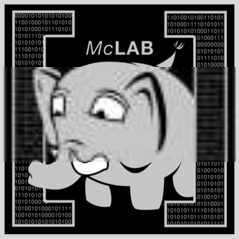
McLab Tutorial
www.sable.mcgill.ca/mclab



Laurie Hendren, Rahul Garg and Nurudeen Lameed

Other McLab team members:

Andrew Casey, Jesse Doherty, Anton Dubrau, Jun Li, Amina Aslam, Toheed Aslam, Maxime Chevalier-Boisvert, Soroush Radpour, Oliver Savary , Maja Frydrychowicz, Clark Verbrugge

Sable Research Group
School of Computer Science
McGill University, Montreal, Canada

This tutorial is intended to provide an overview of the challenges of compiling MATLAB and the tools provided by McGill's McLab project. Please feel free to reuse these slides, however please make sure you credit the authors of the slides and that you indicate the source of the original slides.

Tutorial Overview

- Why MATLAB?
- Introduction to MATLAB – challenges
- Overview of the McLab tools
  - Introduction to the front-end and extensions
  - IRs, Flow analysis framework and examples
  - Back-ends including the McVM virtual machine
- Wrap-up

This tutorial starts with an exploration of why it is important for compiler/PL researchers to work on MATLAB and languages like MATLAB.

We then proceed to an introduction to the MATLAB language, and we illustrate some of the challenges of dealing with MATLAB.

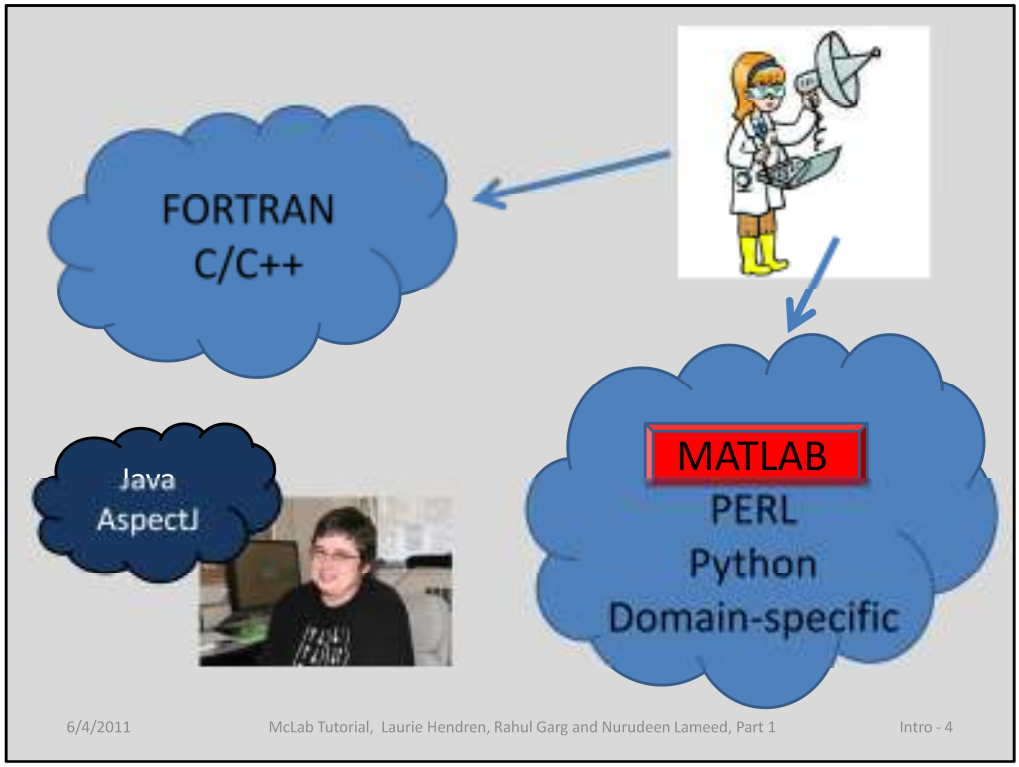
The main body of the tutorial is composed of an introduction to the McLab toolset. We will give an introduction to the front-end and how it can be used to build MATLAB extensions, then we introduce our two IRs, McAST a high-level AST and McLAST a lower-level AST. We then move to an overview of our back-ends, with a particular focus on McVM and McJIT. Finally, we will give a short wrap-up.

Nature Article: "Why Scientific Computing does not compute

- 38% of scientists spend at least 1/5th of their time programming.
- Codes often buggy, sometimes leading to papers being retracted. Self-taught programmers.
- Monster codes, poorly documented, poorly tested, and often used inappropriately.
- 45% say scientists spend more time programming than 5 years ago.

October 2010 article in Nature, by Zeeya Merali. Survey of 2000 scientists.

It is important that compiler/PL researchers aim to provide programming languages and systems that both provide:
• programming environments in which scientists can program easily
• systems that lead to solid and extensible code.



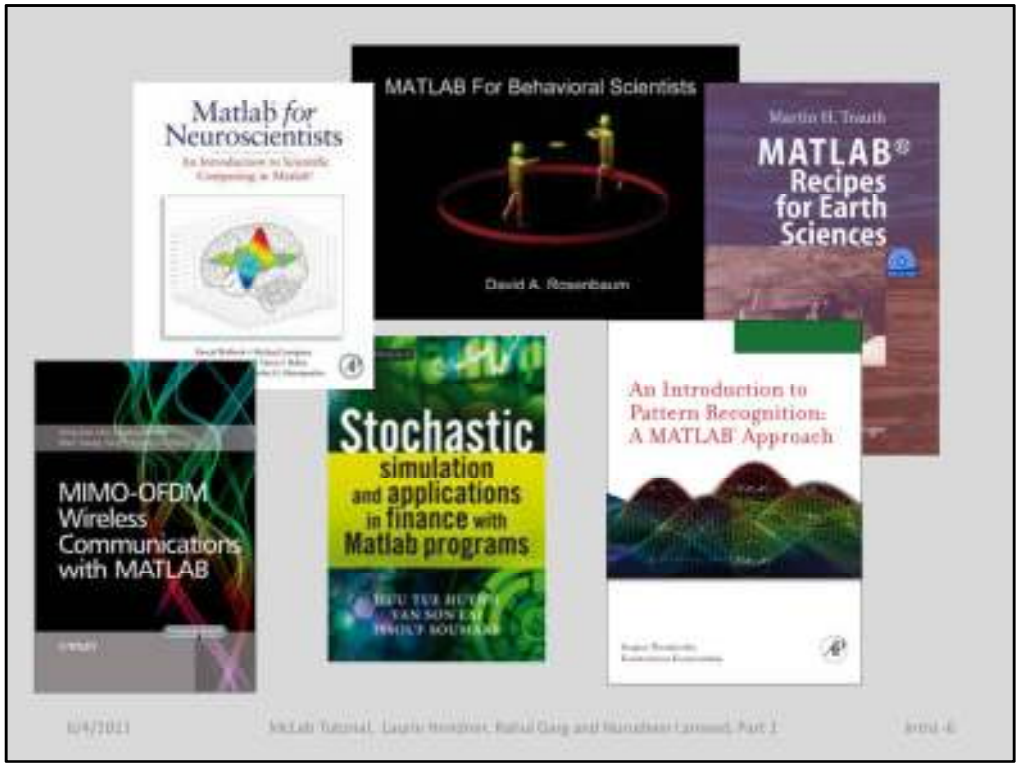
Scientist in upper right ... Many different applications to program, which language to pick? Increasingly picking dynamic or scripting languages. Many scientific and engineering computations use MATLAB.

Computer Scientist, Compiler writer, lower left. Has worked on compilers and tools for object-oriented and aspect-oriented languages ... But scientists are not interested in these languages.

A lot of MATLAB programmers!

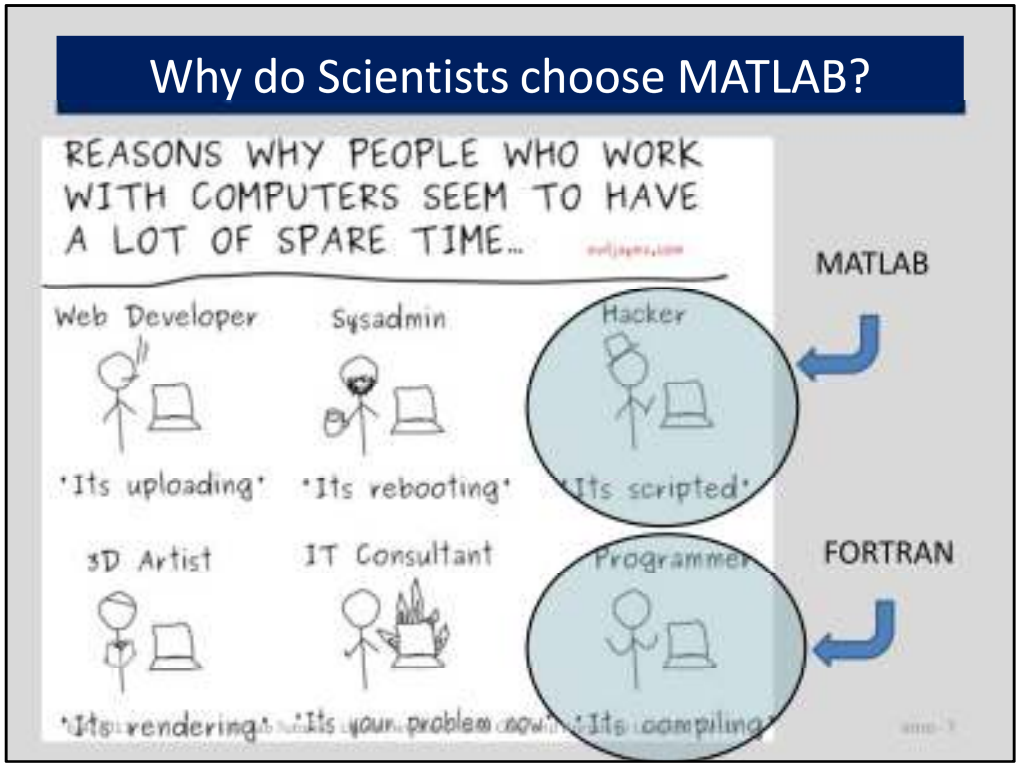
- Started as an interface to standard FORTRAN libraries for use by students.... but now
  - 1 million MATLAB programmers in 2004, number doubling every 1.5 to 2 years.
  - over 1200 MATLAB/Simulink books
  - used in many sciences and engineering disciplines
- Even more "unofficial" MATLAB programmers including those using free systems such as Octave or SciLab.

There are a lot of MATLAB users, shouldn't we be doing something for them?



Check out the number and variety of disciplines ….

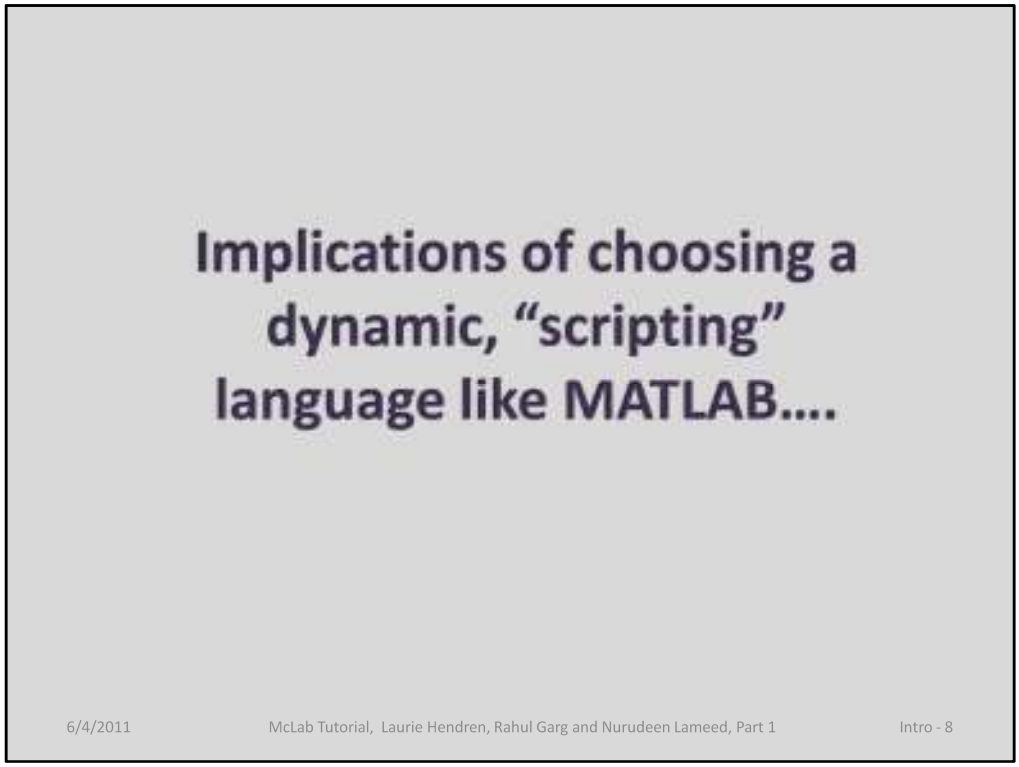
Books are often "how to" in terms of using MATLAB. We also need some books that describe MATLAB in way that both uses solid PL terminology and foundations, but also talks about the domain-specific applications.



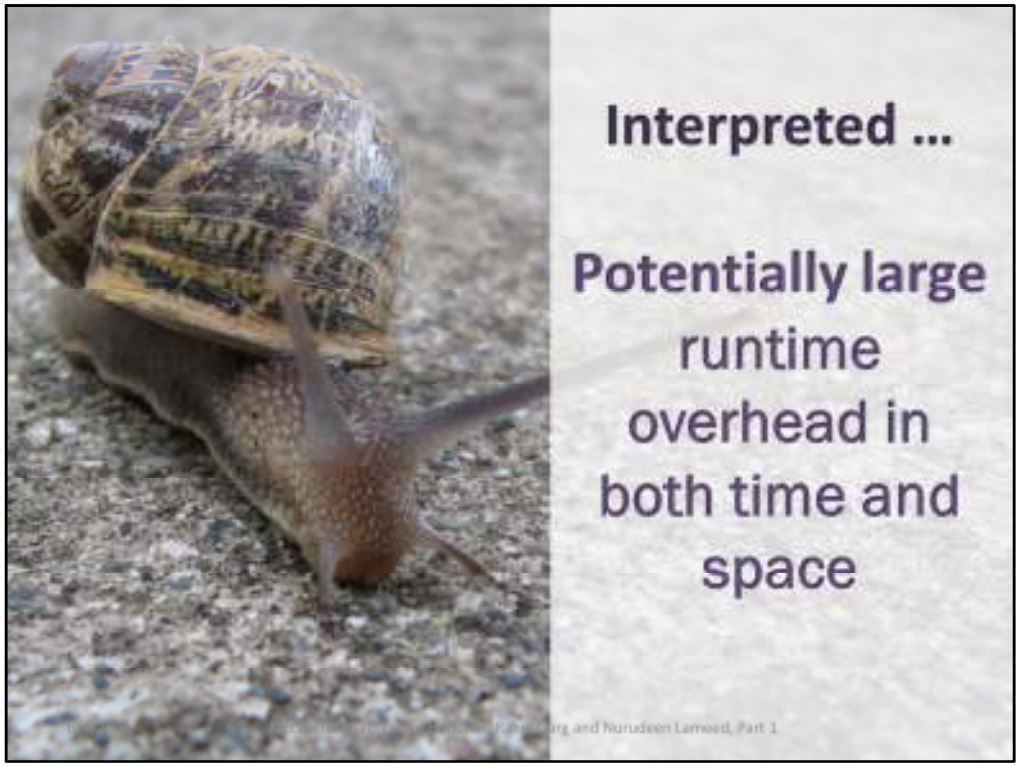
Why do scientists choose MATLAB?

Why not something like FORTRAN? - advantages are good compilers, efficient execution.

But programmers are choosing MATAB – faster prototyping – no types, lots of toolboxes, interactive development style...

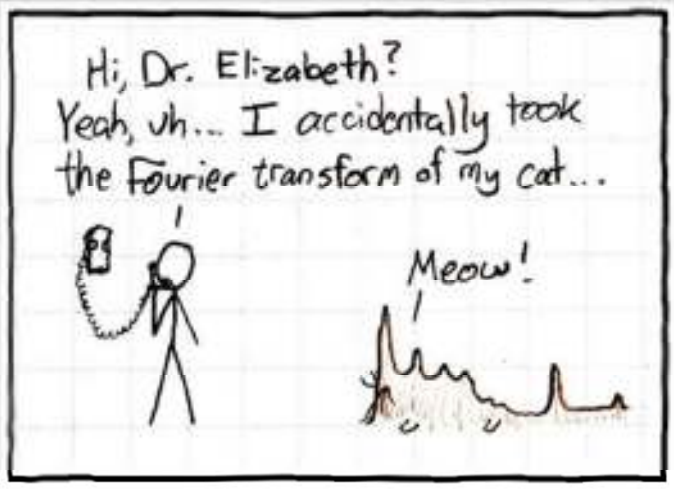


Although Scientists like the interactive and "wild west" development style of MATLAB, what are the implications of choosing a dynamic "scripting" language like MATLAB?



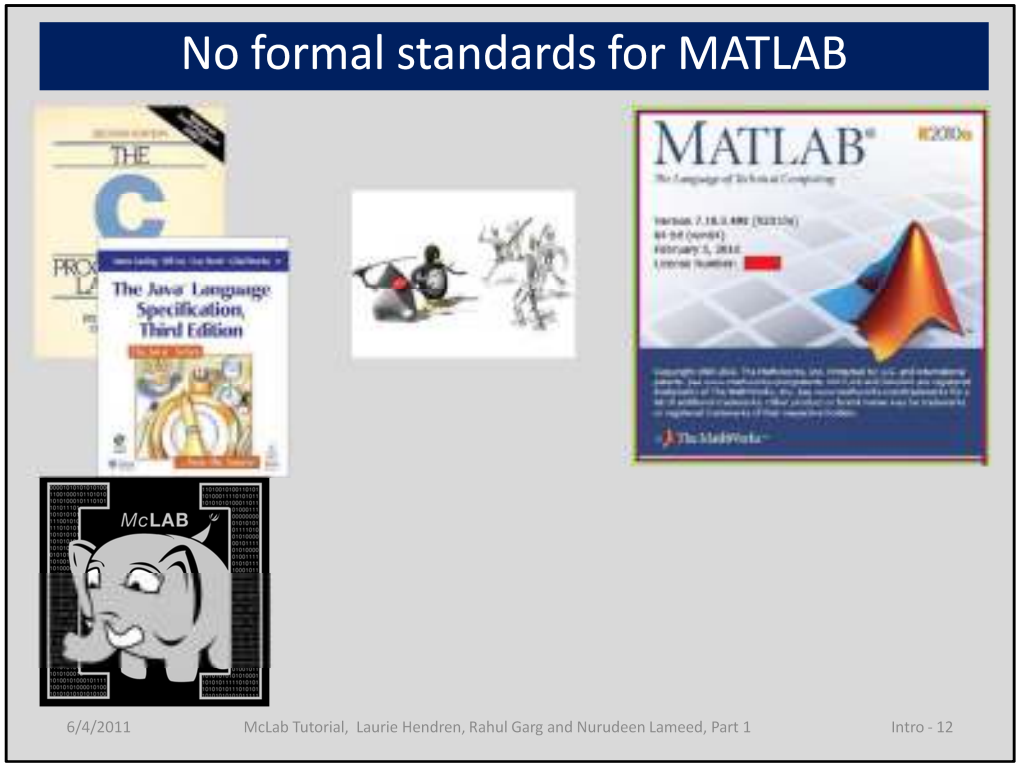
Original implementation by MATHWORKs interpreted, their system now contains a JIT (which they call an "accelerater"). Open implementations like Octave and Scilab are interepreted.

No types and "flexible" syntax



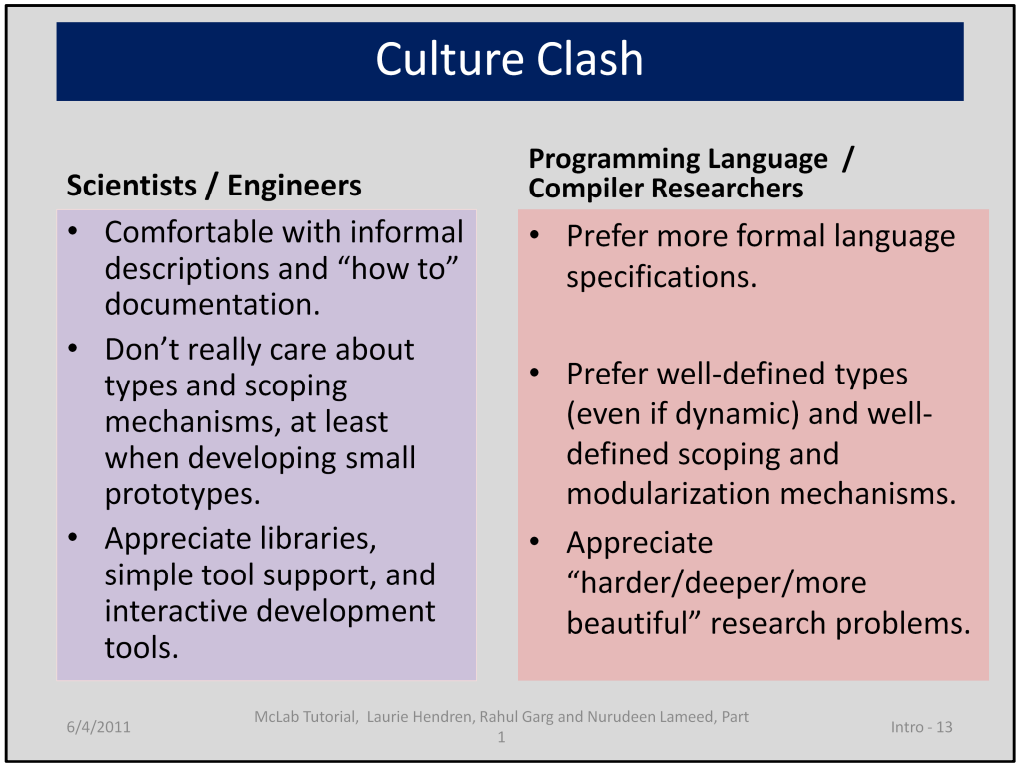
MATLAB often computes something, even if it was not was intended.

MATLAB programmers get very few static guarantees, but quite often program in some dynamic checks.



Lack of a standard – the semantics can change in a new release from Mathworks.

If the research community can help distill out a proper specification, it will enhance research opportunities and perhaps encourage some standardization.



PL and compiler researchers need to consider the scientific community and their perspective.

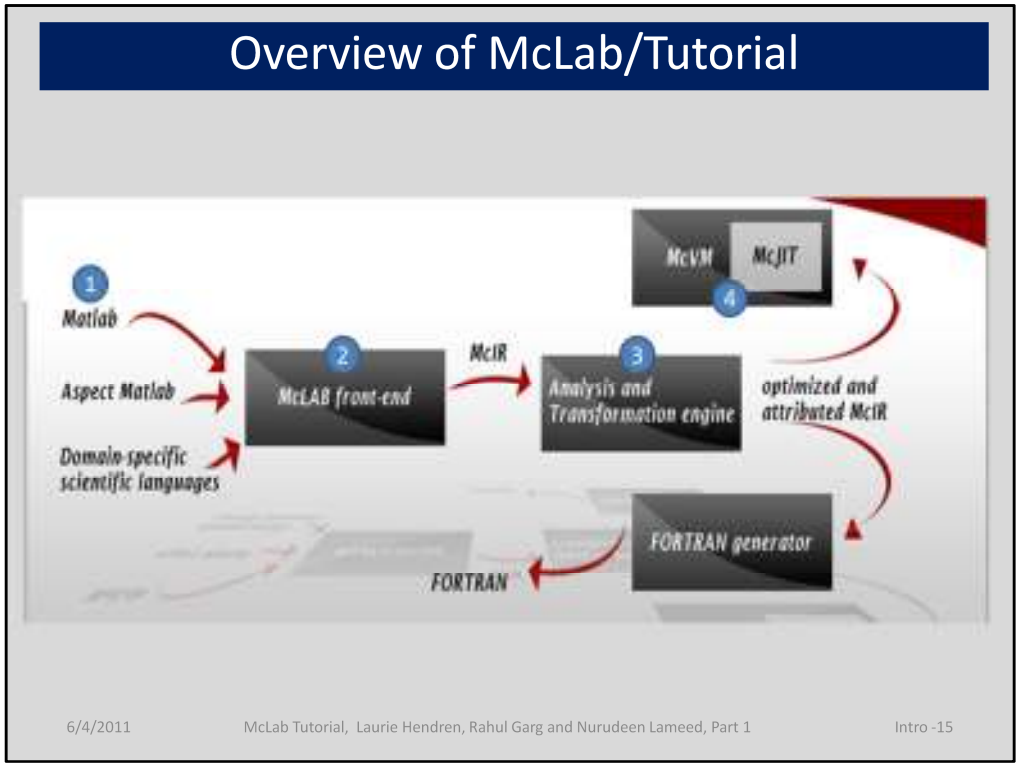
What can we do to enhance their programming experience, while still doing interesting research from a CS perspective?

Does the PL/compiler community need to broaden their perspective of what is useful/good research?

Goals of the McLab Project

- Improve the understanding and documentation of the semantics of MATLAB.
- Provide front-end compiler tools suitable for MATLAB and language extensions of MATLAB.
- Provide a flow-analysis framework and a suite of analyses suitable for a wide range of compiler/soft. eng. applications.
- Provide back-ends that enable experimentation with JIT and ahead-of-time compilation.

Our goals are to provide an infrastructure that supports the research for MATLAB, and to do such research ourselves.



The rest of this tutorial will:
(1) Introduce MATLAB
(2) Give an overview of the McLab front-end and a small example of an extension
(3) Give an overview of the IRs and the Analysis Framework
(4) Discuss the back-ends, concentrating on McVM/McJIT