Quaternion Calculator

System Manual

Document Version:1.0

Date: 3/4/2014

Contents

[Table of Contents]

1 Document Management

1.1 Contributors

Please provide details of all contributors to this document

Role	Unit	Name
Project Leader	*Team 4*	*Scott Tilson*
Project Analyst	*Team 4*	*Josh Werner*
Billing	*Team 4*	*Brice Thrower*
Business Analyst	*Team 4*	*Murl Westheffer*
Programmer	*Team 4*	*Alexa Varady*
Programmer	*Team 4*	*David Young*
Documentation	*Team 4*	*Megan Teahan*
Documentation	*Team 4*	*Patrick Walter*
Documentation	*Team 4*	*Su Yan*
Support/Testing	*Team 4*	*Sean Stout*

1.2 Version Control

Please document all changes made to this document since initial distribution.

[Table of Contents]

Date	Version	Author	Amendment
1/30/2014	**1.0**	Alexa & David	Initial creation of quaternion.rkt file
1/31/2014	**1.0**	Alexa	Added many of basic calculation functions to quaternion.rkt file.
2/1/2014	**1.0**	Alexa	Created /fixed quaternion-diff and quaternion-div.
2/3/2014	**1.0**	Alexa & David	Creation of website (web-server.rkt). Fixed bugs in quaternion-prod function.
2/4/2014	**1.0**	Alexa	Created sin/cos functions to program
2/5/2014	**1.0**	Alexa & David	Added cos/sin to eval.rkt. Fixed bugs in quaternion-prod. Added graphical example
2/6/2014	**1.0**	Alexa & David	Formatting of files for readability. Added support for negative numbers. Bug fixes with ‘i’. Bug fixes in addition->quaternion function
2/7/2014	**1.0**	David	Updated printing of output for quaternion calculator
2/10/2014	**1.0**	Alexa	Minor fix to printing function
2/13/2014	**1.0**	David	Added = operator
2/17/2014	**1.0**	David	Added .gitignore to project and made quaternion-magnitudes output compatible with other functions input
2/21/2014	**1.0**	David	Fixed decimal point and rational syntax parsing
2/22/2014	**1.0**	Alexa	Fixed unexpected argument crashes
2/24/2014	**1.0**	Alexa & David	Fixed leading zero problem and the issue with parse errors being raised on valid input. Fixed output format. Rename quaternion.rkt to operations.rkt. Update and rename web-server.rkt to Quaternion.rkt
2/25/2014	**1.0**	Alexa	Fixed bugs with quaternion-mag
2/27/2014	**1.0**	David	Fixed quaternion-sin
3/2/2014	**1.0**	Alexa & David	Added tutorial to program web page. Fixed quaternion-div. Made (+) and (*) work.

[Table of Contents]

2 Overview

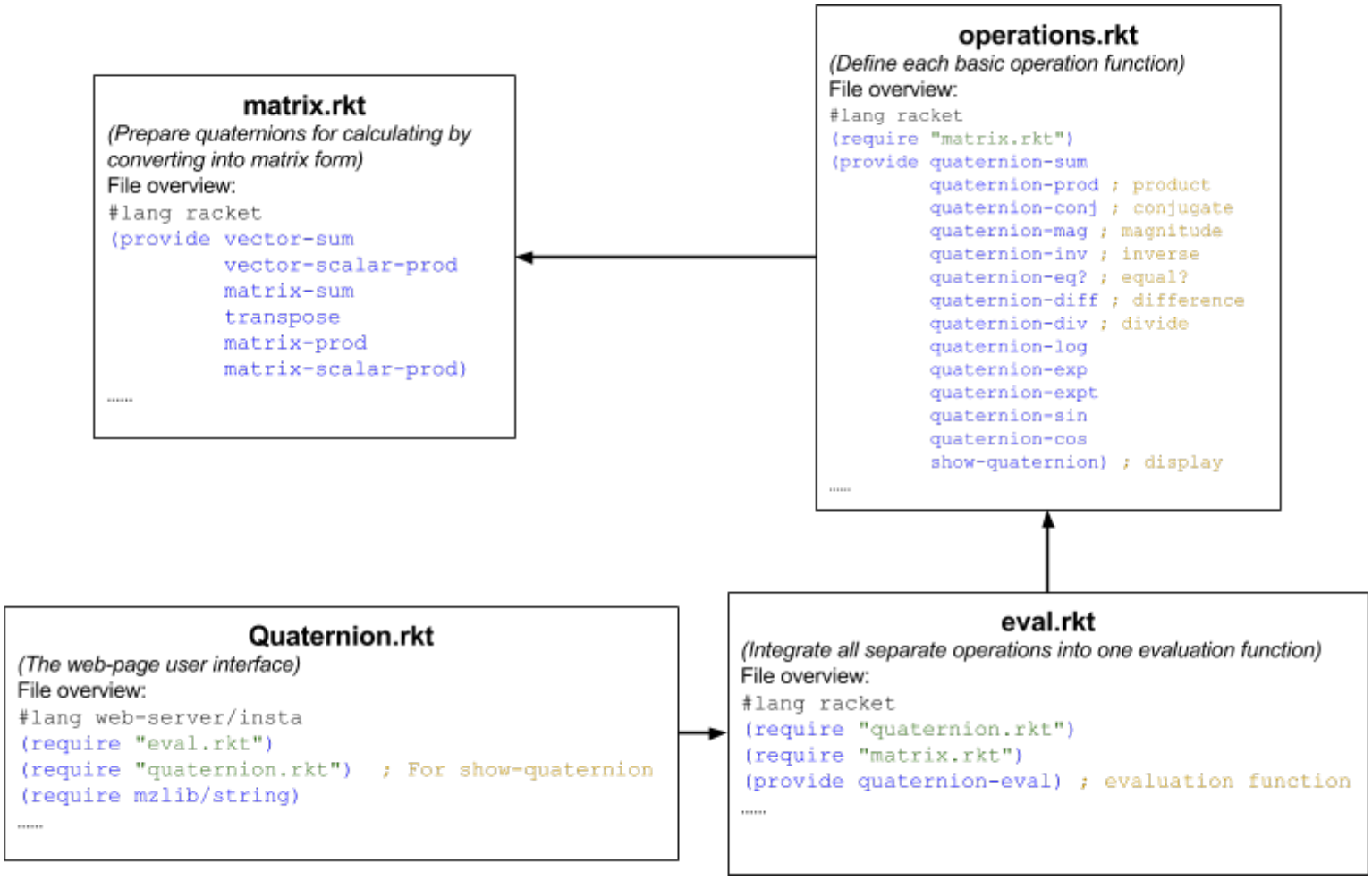
A quaternion calculator is a program designed to handle the four variable expansion of the complex number system. In day to day applications, this will most often translate into a vector with a rotation angle acting as the four variables. Because of this added storage, you can make the complex calculations associated with these vector matrices expediently and also avoid Gimbal Lock. This is when an object must rotate in space through all its axes and upon passing a lower level rotation axis -- most computers calculate x then y then z in that order -- will lock the rotation to the higher level axis allowing it only to move in that same direction on the locked planes. Preventing this is important in 3D graphics when you want an asset to render the proper movement, and essential in real world applications where a gimbal lock would affect a pilot's ability to fly a plane that uses computer driven, fly-by-wire flight controls. Because this system has the ability to apply a quaternion for position in any three dimensional space, it can be used to augment virtually any product that may require spatial awareness.

2.1 Service Description

The quaternion calculator is a powerful mathematical tool that excels in manipulating four dimensional data sets. The calculator can perform basic mathematical operations such as addition, subtraction, multiplication and division, while also being capable of performing trigonometric and exponential operations. In order to accommodate this powerful functionality, the interface to this calculator is a web page that enables the user to perform a variety of operations on quaternions. This interface includes an input field and an output field. in the input field, the quaternions with their associated operators should be entered in Scheme Standard Prefix notation. The result will be then be displayed in the output field just below the input field. All previous calculations and their results will be displayed in chronological order under the input field so that you can easily track a series of quaternion manipulations.

[Table of Contents]

2.2 System Structure



2.3 Technology Methods

The quaternion calculator employs several useful programming techniques that make the code both simply and faster.

- Functional abstraction
    - Functional abstraction allows the processes behind a certain operation and the actual operation itself to be hidden from each other. This in turn makes the code more readable and more modular.
- Using Lists to Represent Quaternions

    - Lists were used to represent the quaternions. Since lists have very powerful built in functionality, this made the code very simple and allowed more energy to be focused in other areas.
- Modules
    - The program was divided into four different modules. This made debugging and testing of each individual subroutine very straightforward.

2.4 Methodology

Methodologies used in this project first thought in a top-down perspective. Programmers initially viewed program based off how the user would interact with the program: how they would input their equations to be evaluated and how the calculations would be outputted. Once the basic design for the page and interface was mapped out, they moved onto the next level down: the actual calculations of the equations given. This dealt mostly with which functions and mathematical operations were needed to create a fully functioning calculator and how to create these to apply to quaternions. Then lastly at the base level how they were going to represent these quaternions and how they were to interact with the calculation functions.

In the actual implementation of the program, started from the bottom level and worked up towards to top level. First was the creation of the quaternion model by having represented as a matrix. From there they moved up to creating calculation functions (addition, multiplication, subtraction, division, etc…) that worked directly with the quaternion now in a matrix form. After creating all the basic mathematical functions. The programmers began to finish up the program with the design of the web page and user interface: giving parameters for input of equations to be evaluated, the output of the calculations, and overall design of the page.

In short, our programmers used a hybrid of top-down perspective when it came to brainstorming and planning out what was needed for the program to be successful, but a bottom-up technique for the actual implementation of the program.

2.5 Development Tools

Throughout the development of the quaternion calculator, three tools were used to address the following three aspects of the process: construction, management and sharing.

- **Construction/Programming:**
    - DrRacket IDE was used for the construction of the calculator.
- **Management/Documentation:**
    - Google Documents was used as a tool for documentation, managing development activity, progress and deadlines.
- **Sharing/Storage:**

- GitHub was used so that all team members could view and modify the current version of the calculator.

2.6 Source and Reference

- Särkkä,Simo (June 2007). ***Notes on Quaternions***. Available at http://www.lce.hut.fi/~ssarkka/pub/quat.pdf
  - This note concludes the basic properties of quaternions in many operations.
  - Function definitions of operation log, exp and expt are inspired from this note according to programmer Varady, A.

- Sperber, Michael (December 2012). ***Revised^6 Report on the Algorithmic Language Scheme (****R6RS).* Available at http://www.r6rs.org/final/html/r6rs/r6rs.html
  - This report defines the standard concepts for all kinds of Scheme language (e.g. Racket, Lisp) in many dimensions.
  - The standard Scheme representations for expressions and numbers can be referred in Chapter 1 of this report.

- ***The Racket Reference*** *(v 5.9.3)*. Available at http://docs.racket-lang.org/reference/index.html
  - Most operation functions in the quaternion computing system are defined similar to the same functions in this reference.

- Abelson, Harold and Sussman, Gerald J. (1996). ***Structure and Interpretation of Computer Programs*** *(Second Edition)*
  - As one of the textbook used in class, the idea of function abstraction and modularity helps build the structure of the quaternion computing system.

2.7 User Documentation

A link to the user documentation.
https://docs.google.com/document/d/1O7DqCK9YbOLPxi0hYSvr3LuEVzclaUTBe6jzWTpR8fc/edit?usp=sharing

Table of contents:

2.8 System Documentation

Files

- ❑ matrix.rkt
- ❑ operations.rkt
- ❑ eval.rkt
- ❑ Quaternion.rkt

2.8.1 Matrix

Functions

- **vector-sum**
  - **Prototype:** `(vector-sum)`

- **Parameter:** list(s)
- **Returns:** list
- **Description**: Adds up vectors to each one of the parts in the vector (i, j, k)

- **vector-scalar-prod**
  - **Prototype:** `(vector-scalar-prod a v)`
  - **Parameter:** a (number), v (list)
  - **Returns:** list
  - **Description:** Takes in a scalar value (a) and then multiplies it across the vector (v) and creates a new vector

- **matrix-sum**
  - **Prototype:** `(matrix-sum)`
  - **Parameter:** list(s) of lists
  - **Returns:** list(s) of lists
  - **Description:** Builds on vector sum, it adds matrices together, it does this by building on doing what vector-sum does. Going row by row, adding the corresponding rows together to create the new matrix.

- **transpose**
  - **Prototype:** `(transpose m)`
  - **Parameter:** m (list of lists)
  - **Returns:** list of lists
  - **Description:** Checks to see if the matrix if null or is greater than 1 row, else it returns an empty list otherwise, takes the list of lists, takes the first element of each row, creates a list for it then takes the remaining items in the lists, passes it into transpose until it has gone through the whole matrix transposing it

- **matrix-prod**
  - **Prototype:** `(matrix-prod orig-a orig-b)`
  - **Parameter:** orig-a (list of lists), orig-b (list of lists)
  - **Returns:** list of lists
  - **Description:** The matrix-prod takes in the two matrices that the user inputs (lists of lists) it then transposes the second one (orig-b) and then passes orig-a and orig-b into go. Go then checks to see if the matrices are not empty, it then multiplies the matrices across each other and creates a new matrix.

- **matrix-scalar-prod**
  - **Prototype:** `(matrix-scalar-prod a m)`
  - **Parameter:** a (number), m (list of lists)
  - **Returns:** list of lists

- **Description:** It takes in a scalar (a) and a matrix (list of lists) (m). It first applies 'a' across the matrices by multiplying 'a' by each item in the rows (lists), it then maps and creates a new matrix with 'a' multiplied across all of its values

2.8.2 Operations (megan)

Functions

- **quaternion->matrix**
  - **Prototype:** `(quaternion->matrix q)`
  - **Parameter:** q (list)
  - **Returns:** list of lists
  - **Description:** Convert a quaternion number/vector (list) to matrix form (list of lists)

- **matrix->quaternion**
  - **Prototype:** `(matrix->quaternion m)`
  - **Parameter:** m (list of lists)
  - **Returns:** list
  - **Description:** Convert a quaternion number's matrix representation (list of lists) to quaternion form (list of four numbers)

- **quaternion-op**
  - **Prototype:** `(quaternion-op f . qs)`
  - **Parameter:** f (function name), qs (list)
  - **Returns:** list
  - **Description:** Allows for matrix operations to be used directly on quaternions. qs is a list of quaternions that are converted into their equivalent matrices, the function f is then applied to these matrices and finally the matrices are converted back into lists.

- **quaternion-sum**
  - **Prototype:** `(quaternion-sum)`
  - **Parameter:** list(s) of four numbers
  - **Returns:** list
  - **Description:** Adds up the quaternions, by building on the same implementation to add matrices together used in matrix.rkt.

- **quaternion-prod**
  - **Prototype:** `(quaternion-prod qa qb)`
  - **Parameter:** qa (list), qb (list)
  - **Returns:** list
  - **Description:** Takes two quaternion-matrices, and multiplies them together to get the product of two quaternion numbers

- **quaternion-conj**
  - **Prototype:** `(quaternion-conj q)`
  - **Parameter:** q (list)
  - **Returns:** list
  - **Description:** Calculates the conjugate of a quaternion. If q is a unit quaternion, the conjugate is equal to the inverse

- **real**
  - **Prototype:** `(real Q))`
  - **Parameter:** Q (list)
  - **Returns:** number
  - **Description:** Returns the real number of the quaternion.

- **imaginary**
  - **Prototype:** `(imaginary Q)`
  - **Parameter:** Q (list)
  - **Returns:** number
  - **Description:** Returns the imaginary numbers of the quaternion.

- **i-coeff**
  - **Prototype:** `(i-coeff Q)`
  - **Parameter:** Q (list)
  - **Returns:** number
  - **Description:** Returns the i-coefficient number of the quaternion.

- **j-coeff**
  - **Prototype:** `(j-coeff Q)`
  - **Parameter:** Q (list)
  - **Returns:** number
  - **Description:** Returns the j-coefficient number of the quaternion.

- **k-coeff**
  - **Prototype:** `(k-coeff Q)`
  - **Parameter:** Q (list)
  - **Returns:** number
  - **Description:** Returns the k-coefficient number of the quaternion.

- **norm**
  - **Prototype:** `(norm V)`
  - **Parameter:** V(list)
  - **Returns:** number
  - **Description:** Returns the norm of a general vector

- **quaternion-mag**
  - **Prototype:** `(quaternion-mag Q)`
  - **Parameter:** Q (list)
  - **Returns:** number
  - **Description:** Returns the magnitude of a quaternion

- **quaternion-inv**
  - **Prototype:** `(quaternion-inv Q)`
  - **Parameter:** Q (list)
  - **Returns:** list
  - **Description:** Calculates the inverse of a quaternion, takes in a quaternion in vector form

- **quaternion-eq?**
  - **Prototype:** `(quaternion-eq? Q Rs)`
  - **Parameter:** Q (list), Rs (list)
  - **Returns:** boolean
  - **Description:** Checks to see if two quaternions are equal, and outputs whether that it is true or false

- **multiplyByC**
  - **Prototype:** `(multiplyByC C V)`
  - **Parameter:** C (number), V (list)
  - **Returns:** list
  - **Description:** Multiplies a general vector (V) by a constant C

- **quaternion-diff**
  - **Prototype:** `(quaternion-diff Q . Qs)`
  - **Parameter:** Q (list), Qs (list)
  - **Returns:** list
  - **Description:** Subtracts Qs from Q.

- **quaternion-div**
  - **Prototype:** `(quaternion-div Q . Qs)`
  - **Parameter:** Q (list), Qs (list)
  - **Returns:** list
  - **Description:** Divides first Q by all other Qs, consistent with Scheme standard. Takes a quaternion in vector form. To divide, it multiplies by inverse

- **quaternion-log**
  - **Prototype:** `(quaternion-log Q)`
  - **Parameter:** Q (list)
  - **Returns:** list

- **Description:** Gives the logarithm of a quaternion. Takes a quaternion in vector form. *Note that log is our "ln" - the natural log

- **quaternion-exp**
  - **Prototype:** `(quaternion-exp Q)`
  - **Parameter:** Q (list)
  - **Returns:** list
  - **Description:** Gives the exponential of a quaternion. Takes a quaternion in vector form. Note that this is e^quaternion.

- **quaternion-expt**
  - **Prototype:** `(quaternion-expt Q P)`
  - **Parameter:** Q (list or number), P (list or number)
  - **Returns:** list
  - **Description:** Gives Q^P, where at least one is a quaternion. Can take quaternion^#, #^quaternion, or quaternion^quaternion. Takes quaternions in vector form.

- **quaternion-cos**
  - **Prototype:** `(quaternion-cos Q)`
  - **Parameter:** Q (list)
  - **Returns:** list
  - **Description:** Takes the cosine of a quaternion

- **quaternion-sin**
  - **Prototype:** `(quaternion-sin Q)`
  - **Parameter:** Q (list)
  - **Returns:** list
  - **Description:** Takes the sine of a quaternion

- **show-quaternion**
  - **Prototype:** `(show-quaternion q)`
  - **Parameter:** Q (list)
  - **Returns:** string
  - **Description:** Restores the quaternion from an ordered list of numbers to a string of numbers with the corresponding real, I, J, and K elements in place.

2.8.3 Eval

Functions

- **string->symOrNum**
  - **Prototype:** `(string->symOrNum str)`
  - **Parameter:** str (string)

  - **Returns:** number or symbol
  - **Description:** Evaluates whether or not the given string is a number. If it is a number, it returns the number. If it is a concatenation of numbers and other elements, it returns a symbol, denoted in racket by an apostrophe followed by two vertical bars that enclose the symbol: '|`symbol`|

- **last**
  - **Prototype:** `(last L)`
  - **Parameter:** L (number or string)
  - **Returns:** number
  - **Description:** Given a single number of a string, this function returns the number of elements in the string or the number of digits in the number.

- **penultimate**
  - **Prototype:** `(penultimate L)`
  - **Parameter:** L (number or string)
  - **Returns:** number
  - **Description:** Given a single number of a string, this function returns one less than the number of elements in the string or the number of digits in the number.

- **coeff**
  - **Prototype:** `(coeff L)`
  - **Parameter:** L (string)
  - **Returns:** number
  - **Description:** Extracts the numerical coefficient preceding a single character.

- **first**
  - **Prototype:** `(first L)`
  - **Parameter:** L (list of at least 2 elements)
  - **Returns:** Second element of a list
  - **Description:** Extracts the second element of a list. In terms of a quaternion operation, this would return the first operand.

- **second**
  - **Prototype:** `(second L)`
  - **Parameter:** L (list of at least 3 elements)
  - **Returns:** Third element of a list
  - **Description:** Extracts the third element of a list. In terms of a quaternion operation, this would return the second operand.

- **restOf**
  - **Prototype:** `(restOf L)`
  - **Parameter:** L (list of at least 3 elements)
  - **Returns:** L, excluding the first two elements

- **Description:** Removes the first two items from the list and returns the shortened list.

- **arguments**
  - **Prototype:** `(arguments E)`
  - **Parameter:** E (list)
  - **Returns:** L, excluding the first element
  - **Description:** Removes the first element from the list and returns the shortened list. In terms of quaternion operations, this would return the arguments following an operator.

- **addition->quaternion**
  - **Prototype:** `(addition->quaternion L-orig)`
  - **Parameter:** L-orig (list)
  - **Returns:** list
  - **Description:** Accepts a series of quaternion addition operators and numerical operands with their associated i, j or k elements, in standard Scheme prefix notation. It performs the operations described by the prefix input and returns a list of four elements containing the real, i, j and k coefficients.

- **quaternion-eval**
  - **Prototype:** `(quaternion-eval E)`
  - **Parameter:** E (list)
  - **Returns:** list
  - **Description:** Accepts a series of quaternion operators and numerical operands with their associated i, j or k elements, in standard Scheme prefix notation. It performs the operations described by the prefix input and returns a list of four elements containing the real, i, j and k coefficients.

2.8.4 Quaternion (web-server)

Structures

- **blog**
  - **Prototype:** `(struct blog (posts) #:mutable)`
- **post**
  - **Prototype:** `(struct post (expression))`

Functions

- **BLOG**
  - **Prototype:** `(BLOG (blog '())`
  - **Parameter:** N/A
  - **Returns:** N/A
  - **Description:** Instantiates an instance of the struct “blog” and creates a collection of “blog” posts.

- **start**
  - **Prototype:** `(start request)`
  - **Parameter:** request
  - **Returns:** N/A
  - **Description:** Initiates the rendering of the web page by calling the render-page function.

- **parse-post**
  - **Prototype:** `(parse-post bindings)`
  - **Parameter:** bindings
  - **Returns:** struct
  - **Description:** Extracts information from the form and puts it in a post structure. This simply takes the expression but does not evaluate it.

- **render-page**
  - **Prototype:** `(render-page request)`
  - **Parameter:** request
  - **Returns:** N/A
  - **Description:** Renders the web page and handles input from the user.

- **blog-insert-post!**
  - **Prototype:** `(blog-insert-post! a-blog a-post)`
  - **Parameter:** a-blog, a-post
  - **Returns:** a-blog with a-post added to the beginning
  - **Description:** Adds a post to an existing blog.

- **render-post**
  - **Prototype:** `(render-post a-post)`
  - **Parameter:** a-post
  - **Returns:** HTML-formatted, evaluated post
  - **Description:** Retrieves the information from a post structure and then creates a single post. This is an HTML-formatted post which calls the eval function and handles the user input and output.

- **render-posts**
  - **Prototype:** `(render-posts)`
  - **Parameter:** N/A
  - **Returns:** HTML-formatted posts
  - **Description:** This contains the CSS and renders all of the posts.

[Table of Contents]

3 Related Projects or Major Support Work

The following table lists previous projects or major pieces of support work that have been carried out on this service:

Project / Major Support Work	Developer	Key words Related	Description
Nth Symbolic and Numeric Differentiation	*Varady, Alexa*	• *Racket* • *Web server implementation* • *Prefix expression* • *Function abstraction*	*A Simple web based program to find Nth derivative of a function. This program is also developed using DrRacket IDE and follows the Scheme standard on all input and output expressions.*

4 FAQ

- *How to access the application?*
  - This app is free and open-source. You can download the latest version of source code files at https://github.com/roboguy13/quaternions to your own computer and run it.

- *Is there anything else I need to have installed for this to run and if so, where can the installation instructions be found?*
  - You do need to install the DrRacket IDE which is our development environment. Open the *Quaternion.rkt* in DrRacket and click “Run” button on the upper right side of the window. The evaluator web page will be opened in your web browser.
  - You can download DrRacket at http://racket-lang.org/download/. We recommend version 5.3.x for best results.

- *Are there any 'features' of which I need to be aware?*
  - Since the format of expressions can be different from mathematical traditions, Section 1.3. *Format of expressions* in the user tutorial is recommended for any users who are not familiar with the Scheme standard representation of numbers and expressions.

- *How this application was tested?*
    - We use peer testing method inside the develop team. If anyone find any bugs, we will report them to programmers and they will fix it in the next release version.

- *What can I do when I encounter a Servlet Error?*
    - The Servlet Errors are usually caused by improper inputs. To reset the evaluator web page, go back to your DrRacket window and click “Stop” button then click “Run” again to start a new session.
    - If you get the message “Sorry, this page has expired. Please go back.” from the page, do the same thing as above. Don’t try to use the navigation key “back” on your browser to fix the problem.
    - Examples in the user guide clearly distinguish accepted inputs from wrong inputs.

5 Troubleshooting Guide

- **Web Browser Compatibility**
    - The web server has been tested with Internet Explorer, FireFox, and Chrome. All of these work. IE will have some small graphical bugs/layout issues, but the evaluator itself will work. It is recommended to not use IE with the calculator for the best results. For best results use Chrome.

- **Web Server Timeout Error**
    - If the web server is left unattended for 2 or more minutes, it will time out. If in constant use though, it will not timeout. If it times out, the racket program needs to be stopped and restarted.

- **Invalid Input Error**
    - If an invalid input is put in, the web server will go to an error page. If this happens, the racket program needs to be stopped and restarted. Backing out--pushing the back button on a browser--and retrying will not work.

- **Error Message Format**
    - Error messages will be displayed by the web server, the timeout error will be simply understood. The invalid input error is the error screen with the purple boxes. It will signify that the equation has been inputted incorrectly.

- **Invalid Input Bugs**

    - The only errors that can be predicted are with certain invalid inputs that go through. These bugs are currently being worked on by the programmers, but at this moment remain unresolved. As long as valid input is used, there should be no issues. A current example of the evaluating invalid input problem is (+ 3i+4j 3k+4j*k). It will evaluate the equation, though it should not because it is not being inputted with the right syntax. An error that occurs is that the parser seems to read the * as a +, which causes calculation errors.

- **Restart Program After Errors**
    - If invalid input is used, the only way to recover from the error is to restart the web server. Also after web server timeout the only way to recover is to restart.

- **How to avoid Invalid Input/Evaluation Errors**
    - If an evaluation fails, the web server must be restarted. If an evaluation fails due to invalid input, depending on the input it can be fixed by using standard prefix notation. (EX: You want to evaluate (3ik + 4j) + (5k + 9ji). You input (+ 3ik+4j 5k+9ji), it will not work.) To get around this, you can use prefix notation and put in (+ (* 3i k)+4j 5k+(* 9j i)) and this will give the correct answer.

6 Document Sign Off

Please add other sign off roles where required:

Project Leader	*Scott Tilson*	*3/4/2014*
Project Analyst	*Josh Werner*	*3/4/2014*
Billing	*Brice Thrower*	*3/4/2014*
Business Analyst	*Murl Westheffer*	*3/4/2014*
Programmer	*Alexa Varady*	*3/4/2014*
Programmer	*David Young*	*3/4/2014*
Documentation	*Megan Teahan*	*3/4/2014*
Documentation	*Patrick Walter*	*3/4/2014*

Documentation	*Su Yan*	*3/4/2014*
Support/Testing	*Sean Stout*	*3/4/2014*