import numpy as np  
from datetime import datetime

**==========================================================**

**Algorithm Theoretical Basis Document (ATBD)**

**Estimate Zonal Winds from Jovian Cylindrical Maps**

**==========================================================**

"""

1. Introduction

This Algorithm Theoretical Basis Document (ATBD) describes the method implemented in the function estimate\_zonal\_winds and its helper functions \_xcorr1d\_circular, \_meters\_per\_px\_lon\_at\_lat, and \_latitudes\_from\_height. The algorithm estimates zonal wind speeds on Jupiter using two sequentially observed global maps (typically cylindrical projections).

The algorithm is tailored for Jovian atmospheric studies where wind measurements are inferred from the advection of cloud tracers. It leverages one-dimensional circular cross-correlation applied along lines of constant latitude.

1. Inputs and Outputs

Inputs:

* img1\_path, img2\_path: File paths to the first and second 2D images of Jupiter’s cylindrical projection.
* dt\_seconds: Time interval between observations (in seconds).
* meters\_per\_pixel (optional): Direct pixel scale in meters/pixel. If not given, derived from deg\_per\_px.
* deg\_per\_px (optional): Degrees per pixel in longitude.
* ref\_latitude\_deg (optional): Reference latitude for projection scaling.

Outputs:

* Arrays of planetographic latitude values and corresponding mean zonal wind velocities in m/s.

1. Mathematical Basis

**3.1 Cross-Correlation to Measure Displacements**

At each latitude row of the cylindrical projection, cloud features are treated as intensity variations in a 1D signal I(λ)I(\lambda), where λ\lambda is longitude.

The displacement Δx\Delta x between two images is estimated by maximizing the circular cross-correlation:

R12(τ)=∑xI1(x)⋅I2(x+τ)R\_{12}(\tau) = \sum\_{x} I\_1(x) \cdot I\_2(x+\tau)

where indices wrap around the global longitude extent (circular boundary conditions). The offset τ\tau at which R12(τ)R\_{12}(\tau) is maximized corresponds to the pixel displacement in longitude.

Python implementation:

* scipy.signal.correlate and scipy.signal.correlation\_lags provide correlation computation and lag indexing.
* Numpy (numpy.argmax) locates the displacement peak.

References:

* [SciPy Correlate Documentation](https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.correlate.html)

**3.2 Pixel-to-Meter Conversion at a Given Latitude**

The local linear distance corresponding to a pixel in longitude depends on latitude. For a planet with equatorial radius ReqR\_{eq} and flattening ff:

e2=2f−f2e^2 = 2f - f^2 ϕc=arctan⁡((1−e2)⋅tan⁡(ϕ))\phi\_c = \arctan((1-e^2) \cdot \tan(\phi)) Rlat=Req⋅cos⁡(ϕc)R\_{lat} = R\_{eq} \cdot \cos(\phi\_c)

where:

* ϕ\phi = planetographic latitude
* ϕc\phi\_c = corresponding planetocentric latitude
* RlatR\_{lat} = local radius of rotation

The meters per pixel along longitude at latitude ϕ\phi:

mpx(ϕ)=Rlat⋅Δλpxm\_{px}(\phi) = R\_{lat} \cdot \Delta \lambda\_{px}

where Δλpx\Delta \lambda\_{px} is the pixel resolution in radians.

Python implementation:

* Trigonometric transformations via numpy.radians, numpy.cos, numpy.tan, and numpy.arctan.
* Pixel scaling in meters computed by \_meters\_per\_px\_lon\_at\_lat.

**3.3 Zonal Wind Velocity Computation**

The zonal velocity u(ϕ)u(\phi) at latitude ϕ\phi is given by:

u(ϕ)=ΔxmetersΔtu(\phi) = \frac{\Delta x\_{meters}}{\Delta t}

where:

* Δxmeters=Δxpixels⋅mpx(ϕ)\Delta x\_{meters} = \Delta x\_{pixels} \cdot m\_{px}(\phi)
* Δt\Delta t = observation time difference (seconds)

Python implementation:

* Conversion via numpy array arithmetic.
* Time interval provided as dt\_seconds.

**3.4 Latitude Grid Mapping**

Each row in the cylindrical projection corresponds to a latitude. Assuming a cylindrical map of height Ny pixels:

ϕ(y)=(yNy−0.5)×180∘\phi(y) = \left( \frac{y}{Ny} - 0.5 \right) \times 180^{\circ}

Python implementation:

* Computed in \_latitudes\_from\_height using numpy.linspace.

1. Assumptions and Limitations

* Assumes cylindrical projection maps are correctly navigated (latitude/longitude registered).
* Assumes uniform time difference between maps.
* Zonal flows dominate over meridional displacements for latitudinal averaging.
* Cloud features are assumed to act as passive tracers.

1. Validation Strategy

* Compare with published HST-derived wind profiles (e.g., García-Melendo & Sánchez-Lavega, 2001).
* Internal validation using synthetic maps with known shifts.

1. References

* García-Melendo, E., & Sánchez-Lavega, A. (2001). A Study of the Stability of Jovian Zonal Winds from HST Images: 1995–2000. *Icarus*, 152(2), 316-330.
* SciPy Signal Processing Documentation: <https://docs.scipy.org/doc/scipy/reference/signal.html>
* NumPy Documentation: <https://numpy.org/doc/stable/>  
  """