Proposed Convention for Complex Numbers in CF NetCDF

Mike Dixon, EOL, NCAR, Boulder, CO, USA
Joe Hardin, PNNL, DOE/ARM, Richland, WA, USA
Ken Kehoe, NOAA, OU and DOW/ARM, Norman, OK, USA
Samuel Haimov, University of Wyoming, Laramie, WY, USA
Daniel Michelson, Environment and Climate Canada, Toronto, Canada
Mark Curtiss, Bureau of Meteorology, Melbourne, Australia

2017/12/28

Motivation

The ongoing development of the CfRadial convention, aimed at representing radar and lidar data in CF NetCDF, has highlighted the need to properly represent complex numbers in CF.

The intention with this proposal is to obtain agreement on adopting a complex number convention that (a) respects the CF approach, (b) satisfies the requirements of the radar and lidar research community and (c) provide a general solution so that complex numbers from any discipline can be accurately and simply represented.

The nature of complex numbers

A complex number can be expressed in the form:

$$P = X + iY$$

where **X** and **Y** are real numbers, and **i** = $\sqrt{-1}$.

For the complex number **P = X + iY**, **X** is called the **real** part, and **Y** is called the **imaginary** part.

A complex number can be represented on a Gaussian plane, as shown in Figure 1.

The point P is plotted at the (X, Y) Cartesian location.

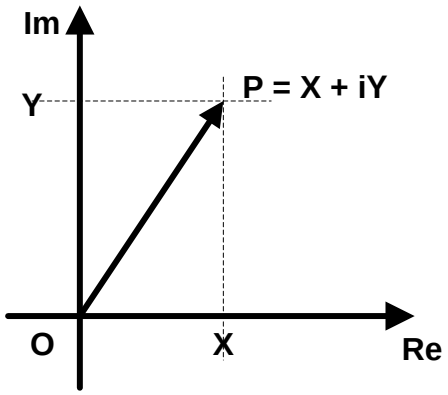

Figure 1: Complex number represented as a Cartesian point P on a Gaussian plane

Complex numbers can also be represented in a polar form.

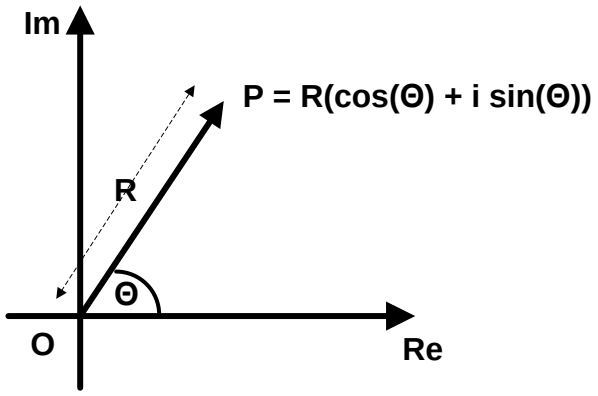

Figure 2: Complex number represented in polar form

R is the distance of point P from the origin O. The argument θ is the angle between the Real axis and the vector, in a counter-clockwise direction.

The relationship between the Cartesian form and the polar form is:

$$X + iY = R(\cos(\theta) + i\sin(\theta))$$

Furthermore, Euler's formula can be used to derive the exponential form:

$$\mathrm{Re}^{i\theta} = R(\cos(\theta) + i\sin(\theta))$$

Representing complex numbers as two reals

The above discussion demonstrates that no matter which form you choose to use, a complex number requires the storage of two real numbers, as well as a clear description of which form you are using.

It seems reasonable to determine the form being used by inspecting the units attribute.

Use of complex numbers in radar data

At the most raw level, radars generate time series of I (in-phase) and Q (quadrature) signals. These two signal components are considered to comprise a complex number.

Additionally, the spectra of radar signals are stored as complex numbers.

Because of the very large dynamic range of radar signals (up to 10 orders of magnitude), these complex numbers are often stored as power, in log units, and phase, in degrees.

Storing complex numbers in NetCDF

Because complex numbers must be represented by 2 real numbers, a complex variable will have one extra dimension, of size 2.

We propose that this dimension should be the last of the dimensions used for a variable – i.e. the 2 parts of the number are stored immediately one after the other in the array.

So, for example, a radar variable holding complex data for an I/Q time series could be represented as follows

```
time = 3000 ;
range = 996 ;
complex = 2;
float IQ(time, range, complex) ;
   IQ:is_complex = “true” ;
   IQ:long_name = "time_series_inphase_and_quadrature" ;
   IQ:standard_name = "radar_time_series_inphase_and_quadrature" ;
   IQ:units = "volt" ;
   IQ:_FillValue = -9999F ;
   IQ:coordinates = "time range" ;
```

Attribute indicating variable is complex

As indicated in the example above, we propose that the string attribute:

```
is_complex = "true";
```

be attached to any variable that holds complex numbers.

Units attribute for real/imaginary form

If the complex number is stored using real and imaginary parts, the units of both parts will be the same.

This is true for the example above, where we have:

```
IQ:units = "volts";
```

Units attribute for polar form

If the polar or exponential form is used, the units of the two parts will not be the same.

We could use a single units attribute, as now, but make it comma-delimited to separate the different units.

For example:

```
IQ:units = "dBm,degree";
```

Or we could have 2 separate units attributes:

For example:

```
IQ:units_first_part = "dBm";
IQ:units_second_part = "degree";
```