MATVEC:

MATRIX-VECTOR COMPUTATION PROGRAMMING LANGUAGE PROPOSAL

John C. Murphy
jcm2105
Programming Languages and Translators
Professor Stephen Edwards

MATVEC: MATRIX-VECTOR COMPUTATION PROGRAMMING LANGUAGE

INTRODUCTION:

Many programming languages today do not have the built-in functionality to easily handle the manipulation and computation of matrix and vector mathematics. Matrices and vectors are mathematical tools that are used in many different disciplines. Having a programming language that has easily usable tools and constructs for handling matrices and vectors can be very valuable. Although other tools exist today that can perform matrix/vector computations (such as Matlab), these tools often require an expensive license and require some time to learn the software package. This language has easy to use syntax that is set up to perform matrix and vector mathematical computations.

MAIN FEATURES:

SIMPLE:

This programming language provides easy-to-use, self-explanatory lexical syntax for dealing with matrices and vectors. The language will also provide an easy means to normalize vectors. The language will also provide flow-control constructs that will allow the programmer to loop through different code sections a fixed or conditional amount of times. Any novice programmer should be able to pick up and use the programming language rather quickly.

PORTABLE:

Since the programming language will be converted into Java byte code by ANTLR, the code will be executed by the Java Virtual Machine. Thus, this programming language can be run on any platform that is supported by Java.

HIGH-LEVEL FUNCTIONALITY:

The programmer will be able to easily input the values of matrices and vectors and use flow control structures within the language to further manipulate the program variables. The programmer will be able to add, subtract, multiply, and divide two matrices together using the easy-to-use program syntax. The programmer will also be able to normalize a vector if needed. Normalizing a vector simply means that the length of the vector (through any number of dimensions) is equal to one. The program will limit the normalizing of a vector to three-dimensional space.

DATA TYPES:

For the purposes of the computation, all values within a matrix or vector will assume that the values are real values. Any integer values that are input into the matrix will

automatically be converted into type real. Any variable that is not specifically declared will be assumed to be of type real.

A new data type will be established for matrix and vector computations. The data structure will be either of type matrix or vector. These will be declared at the beginning of the program.

ERROR CHECKING:
The translator will check that no invalid matrix computations are attempted. For example, addition and subtraction of two matrices requires that both matrices are identical in size. If the programmer attempted to perform a matrix addition on matrices of two different sizes, the compiler would return the appropriate error. The translator would also check whether the appropriate sizes for both matrices and vectors are used when multiplying or dividing.

The lexer from ANTLR will also check that the proper syntax and tokens have been used in the language before attempting to translate the program into Java byte code.

SCOPE:
For the purposes of computation time, the size of the matrices should be limited to 3 x 3 matrices. The error checker will make sure that the size of the matrices and vectors do not exceed this limit. For the purposes of demonstrating the functionality of the programming language, this should be a reasonable limitation.

CONTROL STRUCTURES:
The programming language will support flow control constructs such as the WHILE loop and FOR loop.

SAMPLE CODE:
The following code is an example of a typical program.

```
matrix m = [1 2 3 | 4 5 6 | 7 8 9];
matrix n = [4 5 6 | 9 8 7 | 3 2 1];
vector v = [5 8 4];
matrix p;

/* Basic matrix mathematics */
i = 0;
p = m + n;
normalize(v);

/* Loop control structure */
```

```
while (i < 5) {
        p = p * v;
        p = p + [1 1 i];
        i = i + 1;
}

/* Print out the matrix */
print p;
```

POSSIBLE EXTENSIONS:

Time-permitting, the following features may be added to the language:

- Use of JFreeChart to display any three-dimensional vectors in space starting from the origin of the coordinate system.
- Input matrices and vectors from a file which would allow the program to process a number of different matrices at runtime.

CONCLUSION:

The purpose of MATVEC is to provide a simple-to-use programming language that can be used to perform mathematical operations on vectors and matrices. The language also provides a function for normalizing vectors as well. With the easy-to-use syntax of the language, a novice programmer should be able to perform various arithmetic computations on matrices and vectors.