Quaternion Calculator

User Manual

Document Version:1.0

Date: 3/4/2014

Table of Contents:

Chapter 1 Introduction & Backgrounds

1.1. What it can do

Built with the programming language Racket, this program is expected to have following features:

- Accept any computation tasks involving real, complex or quaternion numbers with either fixnum or flonum representations

- Can do many kinds of common operations: +, -, *, /, exp, expt, log, sin, cos, =, magnitude

1.2. The number system & syntax

- Please note that the general term “number” or “number?” appears anywhere in this document refer to all kinds of numbers stated below.

1.2.1. Real number

- As the most common and relatively simple number system, integers, fractions and decimals are considered as real number.
  - For example:
    - 56
    - 56.3
    - 7/8

- Any decimals used as numerators or denominators are not accepted as fractions
  - For example:
    - 56.3/78 is not accepted. Use (/ 56.3 78) instead.

- Same as the mathematical notation, the integer part of a decimal number can be omitted if it is 0.
  - Example:
    - .14 is equivalent to 0.14

1.2.2. Complex number

- A complex has a real part and an imaginary part. Generally it is in the form a+bi. Note that a and b are real numbers.
  - The following input are valid:
    - 3.2-6/11i
    - 0+0.8i

- b can only be a number, not an expression.
  - For example:
    - (/ 58 6)+7/5i is valid.
    - (cos 5)i is not valid. It should be written as (* (cos 5) i).

1.2.3. Quaternion number

- The quaternions are a number system that extends the complex numbers. Its general form is a + bi + cj + dk. Note that a, b, c and d are real numbers.
  - For example:
    - 2i+3k
    - 5.4-6.2j+6/7k.

- Similarly with complex numbers, i, j and k can only be preceded by a number, not an expression.
  - For example:
    - (* 3 4)k+(/ 5.6 7)j ⇒ 12, Only (* 3 4) has been evaluated!
    - Instead, using (+ (* (* 3 4) k) (* (/ 5.6 7) j)) to get correct result.

1.2.4. Scientific notation

- Scientific notation is used when a number is too small or too large.
  - Examples:
    - (* 271979577247970257395 0.6180339887) ⇒ 1.6809262297150285e+020
    - (sin 1e-027) ⇒ 9.999999999999999e-028

1.3. Format of expressions

1.3.1. Infix or prefix?

- Numbers are specified using infix + and -, including complex and quaternion numbers.
- There cannot be any spaces in a number.
- All operations must be written as prefix.
- For example, 1+5k and 3/2i+5.8j are valid; each of those are considered to be an expression of one quaternion. But 7+(sin 9)j is not valid. This should be written as (+ 7 (* (sin 9) j)).

1.3.2. Format of input

- Since this computation system is based on Racket, the form of expressions follows the definition in R6RS (Revised^6 Report on the Algorithmic Language Scheme), which is different from normal numerical expressions in mathematics.

- The most fundamental expressions are literal expressions
    - For example, True/False: #t / #f; any single number: 23, 5/8, i + j + k.
    - No parentheses needed when input literal expressions. For example:
        - 23 ⇒ 23
        - (23) ⇒ Error!
    - This program only accept numeric literal expressions as inputs. Boolean values cannot be accepted.
        - #t ⇒ Error!
        - (#f) ⇒ Error!

- Compound expressions are formed by placing parentheses around their subexpressions. The first subexpression identifies an operation; the remaining subexpressions are operands to the operation.
    - Some example inputs:
        - (+ 14 (* 23 42)) ⇒ 980
        - (* 2i+k   4+j+8k) ⇒ -8+7i-16j+6k
        - (exp (log 3)) ⇒ 3.0000000000000004
        - (sin 3.1415926535897936) ⇒ -0.1369691314080463

1.3.3. Format of output

- The form of a result should be different based on the input. Specifically, output is one of the number types defined in section 1.2 or a boolean value for the equal operation except some extreme case.
    - Some example outputs:
        - Quaternion number: -8.35+7i-17.4563j+6k
        - Complex number: 2-3/5i
        - Integer: 23
        - (= 3 4) ⇒ #f

- In some extreme cases, when an expression exceeds the calculation capacity of the program, usually some inaccurate result or an "infinity" symbol will be given.
    - For example:
        - (exp 99999999999999999) ⇒ +inf.0 (positive infinity)
        - (expt -999999999999 -9) ⇒ 0 (inaccurate result)
        - (expt -999999999 9999999) ⇒ -inf.0+inf.0i (negative infinity)
    - Developers are not responsible for the accuracy of results in extreme cases since it is limited by the developing environment.

Chapter 2 Operations

2.1. Addition (+)

- `(+ z ...) → number?`
  - z : number?
  - Returns the sum of the z's, adding pairwise from left to right. If no arguments are provided, the result will be 0.
  - Examples:
    - `(+ 1 2) ⇒ 3`
    - `(+ 1.0 2+3i 5) ⇒ 8.0+3.0i`
    - `(+) ⇒ 0`

2.2. Subtraction (-)

- `(- z) → number?`
  - z : number?
- `(- z w ...) → number?`
  - z : number?
  - w : number?
  - When no w's are supplied, returns (- 0 z). Otherwise, returns the subtraction of the w's from z working pairwise from left to right.
  - Examples:
    - `(- 5 3.0) ⇒ 2.0`
    - `(- 1) ⇒ -1`
    - `(- 2+7i 1 3) ⇒ -2+7i`

2.3. Multiplication (*)

- `(* z ...) → number?`
  - z : number?
  - Returns the product of the z's, multiplying pairwise from left to right. If no arguments are provided, the result is 1. Multiplying any number by exact 0 produces exact 0.
  - Examples:
    - `(* 2 3) ⇒ 6`
    - `(* 8.0 9) ⇒ 72.0`
    - `(* 1+2i 3+4i) ⇒ -5+10i`
    - `(*) ⇒ 1`

2.4. Division (/)

- `(/ z) → number?`
  - z : number?

- `(/ z w ...) → number?`
  - z : number?
  - w : number?
  - When no w’s are supplied, returns (/ 1 z). Otherwise, returns the division of z by the w’s working pairwise from left to right.
    If z is exact 0 and no w is exact 0, then the result is exact 0. If any w is exact 0, the divide-by-zero exception is raised.
  - Examples:
    - `(/ 3 4) ⇒ 3/4`
    - `(/ 81 3 3) ⇒ 9`
    - `(/ 10.0) ⇒ 0.1`
    - `(/ 1+2i 3+4i) ⇒ 11/25+2/25i`

2.5. Exponential function (expt)

- `(expt z w) → number?`
  - z : number?
  - w : number?
  - Returns z raised to the power of w.
    If w is exact 0, the result is exact 1. If w is 0.0 or -0.0 and z is a real number, the result is 1.0 (even if z is +nan.0).
    If z is exact 1, the result is exact 1. If z is 1.0 and w is a real number, the result is 1.0 (even if w is +nan.0).
    If z is exact 0 and w is negative, the divide-by-zero exception is raised.
  - Examples:
    - `(expt 2 3) ⇒ 8`
    - `(expt 4 0.5) ⇒ 2.0`
    - `(expt +inf.0 0) ⇒ 1`

2.6. Natural exponential function (exp)

- `(exp z) → number?`
  - z : number?
  - Returns Euler’s number raised to the power of z. The result is normally inexact, but it is exact 1 when z is an exact 0.
  - Examples:
    - `(exp 1) ⇒ 2.718281828459045`
    - `(exp 2+3i) ⇒`
      `-7.315110094901103+1.0427436562359045i`
    - `(exp 0) ⇒ 1`

2.7. Natural logarithmic function (log)

- `(log z) → number?`
    - z : number?
    - Returns the natural logarithm of z. The result is normally inexact, but it is exact 0 when z is an exact 1. When z is exact 0, exn:fail:contract:divide-by-zero exception is raised.
    - Examples:
        - `(log (exp 1)) ⇒ 1.0`
        - `(log 2+3i) ⇒ 1.2824746787307684+0.982793723247329i`
        - `(log 1) ⇒ 0`

2.8. Trigonometric functions (sin, cos)

- `(sin z) → number?`
    - z : number?
    - Returns the sine of z, where z is in radians. The result is normally inexact, but it is exact 0 if z is exact 0.
    - Examples:
        - `(sin 3.14159) ⇒ 2.65358979335273e-06`
        - `(sin 1.0+5.0i) ⇒ 62.44551846769653+40.0921657779984i`
- `(cos z)  → number?`
    - z : number?
    - Returns the cosine of z, where z is in radians. The result is normally inexact, but it is exact 0 if z is exact 0.
    - Examples:
        - `(cos 0) ⇒ 1.0`
        - `(cos 3+i) ⇒ -1.5276382501165433-0.1658444019189788i`

2.9. Equality (=)

- `(= z w ...+) → boolean?`
    - z : number?
    - w : number?
    - Returns #t if all of the arguments are numerically equal, #f otherwise. An inexact number is numerically equal to an exact number when the exact coercion of the inexact number is the exact number. Also, 0.0 and -0.0 are numerically equal, but +nan.0 is not numerically equal to itself.
    - Examples:
        - `(= 1 1.0) ⇒ #t`

    - `(= 1 2) ⇒ #f`
    - `(= 2+3i 2+3i 2+3i) ⇒ #t`

2.10. Magnitude

- `(magnitude z) → number?`
  - z: number?
  - z is
    - real number: Returns the magnitude of the real number z.
    - complex number: Returns the magnitude of the complex number `z` in polar coordinates.
    - quaternion number: works the same as complex number.
  - Examples:
    - `(magnitude -5) ⇒ 5`
    - `(magnitude 3+4i)  ⇒ 5`
    - `(magnitude 1+i+j+k) ⇒ 2`

2.11. Square root function (sqrt)

- `(sqrt z)→ number?`
  - z: number?
  - Returns the square root of number z.
  - Examples:
    - `(sqrt 2) ⇒ 1.414213562373095`
    - `(sqrt 3+2i) ⇒ 1.8173540210239707+0.5502505227003375i`
    - `(sqrt i+j+k) ⇒ 0.9306048591020994+0.537284965911771i+0.537284965911771j+0.537284965911771k`

Chapter 3 User Interface

3.1. Main web interface

- Consult the FAQ section in System Manual about how to enter the web interface.

- The link to documentations of this evaluator can be found in the web page interface.

3.2. Common error types & error information

- Unexpected Arguments
    - This type of error is displayed as outputs of your incorrect inputs. Nothing needed to fix this error, just type a new expression in the evaluator.

    - This type of error happens but not limit to following cases:
        - Input parameter type mismatch.
        - Input expression not closed by parentheses.

- Contract violation
    - An uninteractable exception page titled “Servlet Error” will appear. You need to go back to your DrRacket window and click “Stop” button then click “Run” again to start a new session.

    - This type of error is caused by parameter violation(s) of an internal function.

    - This type of error happens but not limit to following cases:
        - Input is a unrecognized function closed with parentheses.
        - Missing one or more arguments for the input function.
        - Input is not a prefix expression.

    - Technical exception message will be shown in this format:

```
<function_name>: contract violation
expected: <expected_value_type>
given: <input_value_type / empty>
```

- Division by zero
    - “Servlet Error” page appears as stated above.

    - This type of error is caused by using 0 as a divisor.

    - Technical exception message will be shown in this format:

```
        read: division by zero: <expression>
```

- Page expired
    - Each web page session will expire after left unattended for 2 or more minutes.

    - The error message is the following:
        - "Sorry, this page has expired. Please go back."

    - To fix this error, go back to your DrRacket window and click "Stop" button then click "Run" again to start a new session.