

Re: EE Student, Edit, Proposal Masters, Help (concepts of functional programming, symbolic programming and MATLAB)

Source: <http://sci.tech-archive.net/Archive/sci.math.symbolic/2005-03/0095.html>

From: John Creighton (*JohnCreighton__at_hotmail.com*)

Date: 03/09/05

Date: 8 Mar 2005 20:51:09 -0800

israel@math.ubc.ca (Robert Israel) wrote in message news:<d0l8tc\$6sh\$1@nntp.itservices.ubc.ca>...

> *You might start with the Maple 9.5 "Introductory Programming Guide".*

>

> *Robert Israel israel@math.ubc.ca*

> *Department of Mathematics <http://www.math.ubc.ca/~israel>*

> *University of British Columbia Vancouver, BC, Canada*

Okay, I'll take a look. I have been listening to the feedback and trying to take others suggestions into account as best as possible. Unfortunately, I might end up cutting it out of my proposal. I have expanded the history of the Kalman filter section and have now exceeded the 8 page limit by a page. I might be able to include it as an appendix. I will definitely be able to include it in the body of my final thesis. Currently my proposal has the following structure:

1 INTRODUCTION 3
2 LITERATURE REVIEW 4
2.1 THE HISTORY OF THE KALMAN FILTER 4
2.2 THE HISTORY OF DESCRIBING FUNCTIONS 4
2.3 A FEW COMMENTS ON SYMBOLIC AND FUNCTIONAL PROGRAMMING 5
3 OBJECTIVE 6
4 SIGNIFICANCE 7
5 THE KALMAN FILTERS IN THE CONTEXT OF ESTIMATION THEORY 7
5 PRELIMINARY RESULTS 9
6 CONCLUSIONS 10
REFERENCES 10

Bellow is the update to my update to section 2.3:

1.1.3 A few comments on Symbolic and Functional Programming

Symbolic programming languages fall under two categories, functional (e.g. lisp, Maple) and logical (e.g. PROLOG). In functional

programming languages each program is a function similar to a mathematical function. A program is made up of a composition of functions [9]. The arguments to the functions can include the data types, lists, expressions, primitives and variables. As in all languages data types are built up of elemental units called primitives. However, in some third generation languages (e.g. Maple) the primitives are not part of the language. This is analogous to assembler and machine code falling outside the scope of a second generation language. Maple uses a numeric data type which can be of arbitrary precision. The software must decide how to reduce the higher level computation down to machine level operations on primitive data.

Unlike pure functional languages such as Haskell, Maple uses assignment instead of monads. In Maple if a variable x is assigned a variable y then x points to y . If y is then assigned the value z then x points to y which points to z . Programmatically the value at the end of the chain of assignment can be returned by evaluating a variable in the chain. The assignment chain is not altered by evaluation. It is only altered when one of the variables in the chain is reassigned. So for instance the chain could be altered by assigning the first variable in the chain the value at the end of the chain as follows:

```
x:=eval(x)
```

Variables can represent many kinds of data structures. A data structure that can be evaluated is known as an expression. An expression could be formed from a single variable, a single function at a given domain or though the composition of several expressions.

Another data structure is a list. Linguistically a list is an ordered collection of items. In computer science a list usually refers to a data structure that is easy to add and remove stuff from but must be accessed in a sequential fashion. In a purely functional language a list can be defined by the composition of functions in a recursive manner. For instance the list function could take two arguments where one argument is the first element of the list and the next argument is the rest of the list. Since, a list can be represented as a composition of functions; functions can be evaluated and; expressions are something that can be evaluated then: a list could be thought of as an expression. Conversely an expression could be represented with a composition of lists by letting the first element of the list be a function name or reference and the subsequent elements be the arguments where the arguments can also be lists of the same format.

When the arguments of the some functions in an expression are expressions it becomes natural to represent the overall expression as a tree. This representation is known as the operation tree. Given that in an operation tree the outer function is the easiest to access, there are certain efficiency advantages to evaluating the outer function first. For instance if the outer function involves the multiplication of an expression by zero it may not be necessary to

know the exact nature of the expression which is being multiplied by zero to determine the result. The evaluation of the outer function (e.g. multiplication) first and only evaluating the arguments if needed is referred to as lazy evaluation. A function programming language which does not use lazy evaluation is known as a strict functional language. Functional programming languages are well suited to mathematic symbolic packages such as Maple and Mathematica.

The symbolic package for MATLAB performs symbolic operations by storing symbolic expressions as objects. MATLAB manipulates expressions by: passing the expression to the Maple kernel and then storing the result as a symbolic object [7]. The Method used in MATLAB is not the most efficient way to perform symbolic operations but was chosen given the limitation of MATLAB in dealing with variable references and pointers. For the proposed research the efficiency limitations are not usually a serious concern since the symbolic computations must only be done in the design phase. However, parsing large symbolic expressions can be a problem and there is a size limitation on the symbolic variables in MATLAB.