

# COMP 621 Suggested Projects - Winter 2012

Due Dates:

- Project descriptions available and signups, Tuesday Feb 7 to Friday Feb 10
- Project proposal due, Thursday Feb 16
- Project update #1 due, Tuesday March 6
- Project update #2 due, Thursday March 22
- Project presentations start: Tuesday April 10
- Project reports due, Tuesday April 17

## 1 Introduction

The purpose of the final project is to give you a chance to focus on a specific problem, and to work out the solution to this problem in detail. Your project should give you the opportunity to review the current literature in your chosen area, to implement either the ideas you read about or to implement your own improvements on those ideas, and to evaluate the effectiveness of the ideas. You are encouraged to try to develop improvements over the current state-of-the-art. Even if your ideas do not turn out better, it is still worth trying (after all, that is what research is all about). It is worth noting that publishing novel work done in this course is not out of the question.

I have listed the various possible project areas below. Students should work individually. Projects must be done specifically for this course and should not contain material used for other course projects.

The projects will be given out on a first-come, first-served basis. You may sign-up for your selected area on the sheet outside of my office, McConnell 228. You are also free to design your own project, or to change a suggested project to more suit your interests. If you wish to modify or design your own project, please contact the instructor to discuss the project.

Your project proposal should be done using the document outline in:

<http://www.sable.mcgill.ca/~hendren/621/Docs>

Latex documents are preferred, but not absolutely necessary. In any case, you should follow the overall organization given in <http://www.sable.mcgill.ca/~hendren/621/Docs/prop621.tex>.

## 2 Suggested Projects

### 2.1 Detect loops that correspond to known and efficient library routines

Mentor: Laurie (hendren@cs.mcgill.ca)

Problem Statement: MATLAB is an interactive programming environment for scientific computing. It is designed for vector and matrix computation. Naively written MATLAB code can suffer a performance slowdown. One way to address the performance problem of high-level domain-specific languages, such as Octave or MATLAB is to use optimized library routines instead performing operations on vectors and matrices iteratively (using loops).

While it is tempting for those with experience in traditional programming languages to use explicit loop constructs for mathematical operations, this temptation should be resisted strenuously. For example, instead of:

```
port_val = 0;
for j = 1:n
    port_val = port_val + ( holdings(j) * prices(j));
end
```

Instead:

```
port_val = holdings*prices;
or
port_val = mtimes(holdings,prices);
```

The latter is more succinct, far clearer, and will run much faster if the appropriate library calls are made by the MATLAB system. The goal of this project is to develop one or more analyses that detect loops that correspond to known and efficient library routines, and to automatically rewrite the loops to higher-level code that will be compiled to an efficient library routine.

See the paper at <http://portal.acm.org/citation.cfm?id=1267941> for one relevant reference.

### 2.2 Loop Vectorization

Suggested by Nurudeen (nlamee@cs.mcgill.ca).

Please note that this project is a variation of the one above. If both of these projects are taken, then I would expect students to take different approaches to the problem.

MATLAB is designed for vector and matrix operations, and one way of speeding up the performance of MATLAB programs is to express loops as vector operations. As an example consider the following MATLAB code that builds a table of  $\log(x) + \sin(x)$ , with the values of  $x$  ranging from 0 to 10 and with an interval of 0.001. The loop performs 10001 iterations.

```
i = 1
for x = 0:0.001:10
    y(i) = log(x) + sin(x);
    i = i + 1;
end
```

This loop could be rewritten as

```
x = 0:0.001:10;
y = log(x) + sin(x);
```

Other more complex loops are not easily vectorizable. If it is possible to determine certain properties (e.g., loop bounds, iteration count) of a loop, including those of the statements in the loop, it may be possible to rewrite some loops as vector operations.

The goal of this project is to develop an analysis that determines whether a loop can be rewritten as vector operation(s). This involves determining loop properties that are useful for the analysis.

## 2.3 Loop-Invariant Code Motion

Suggested by Nurudeen (nlamee@cs.mcgill.ca).

Typically, results of computations in loops change with loop iterations. However, it is not uncommon to find computations in loops whose results are fixed throughout the entire loop iterations. Such computations could be executed once and the result stored in a temporary variable before the loop. However, care must be taken when moving computations out of a loop. We must be mindful of the fact that code that would not have been executed without the optimization must not be executed. Consider the following code snippet:

```
...
5: while i < 10
...
10: a(i) = t + c;
...
end
```

assuming that neither  $t$  nor  $c$  is redefined within the loop, we could transform the program by assigning  $t + c$  into a temporary variable and use the result in line 10 above. The goal of this optimization is to find expressions or statements that could safely be moved out of a loop. Certain considerations, such as side-effects of computations, must be made to determine when it is safe to move computations out of a loop.

This analysis and transformation could be done within the McLab front-end and could be used as a source-to-source transformation.

## 2.4 Programmer tools for Matlab

Mentor is Laurie (hendren@cs.mcgill.ca).

### 2.4.1 Motivation

The Matlab language gives a lot of freedom to the programmer. Unfortunately much of this freedom can lead to poor performance or unintuitive behaviour. Growing arrays can cause excessive copies of the array being grown; nested functions and variables can be very unintuitive; redefining loop variables in the body of a for loop could be confusing to programmers not used to such for each loops. There are also various style issues one might want to warn about. These can include never using a definition of a variable and strange use of semicolons in scripts.

Matlab currently has tool for this. It is called mlint. But since we already have a system capable of parsing and analyzing Matlab it would be interesting and useful to implement and experiment with this functionality using the McLab frontend and analysis framework.

### 2.4.2 Description

The goal of this project is to develop a set of analyses used to generate warnings to the programmer. These warnings should be for style, performance and sanity issues. Sanity issues include things that might be confusing to the programmer, to make sure that they are aware that Matlab might do some strange things.

## 2.5 Automated Refactorings

Mentor is Laurie (hendren@cs.mcgill.ca)

Many modern languages have tools to help programmers refactor their code. Examples are “rename method”, “extract method” and so on. Recent work has supported some key refactorings for Matlab (see Laurie for a draft of a M.Sc. thesis) which include “inline function”, “inline script”, “convert script to function” and “convert feval to function call”.

The purpose of this project would be to identify some further refactorings useful for Matlab programmers, to design and implement the analyses required to determine when such a refactoring is semantics-preserving, and to implement the actual refactoring transformations.

## 2.6 Improving AspectMatlab Performance

Mentor is Laurie (hendren@cs.mcgill.ca)

In Assignment 1 we have seen some limitations of AspectMatlab, and that weaving aspects can introduce significant runtime overheads. The purpose of this project is to improve AspectMatlab both in functionality and in speed. Previously written aspects, plus aspects used in Assignment #1, can be used for benchmarks.

## 2.7 Faster Intraprocedural Analyses

Mentor is Laurie (hendren@cs.mcgill.ca)

McLAB has an existing analysis framework with very general functionality. However, the speed of the analyses are not likely to be optimal. The purpose of this project is to design and implement faster versions of the analysis framework. Faster versions may work for only specific kinds of analyses (for example, those which have pre-computed gen and kill sets) or for functions that have a specific structure (a completely well-structured function, with well-defined loop behaviour).

## 3 Propose a Project

You may also propose a new project that is not listed here. If you choose something else, it should have some element of “program” analysis in it, although the “program” being analyzed could be something from a different domain such as models or games.

If you are proposing your own project, please e-mail the instructor (hendren@cs.mcgill.ca) with a brief project proposal similar to the ones above, and add your project title to the list below.

- 3.1 Student Proposed Project 1**
- 3.2 Student Proposed Project 2**
- 3.3 Student Proposed Project 3**
- 3.4 Student Proposed Project 4**
- 3.5 Student Proposed Project 5**
- 3.6 Student Proposed Project 6**