the scope of your research and supporting your research topic / research gap
  - Need to find the right person

Coding

— We spend most of our time coding models, algorithms, . . .
— Our code defines what results we obtain
— We are self-taught programmers
— We are unaware of good coding practices

Good coding practices

Look at good programming practices (ex: `http://software-carpentry.org/lessons/`):

- Version control (git, subversion, . . .)
  - Keeps track of the full history of your code
  - Avoids having *myprogram_v1.m*, *myprogram_v1.2.m*, *myprogram_old.m* types of files
  - Works very well with Latex manuscript as well
  - Allows you to revert back to what worked after you broke something
  - Excellent for collaborative research (both for the code and the paper)
  - Makes it easy to pick up a project after a couple of months by showing you the last changes you made the last time your worked on a project
- Test-drive programming and unit testing
- Code review, pair programming
- Reproducible science

Good coding practices

Look at good programming practices (ex: `http://software-carpentry.org/lessons/`):

- Version control (git, subversion, . . .)
- Test-drive programming and unit testing
  - Write tests to check your code
  - Debug until all tests are passed
  - If you update your code, check if your tests are still passed
  - Confident booster that your program is working as intended, even after updating it
  - Turn unexpected behaviors into tests
- Code review, pair programming
- Reproducible science

Good coding practices

Look at good programming practices (ex: `http://software-carpentry.org/lessons/`):

- Version control (git, subversion, . . .)
- Test-drive programming and unit testing
- Code review, pair programming
  - Code review = review code from/by others
  - Pair programming = sit together and code to explain your code to others
    - Studies have shown that this is the single best way of finding errors and bugs
- Reproducible science

Good coding practices

Look at good programming practices (ex: `http://software-carpentry.org/lessons/`):

- Version control (git, subversion, ...)
- Test-drive programming and unit testing
- Code review, pair programming
- Reproducible science
  - You get results on 1 January 2015
  - You can reproduce them on 1 January 2016 (after the review came back or because somebody is interested in your code, ...)
  - You send your code to somebody that can reproduce the results in your paper
  - Uses all the above programming practices
  - Is becoming a hot topic

Coding - other tips

- Break down your code into functional units/blocks
  - One function to load the input data
  - One function to process the input data
  - One function to run what you are working on
  - One function to process the outputs and create graphics
  - . . .
- Easier to debug
- Easier to use: you don’t have to re-run the whole think every time
- Easier to maintain: you can add features in some blocks without changing the others

Coding: useful resources

- Single best resource: `http://www.software-carpentry.org`
  - Software carpentry promotes good coding practices for scientific computing
  - Free online lessons for Matlab, python, version control, R, ...
  - Recommended readings: `http://software-carpentry.org/reading/`
- Testing in Matlab
  - Don't do it yourself, Matlab has some builtin functions `http://se.mathworks.com/help/matlab/matlab-unit-test-framework.html?refresh=true`
- Other recommended readings: `http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745`

It takes time to learn new coding practices but it is time well spend.

Project management

- PhD = several year project involving reading, writing, coding, taking courses, ...
- Techniques exist for project and time management
- Gives more structure to your work
- Organisation and keeping track of todos and tasks
  - Better visibility of what is coming
  - Less worried that you have forgotten something
- Ex: define weekly goals and daily tasks
  - You make sure that all you have to do this week (course, paper writing, coding, ...) will be allocated time
  - Don't overestimate how much you can do in one week
  - Don't underestimate how much you can do in a year!
  - Gives you a good overview of you week and a feeling of control
  - You look back at it and realise that you have done a lot!
- Good course: *Learning about learning*: `https://www.coursera.org/learn/learning-how-to-learn`
  - Time management
  - Learning techniques

Tools

- Research note managers
  - Evernote
  - Org-mode in Emacs
  - Wikis
  - Word files
  - Latex files
  - Paper notebooks
- Task managers
  - Post-its, notebooks, apps, . . .
- Reference managers: zotero, mendeley, Excel, . . .
- Scientific computation
  - Matlab
  - Python
  - R
  - Julia
  - . . .
- **Remember that the tool should serve you and not you the tool**

Top three things I would have done differently

— Better programming practices
  - Coding is what takes most of our time.
  - We are not educated or trained properly to program.
  - There exist well-recognized programming practices but we don't know them.
  - Especially: Version control and test-driven development
— Get off the computer and dedicate more time to set my research in perspective (instead of digging into coding right away)
  - Where does my research fit in?
  - What exactly do I want to solve?
  - Talk with different people (fellow PhD students, industry, . . .) $\Rightarrow$ different people = different target groups = formulate your problem differently = different perspectives
— Sometimes, the best you can do is to call it a day.