Hypercomplex GCD in APL

Bob Smith
Sudley Place Software
Originally Written
15 Apr 2016
Updated
11 Apr 2018

Introduction

Euclidean Division[1] (Greatest Common Divisor or GCD) is a simple algorithm in the normal domain of Real numbers, but when we venture into Hypercomplex numbers the picture is different.

In this paper, I examine two ways to define the APL primitive functions Floor, Residue, and GCD on Hypercomplex numbers along with their consequences. I consider two definitions of the Floor function, one by E. E. McDonnell[3] (1973) and one by A. Hurwitz[5] (1888). The Floor function is used in the definition of the Residue function and, in turn, that function is used in the definition of the GCD function.

We also discuss the Principal Values for GCD and provide a definition.

Prerequisite Reading

The paper “Hypercomplex Notation in APL”[9] provides a summary of the notation used in this paper for Hypercomplex numbers, and the paper “Hypercomplex Numbers in APL”[10] provides an overview of Hypercomplex numbers in general.

GCD

The standard GCD function is defined as follows as an operator so as

to pass the appropriate Residue function as the left operand:

```
Z←L (LO GCD) R;T ⎕CT
⍝ L∨R for Hypercomplex numbers
⍝   using LO as the Residue function
⍝ Scalar L and R
⎕CT←1E¯10   ⍝ Needed for FP numbers only
:repeat     ⍝ Euclidean Algorithm
  T←L
  L←L LO R
  :Assert (|L)<|T
  R←T
:until (|L)≤SCT ⍝ FP numbers only
```

Note the `:Assert` statement in the midst of the loop. It is present to catch invalid Residue function results which leads this iterative function not to terminate because the magnitude of `L` must decrease at every iteration. The global value `SCT` holds the value of System Comparison Tolerance (e.g., `4E¯15`).

Residue

Next in line is the standard (recursive) residue function as used by the GCD function above, implemented as an operator so as to pass the appropriate Floor function as the left operand:

```
Z←L (LO Residue) R
⍝ L|R for Hypercomplex numbers
⍝   using LO as the Floor function
⍝ Scalar L and R
⍝ Sensitive to ⎕CT
→(L(=⌊0) 0 1)/L0 L1     ⍝ a.k.a. L=0 1 with ⎕CT←0
Z←1 ∇ R÷L ⋄ :Assert 1>|Z
Z←Z×L     ⋄ :Assert Z=LO Z ⋄ →0
L0:Z←R ⋄ →0
L1:Z←R-LO R
```

The notation =⍠0 uses the Variant operator to execute the Equal function with ⎕CT set to zero. The ∇ symbol invokes the derived function (⎕LO Residue) recursively. Note the first :Assert statement – it ensures that the magnitude of the Residue function's results are always less than the magnitude of the modulus (left argument) so that the GCD function above terminates.

Floor

While the Floor of a Real number is easy to define uniquely, there are several different ways to extend it to Hypercomplex numbers. One definition is by E. E. McDonnell in his paper “Complex Floor”[3], and another is by A. Hurwitz as described by D. Forkes in his paper “Complex Floor Revisited”[4].

Geometrically, on Complex numbers, these definitions appear as follows:

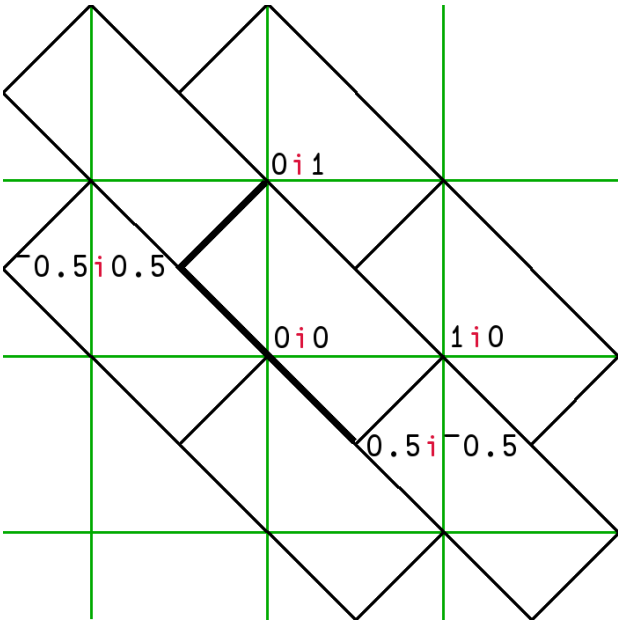

Figure 1: McDonnell's Floor

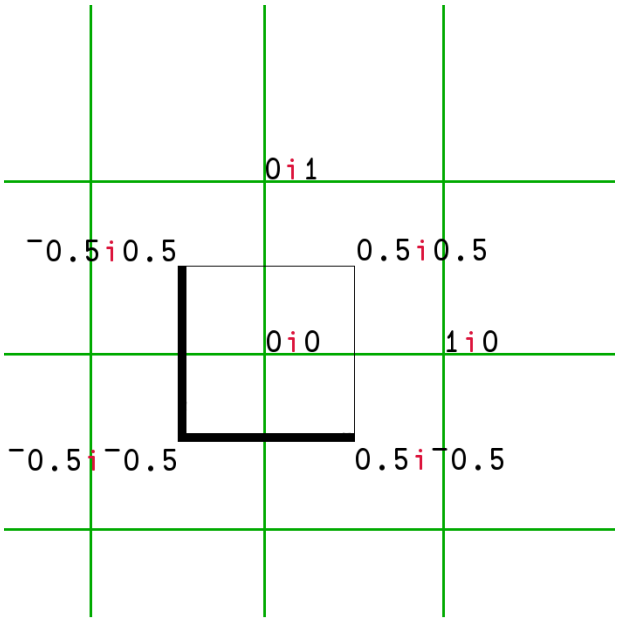

Figure 2: Hurwitz's Floor

These figures illustrate the set of points whose Complex Floor is zero (a.k.a. 0i0) for both McDonnell's (center slanted rectangle) and

Hurwitz's (center outlined square) Floor functions. Both have unit area. The APL functions that implement these definitions are as follows:

McDonnell's Floor

```
Z←SF R;Cof Flr Frc
⍝ ⌊R for Hypercomplex numbers
⍝   using McDonnell's Floor as extended by Smith
⍝ Scalar R
⍝ Sensitive to ⎕CT
Cof←↕R       ⍝ Coefficients
Flr←⌊Cof     ⍝ Lower left corner
Frc←1|Cof    ⍝ Fractional parts
:if 1 (≤⍤0) +/Frc ⍝ a.k.a. 1≤+/Frc with ⎕CT←0
  Flr[Frc⍳⌈/Frc]+←1
:end
Z←<Flr
```

Hurwitz's Floor

```
Z←HF R
⍝ ⌊R for Hypercomplex numbers
⍝   using Hurwitz's Floor
⍝ Scalar R
⍝ Sensitive to ⎕CT
Z←↕R ⋄ Z←<⌊Z+(1+⊃Z)÷2
```

Note the odd construction `Z+(1+⊃Z)÷2` the purpose of which is to add `1÷2` to Z while retaining `Z`'s datatype, especially important if `Z` is a multi-precision Rational/Integer.

Put it All Together

GCD is defined in terms of the Residue primitive function which in turn is defined in terms of the Floor function. The Floor function by E. E. McDonnell is the APL industry standard in that it has been adopted by

at least three APL vendors who have implemented Complex numbers. These two implementations of McDonnell's Floor function are very similar, but not identical. The Floor function by A. Hurwitz was brought to our attention by D. Forkes[4]. We need to examine these two definitions on Complex numbers to see how suitable they are to extend to Quaternions and Octonions. To do this, we need one more concept.

Fractionality

This property of a Floor function means that the “magnitude of the difference between a number and its floor shall be less than one”[3], which McDonnell calls “the fundamental property of the floor function”. In particular, with this property, the iterative GCD algorithm is assured to terminate and converge to the answer; without it, the algorithm need not.

Actually, Fractionality is a property of a Floor function **as applied to a particular type of number**. For example, all of the Floor functions under consideration have Fractionality on both Real and Complex numbers. However, neither Floor function has Fractionality on Quaternions or Octonions.

Why No Fractionality?

The short reason why is that as the dimension increases (in this case from 1 to 2 to 4 to 8) the diagonals (which are the longest distances from the origin to any corner in Figures 1 and 2 when extended to Quaternions) get longer and longer – when they meet or exceed one, we lose Fractionality. Concerning higher dimensions, “As J. H. Conway once explained to Martin Gardner, *There is a lot of room up there.*”[6].

Focusing on just the interior of the two unit areas in Figures 1 and 2 (a slanted rectangle for McDonnell's Floor and a square for Hurwitz's Floor), the Complex Floor of all of the interior points is zero (a.k.a.

`0i0`). As Fractionality requires that the magnitude of the difference between a point and its Floor be less than one, this translates to the magnitude of the point is less than one (because the Floor is zero).

For McDonnell's Floor on Complex numbers, the longest diagonal distance to a corner (either `1i0` or `0i1`) of the unit area is of length just under one, but not equal to one because the points at `1i0` or `0i1` are **not** included in the boundary of the rectangle around `0i0` (they are included in the boundary of adjacent rectangles).

For Hurwitz's Floor on Complex numbers, the longest diagonal distance to a corner (such as `0.5i0.5`) is of length `(√2)÷2`, (`=0.7071…`). For this reason, Hurwitz's Floor tends to converge more quickly than does McDonnell's when used in GCD.

For McDonnell's Floor on Quaternions, the four-dimensional unit space of all such points whose Floor is zero (a.k.a. `0i0j0k0`) has two corners (at `1i0j1k1` and `0i1j1k1`) whose diagonal distance from the origin is `√3`. As this value is greater than one, McDonnell's Floor on Quaternions does not have Fractionality and as such is not a suitable definition of Floor on Quaternions.

For Hurwitz's Floor on Quaternions, all (sixteen) corners of its unit 4-cube are equally farthest from the origin. Because this unit 4-cube tessellates the entire 4-space, half of the corners (and edges) are in the unit 4-cube and half are in one of the adjacent unit 4-cubes. Using one of the corners in this unit 4-cube, its diagonal distance from the origin is exactly one. As this value is not less than one, Hurwitz's Floor on Quaternions does not have Fractionality and as such is not a suitable definition of Floor on Quaternions.

Hurwitz Quaternions

As the literature suggests[2], if we extend Quaternion integers to include Quaternions whose individual components are all half integers

(that is, half of an odd integer), then Euclidean Division is possible – these values are called Hurwitz Quaternions[8].

For example, `1i2j3k4`, `1.5i2.5j3.5k4.5` are both Hurwitz Quaternions, but `1.5i2j3k4` is not because all or none of the coefficients must be half integers – no mixing.

As explained by Conway & Smith[13] and referring to Figure 2, the sixteen corners of the unit 4-cube centered on the origin are exactly the points where the distance from the origin is exactly `1`; all other points in or on the 4-cube are at distance less than `1` from the origin. Now, because those sixteen points are all Hurwitz Integers, we define their Floor to be themselves, and so for those sixteen points, the magnitude of the difference between the number and its Floor is zero.

This change requires a re-definition of Hurwitz's Floor function as follows:

```
Z←UF R;F T f
⍝ ⌊R for Hypercomplex numbers
⍝   using Hurwitz's Floor function
⍝   returning the nearest Hurwitz Quaternion
⍝ Scalar R
⍝ Sensitive to ⎕CT
F←>R  ⋄ f←{ω-(×ω)×0=2|ω}
Z←<  ⌊  F+(1+⍳F)÷2
T←<(f⌊2×F+(1+⍳F)÷4)÷2
:if (|R-Z)≥|R-T ⋄ Z←T ⋄ :end
```

where `Z` calculates the nearest point with all integer coordinates and `T` calculates the nearest Hurwitz Integer (point with all half integer coordinates). Whichever point is closer to the incoming value, we choose it. With this change, GCD using Hurwitz's Residue and Floor has Fractionality and is defined on Quaternions.

An APL implementation of this form of the Residue function for Quaternions using Hurwitz's Floor is as follows:

```
Z←L UR R
⍝ L|R for Hypercomplex numbers
⍝   using Hurwitz's Floor function
⍝ Scalar L and R
⍝ Sensitive to ⎕CT
→(L(=⌊0) 0 1)/L0 L1 ⍝ a.k.a. L=0 1 with ⎕CT←0
Z←L×1∇R÷L ◇ →0
L0:Z←R ◇ →0
L1:Z←R-UF R
```

Octonions

For Octonions, neither definition of Floor has Fractionality as the maximum diagonal distances in both cases are greater than one, and as such there is no suitable definition of Floor on Octonions.

GCD Principal Values

The principal value of the GCD function introduces additional complications. The GCD of (say) `117j44` and `¯63j¯16` may be any of `3j¯4 4j3 ¯3j4 ¯4j¯3` as they all divide both arguments. McDonnell[7] made a good suggestion that for Complex numbers, the principal value be in the first quadrant (`1j1`) or on the positive real axis (`1j0`) which means that `117j44∨¯63j¯16 ↔ 4j3`.

For Complex numbers, because there are only four quadrants, it's always possible to rotate a GCD result from any quadrant into the first quadrant by multiplying it by one of the four Complex integer units, i.e., `±1` and `±1j`.

However, for Quaternions, there are sixteen ($=2^4$) bi-quadrants and only eight Quaternion integer units, i.e., `±1`, `±1j`, `±1j`, and `±1k`. Because Quaternions are non-commutative, we can reach additional

bi-quadrants by multiplying the GCD result on **both** sides by an integer unit, but still the values in half of the bi-quadrants (e.g., `¯1i1j1k1`) cannot be rotated into the first bi-quadrant (`1i1j1k1`).

The solution I propose is to widen the definition of a principal value of GCD for Quaternions to include one nearby bi-quadrant `¯1i1j1k1`, in which case left and right multiplication by integer units can now rotate every GCD result into one of those two corners. The choice of the additional corner for Quaternions is arbitrary as it could be any corner with either one positive coefficient and the rest negative, or one negative coefficient and the rest positive.

Although it is trivial to extend this method to Octonions, there is no need as GCD is not defined on those numbers due to the lack of Fractionality.

This rotation is implemented by computing various static tables which are used by the following algorithm:

```
Z←rotateGCD R;⎕IO T
⍝ Rotate R into the first (all non-negative
⍝   coefficients) corner or second (real coord
⍝   negative, all others non-negative) corner
⍝ Scalar R
⎕IO←0
T←3⊥1+×⍵>R
:switch ≡R  ⍝ Split cases on the dimension of R
  :case 1 ⋄ Z←R
  :case 2 ⋄ Z←           R×u2[T  ⊃s2]
  :case 4 ⋄ Z←u4[T 0⊃s4]×R×u4[T 1⊃s4]
:end
```

where `u2←uf 2` and `u4←uf 4` are the 2- and 4-dimensional units computed from `uf←{<(⊢,-)∘.=⍨⍳⍵}`, and `s2` and `s4` are computed by a separate APL function that calculates the indices of the

appropriate units to use when multiplying on the right for `s2` or left and right for `s4` to rotate the number into the first (Complex) or first or second (Quaternion) corner.

The former (`s2`) is a 9-element vector of indices

```
1 2 2 1 0 0 3 3 0
```

and the latter (`s4`) is an 81-element vector of pairs of indices, one to multiply by the appropriate unit on the left and one for the right, the first few values of which are

```
(0 1) (2 6) (0 1) (0 1) (0 4)…
```

Combining this with the previous `UG1` function yields

```
Z←L UG R;T ⎕CT
⍝ L∨R for Hypercomplex numbers
⍝   using the Hurwitz Residue function
⍝ Scalar L and R
⎕CT←1E¯10   ⍝ Needed for FP numbers only
:repeat     ⍝ Euclidean Algorithm
  T←L
  L←L UR R
  :Assert (|L)<|T
  R←T
:until (|L)≤SCT ⍝ FP numbers only
⍝ Rotate R into the first quadrant
⍝  or first two bi-quadrants
Z←rotateGCD R
```

Conclusions

Starting with two possible definitions of the Floor primitive function, we've shown that only the one by Hurwitz has a valid definition on Quaternions (due to Fractionality) and then only if we extend GCD to

Hurwitz Quaternions so as to enable Euclidean Division on Quaternions.

Thus, the proper definition of Greatest Common Divisor in APL on Quaternions is the derived function that passes `UR` as the Residue function to the `GCD` operator: (`UR GCD`), a.k.a. `UG` above.

Moreover, the principal value for GCD is defined for Quaternions to include rotation of the value into one of two adjacent bi-quadrants.

Neither definition of the Floor function allows Euclidean Division on Octonions, and it is unlikely any definition can.

Online Version

This paper is an ongoing effort and can be out-of-date the next day. To find the most recent version, goto http://sudleyplace.com/APL/ and look for the title of this paper on that page. Related papers such as "Hypercomplex Quotients in APL"[11], "Hypercomplex Numbers in APL"[10] , "Hypercomplex Notation in APL"[9], as well as "Rational & Variable-precision Floating Point Numbers"[12] may be found in the same place.

Executable Version

All of the above APL functions may be executed in NARS2000, an experimental APL interpreter available for free as Open Source software.

The latest released version of the NARS2000 software may be found in http://www.nars2000.org/download/ in either 32- or 64-bit versions. This software runs natively under Microsoft Windows XP or later as well as any Linux or Mac OS version which supports Wine (32-bit only) which acts as a translation layer.

The choice of which Floor function to invoke on Hypercomplex

numbers is under user-control. This choice applies not only to the Floor primitive, but also all other primitive functions directly or indirectly sensitive to Floor. NARS2000 uses McDonnell's Floor function when `⎕FEATURE[3]←0` and Hurwitz's Floor function when `⎕FEATURE[3]←1`.

When using McDonnell's Floor function, all primitives that depend on Floor on Quaternions signal a `DOMAIN ERROR`. Only when using Hurwitz's Floor function does the system produce a result.

References

1. Wikipedia, "Euclidean Division", https://en.wikipedia.org/wiki/Euclidean_division
2. Wikipedia, "Hurwitz Quaternion", https://en.wikipedia.org/wiki/Hurwitz_quaternion
3. McDonnell, E. E., "Complex Floor", APL73 http://www.jsoftware.com/papers/eem/complexfloor1.htm
4. Forkes, D., "Complex Floor Revisited", http://dl.acm.org/citation.cfm?id=805343
5. Hurwitz, A., Über die Entwicklung Komplexer Größen in Kettenbrüche, Acta Mathematica, 11, 1888 http://www.ulb.tu-darmstadt.de/tocs/94627940.pdf
6. Siobhan Roberts, "Genius At Play: The Curious Mind of John Horton Conway" , Bloomsbury, 2015, p. xvi.
7. McDonnell, E. E. "Complex Numbers", SATN40,
8. Hurwitz, A. (1919), “Vorlesungen über die Zahlentheorie der Quaternionen”, Berlin: J. Springer, JFM 47.0106.01
9. "Hypercomplex Notation in APL”

http://www.sudleyplace.com/APL/HyperComplex Notation in APL.pdf

10. "Hypercomplex Numbers in APL" http://www.sudleyplace.com/APL/HyperComplex Numbers in APL.pdf
11. "Hypercomplex Quotients in APL" http://www.sudleyplace.com/APL/HyperComplex Quotients in APL.pdf
12. "Rational & Variable-precision Floating Point Numbers", http://www.sudleyplace.com/APL/Rational & Variable-Precision FP.pdf
13. John Horton Conway, Derek Alan Smith (2003), On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry, A K Peters Ltd., page 56, ISBN 978-1-56881-134-5