# MIX10: A MATLAB TO X10 COMPILER

*by*

*Vineet Kumar*

School of Computer Science
McGill University, Montréal

Thursday, October 31st 2013

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

# Abstract

TBD

# Résumé

TBD

# Acknowledgements

TBD

# Table of Contents

## Appendices

# List of Figures

# List of Tables

# Chapter 1
# Introduction

MATLAB is a popular numeric programming language, used by millions of scientists, engineers as well as students worldwide[Mol]. MATLAB programmers appreciate the high-level matrix operators, the fact that variables and types do not need to be declared, the large number of library and builtin functions available, and the interactive style of program development available through the IDE and the interpreter-style read-eval-print loop. However, even though MATLAB programmers appreciate all of the features that enable rapid prototyping, their applications are often quite compute intensive and time consuming. These applications could perform much more efficiently if they could be easily ported to a high performance computing system.

On the other hand, X10 is an object-oriented and statically-typed language which uses cilk-style arrays indexed by *Point* objects, and has been designed with well-defined semantics and high performance computing in mind. X10 compiler can generate C++ or Java code and supports various communication interfaces including sockets and MPI for communication between nodes on a parallel computing system.

In this thesis we present MIX10, a source-to-source compiler that helps to bridge the gap between MATLAB, a language familiar to scientists, and X10, a language designed for high performance computing systems. MIX10 statically compiles MATLAB programs to X10 and thus allows scientists and engineers to write programs in MATLAB (or use old programs already written in MATLAB) and still get the benefits of high performance computing without having to learn a new language. Also, systems that use MATLAB for

prototyping and C++ or Java for production, can benefit from MIX10 by quickly converting MATLAB prototypes to C++ or Java programs via X10. *Figure 1.1* shows the flow of compilation from MATLAB to executable code via MIX10. In particular, this thesis identifies the key challenges in compiling a dynamically-typed language like MATLAB to a statically-typed object-orinted high performance computing language like X10 and our approach to compiling MATLAB to X10.

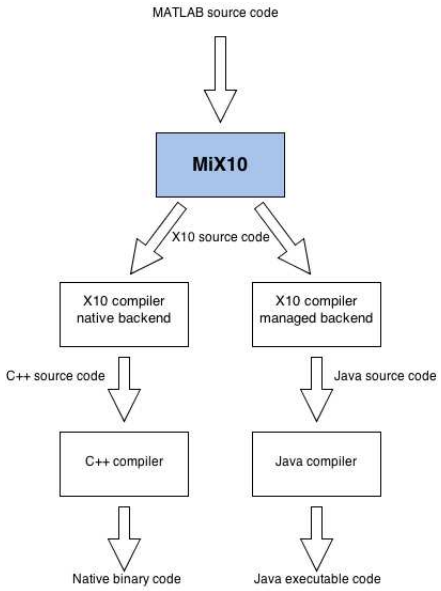


**Figure 1.1** Compilation flow via MIX10

On one hand, all the aforementioned characteristics of MATLAB make it a very user-friendly and thus popular application to develop software among a non-programmer community. On the other hand, these same characteristics make MATLAB a difficult language to write a static compiler for. Lack of formal language specification, unconventional semantics and closed source make compiler writers' task even harder. Mathworks implementation of MATLAB is essentially an interpreter with a *JIT accelarator* which is generally slower than statically compiled languages. Built on top of *McLAB* frontend and static analysis tools, MIX10 provides static compilation for MATLAB via the C++ backend for X10 and thus ultimately aims to have better performance even with sequential code.

## 1.1 Contributions

The major contributions of this thesis are as follows:

**Identifying key challenges:** We have identified the key challenges in performing a semantics-preserving translation of MATLAB to X10.

**Overall design of MIX10:** We provide the design of a source-to-source translator, building upon the McLab front-end and analysis toolkits.

**MIX10 IR design:** In order to provide a convenient target for the first level of translation, we have defined a high-level MIX10 IR. This IR is used for code generation, X10 specific analyses, code simplifications and transformations.

**Comparison of different kinds of X10 arrays:** version 2.4 of X10 (latest version as of this writing) provides two kinds of multi-dimensional arrays, *region-based* arrays and *rail-backed* arrays.

**Static analyses:** We developed various static analyses to aid generation of better optimized code and to support wider range of MATLAB functionalities. Identification of complex numerical values, handling variables with type-conflicts, array-bounds checks and identification of non-mutable variables are the important analyses that we describe in this thesis.

**Template-based builtin framework:** MATLAB supports many builtin operations that can operate over a wide variety of run-time types. We have designed and implemented a template-based system that allows us to generate specialized X10 code for a collection of important builtin operations.

**Code generation strategies for key language constructs:** There are some very significant differences between the semantics of MATLAB and X10. A key difference is that MATLAB is dynamically-typed, whereas X10 is statically-typed. Furthermore, the type rules are quite different, which means that the generated X10 code must include the appropriate explicit type conversion rules, so as to match the MATLAB

semantics. Other MATLAB features, such as multiple returns from functions, a non-standard semantics for `for` loops, and a very general range operator, must also be handled correctly.

**Working implementation and performance results:** We have implemented the MIX10 compiler over various MATLAB compiler tools provided by *McLAB*. In the process we also implemented some enhancements to these existing tools. We provide performance results for different X10 backends over a set of benchmarks and compare them with results from other MATLAB compilers including Mathworks MATLAB implementation and Octave.

## 1.2 Thesis Outline

This thesis is divided into 10 chapters, including this one and is structured as follows.

*Chapter 2* provides an introduction to the X10 language and describes how it compares to MATLAB from the point of view of language design. *Chapter* **??** gives a description of various existing MATLAB compiler tools upon which MIX10 is implemented. *Chapter 3* gives a high-level design of MIX10 and explains the design and need of MIX10 IR. In *Chapter* **??** we introduce different types of arrays provided by X10 and describe code generation strategies for them. We also identify pros and cons of both kinds of arrays in the context of X10 as a target language. *Chapter* **??** presents the template-based specialization framework for handling MATLAB builtin methods. *Chapter* **??** describes the various static analyses that we implemented to help generate more optimized code. In *Chapter 6* we give details of code generation strategies for important MATLAB constructs. *Chapter 8* provides performance results for code generated using MIX10 for a suite of benchmarks. *Chapter 9* provides an overview of related work and *Chapter 10* concludes and outlines possible future work.

# Chapter 2

# Introduction to X10 and contrast with MATLAB

TBD

# Chapter 3
# Background and High level design

TBD

# Chapter 4
# Compiling for high performance of sequential code

---

TBD

# Chapter 5
# Concurrency constructs in MiX10

TBD

# Chapter 6
# Code generation

TBD

# Chapter 7
# isComplex value analysis

---

TBD

# Chapter 8
# Evaluation

TBD

# Chapter 9
# Related work

# Chapter 10
# Conclusions and Future Work

TBD

# Appendix A
# XML structure for builtin framework

---

TBD

# Appendix B
# isComplex analysis Propagation Language

---

TBD

# Appendix C
# MiX10 IR Grammar

---

TBD

# Bibliography

[Mol] Cleve Moler. The Growth of MATLAB and The MathWorks over Two Decades. http://www.mathworks.com/company/newsletters/news_notes/clevescorner/jan06.pdf.