On converting image coordinates to celestial coordinates using the World Coordinate System (WCS)

This note explains how to convert image pixel coordinates (p_x, p_y) to celestial spherical coordinates (α, δ). The equations provided here are taken from Calabretta, M. R., & Greisen, E. W. (2002), “Representations of celestial coordinates in FITS”, Astronomy & Astrophysics, 395(3), 1077-1122. This paper is commonly referred to as WCS paper II. We only consider the following special case:

- Gnomonic (TAN) projection.

WCS-defined keyword and value pairs can be found in FITS images with astrometry.

```
WCSAXES =                    2 / Number of coordinate axes
CRPIX1  =               2400.5 / Pixel coordinate of reference point
CRPIX2  =               2400.5 / Pixel coordinate of reference point
PC1_1   =    0.000112212319481 / Coordinate transformation matrix element
PC1_2   =   -2.85904573777E-05 / Coordinate transformation matrix element
PC2_1   =    2.85904573777E-05 / Coordinate transformation matrix element
PC2_2   =    0.000112212319481 / Coordinate transformation matrix element
CDELT1  =                  1.0 / [deg] Coordinate increment at reference point
CDELT2  =                  1.0 / [deg] Coordinate increment at reference point
CUNIT1  = 'deg'                / Units of coordinate increment and value
CUNIT2  = 'deg'                / Units of coordinate increment and value
CTYPE1  = 'RA---TAN'           / Right ascension, gnomonic projection
CTYPE2  = 'DEC--TAN'           / Declination, gnomonic projection
CRVAL1  =        17.4019485165 / [deg] Coordinate value at reference point
CRVAL2  =   -71.2953701226 / [deg] Coordinate value at reference point
LONPOLE =                180.0 / [deg] Native longitude of celestial pole
LATPOLE =   -71.2953701226 / [deg] Native latitude of celestial pole
MJDREF  =                  0.0 / [d] MJD of fiducial time
RADESYS = 'FK5'                / Equatorial coordinate system
EQUINOX =               2000.0 / [yr] Equinox of equatorial coordinates
```

The WCS defines the (p_x, p_y) to (α, δ) conversion as a three-step process. They are the following.

1. Pixel coordinates to projection plane coordinates: (p_x, p_y) → (x, y)
2. Projection plane coordinates to native spherical coordinates: (x, y) → (ϕ, θ)
3. Native spherical coordinates to celestial spherical coordinates: (ϕ, θ) → (α, δ)

Step 1: $(p_x, p_y) \rightarrow (x, y)$

We can estimate (x, y) from (p_x, p_y) as follows using equation 1 from WCS paper II. Equation 1 from WCS paper II is given below.

$$x_i = s_i \sum_{j=1}^{N} m_{ij}(p_j - r_j)$$

Where
p_j is (p_x, p_y),
r_j is `(CRPIX1, CRPIX2)`,
s_i is `(CDELT1, CDELT2)`, and
m_{ij} is the PC matrix when N = 2.
The PC matrix is given by

```
[[PC1_1, PC1_2],
 [PC2_1, PC2_2]].
```

```
x = CDELT1 * (PC1_1 * (px - CRPIX1) + PC1_2 * (py - CRPIX2))
y = CDELT2 * (PC2_1 * (px - CRPIX1) + PC2_2 * (py - CRPIX2))
```

Step 2: $(x, y) \rightarrow (\phi, \theta)$

To estimate (ϕ, θ) from (x, y), we can use equations 14, 15, and 55 from WCS paper II. The equations are given below.

$$\phi = arg(-y, x)$$

$$R_\theta = \sqrt{\left(x^2 + y^2\right)}$$

$$\theta = tan^{-1}\left(\frac{180}{\pi R_\theta}\right)$$

Where arg() is an inverse tangent function that returns the correct quadrant, i.e.
if

$$(x, y) = (r\, cos\, \beta, r\, sin\, \beta)\ with\, r > 0$$

then

$$arg\,(x, y) = \beta$$

Step 3: (ϕ, θ) → (α, δ)

To estimate (α, δ) from (ϕ, θ), we can use equation 2 from WCS paper II.

$$\alpha = \alpha_p + arg(sin\,\theta\,cos\,\delta_p - cos\,\theta\,sin\,\delta_p\,cos(\phi - \phi_p),\; - cos\,\theta\,sin(\phi - \phi_p))$$

$$\delta = sin^{-1}(sin\,\theta\,sin\,\delta_p + cos\,\theta\,cos\,\delta_p\,cos(\phi - \phi_p))$$

Where
(α_p, δ_p) = (CRVAL1, CRVAL2) and
ϕ_p = 180° (however, ϕ_p is 0° when CRVAL2 = 90°).

Python script

The above equations have been implemented in Python. Then, the results were compared with those given by the Astropy python package that can carry out WCS transformations (link: https://docs.astropy.org/en/stable/wcs/). They were very closely matched. The implementation is provided below.

```
import numpy as np

# Test image coordinates.
px, py = 1000, 3000

# For conversion of (px, py) to (x, y) in projection plane coordinates.
PC1_1 = 0.000112212319481
PC1_2 = -2.85904573777E-05
PC2_1 = 2.85904573777E-05
PC2_2 = 0.000112212319481

CRPIX1 = 2400.5
CRPIX2 = 2400.5

CDELT1 = 1
CDELT2 = 1

x = CDELT1 * ((PC1_1 * ( px - CRPIX1)) + (PC1_2 * ( py - CRPIX2)))
y = CDELT2 * ((PC2_1 * ( px - CRPIX1)) + (PC2_2 * ( py - CRPIX2)))

# For conversion of (x, y) to (phi, theta) in native spherical coordinates.
phi = np.arctan2(x, -1 * y)
R_theta = (x ** 2 + y ** 2) ** 0.5
theta = np.arctan(180 / (np.pi * R_theta))
```

```
# For conversion of (phi, theta) to (alpha, delta) in celestial spherical coordinates.
CRVAL1 = 17.4019485165
CRVAL2 = -71.2953701226

phi_p = np.deg2rad(180)
alpha_p = np.deg2rad(CRVAL1)
delta_p = np.deg2rad(CRVAL2)

alpha = alpha_p + np.arctan2(-1 * np.cos(theta) * np.sin(phi - phi_p),
                             (np.sin(theta) * np.cos(delta_p)) -
                             (np.cos(theta) * np.sin(delta_p) * np.cos(phi - phi_p)),
                      )
delta = np.arcsin(np.sin(theta) * np.sin(delta_p) +
                      np.cos(theta) * np.cos(delta_p) * np.cos(phi - phi_p))

# print alpha, delta.
np.rad2deg(alpha), np.rad2deg(delta)
```

The printed output was (16.85923104, -71.26735802).
The Astropy WCS module gave a value of (16.85923104, -71.26735802).

PC matrix in terms of only rotation and scale

The interpretation of the PC matrix in terms of only rotation and scale is given in WCS paper II, Section 6.1. Equations 187 and 188 tell us that,

```
[[PC1_1, PC1_2],  = [[cos(ρ), -λ * sin(ρ)],
 [PC2_1, PC2_2]]     [(1/λ) * sin(ρ), cos(ρ)]]
```

where λ = CDELT2 / CDELT1.

In our case, λ will be +/-1. So,

for λ = 1,

```
[[PC1_1, PC1_2],  = [[cos(ρ), -sin(ρ)],
 [PC2_1, PC2_2]]     [sin(ρ),  cos(ρ)]]
```

for λ = -1,

```
[[PC1_1, PC1_2],  = [[ cos(ρ), sin(ρ)],
 [PC2_1, PC2_2]]     [-sin(ρ), cos(ρ)]]
```

Here ρ is the rotation angle.

Plate scale estimation

Plate scales along X and Y can be calculated as follows.

```
plate scale along X = SQRT((CDELT1 * PC1_1)^2 + (CDELT2 * PC2_1)^2)
plate scale along Y = SQRT((CDELT1 * PC1_2)^2 + (CDELT2 * PC2_2)^2)
```