Technische Universität Berlin
Flugführung und Luftverkehr

Aircraft Navigation

GNSS

Exercise lecture

Prof. Dr. ir. Maarten Uijt de Haag

Mats Martens, M.Sc.

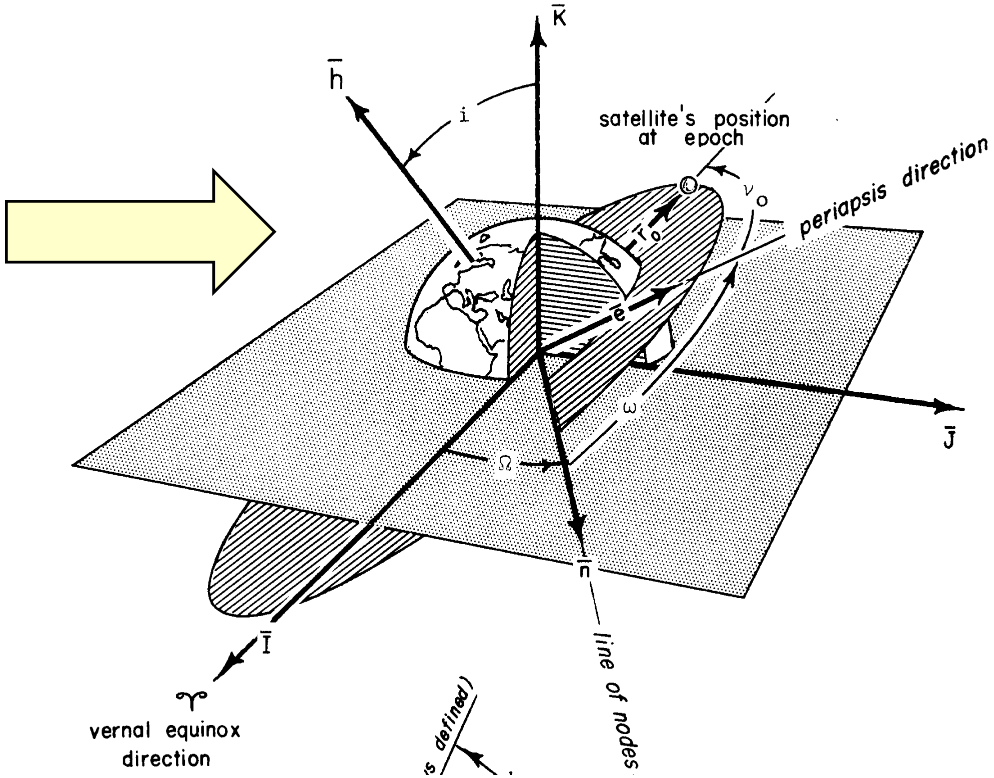
Institute of Aeronautics and Astronautics
Chair of Flight Guidance and Air Transport

Satellite Position Calculation

- An important part of the GPS position calculation is the determination of the satellite position

$$PR_j = \sqrt{(x_u - x_j)^2 + (y_u - y_j)^2 + (z_u - z_j)^2} + c\delta t_u$$

- Given:
  - Set of Ephemeris data (also known as ephemerides)
  - Time (GPS time: seconds in week)
  - 'Ephcal'

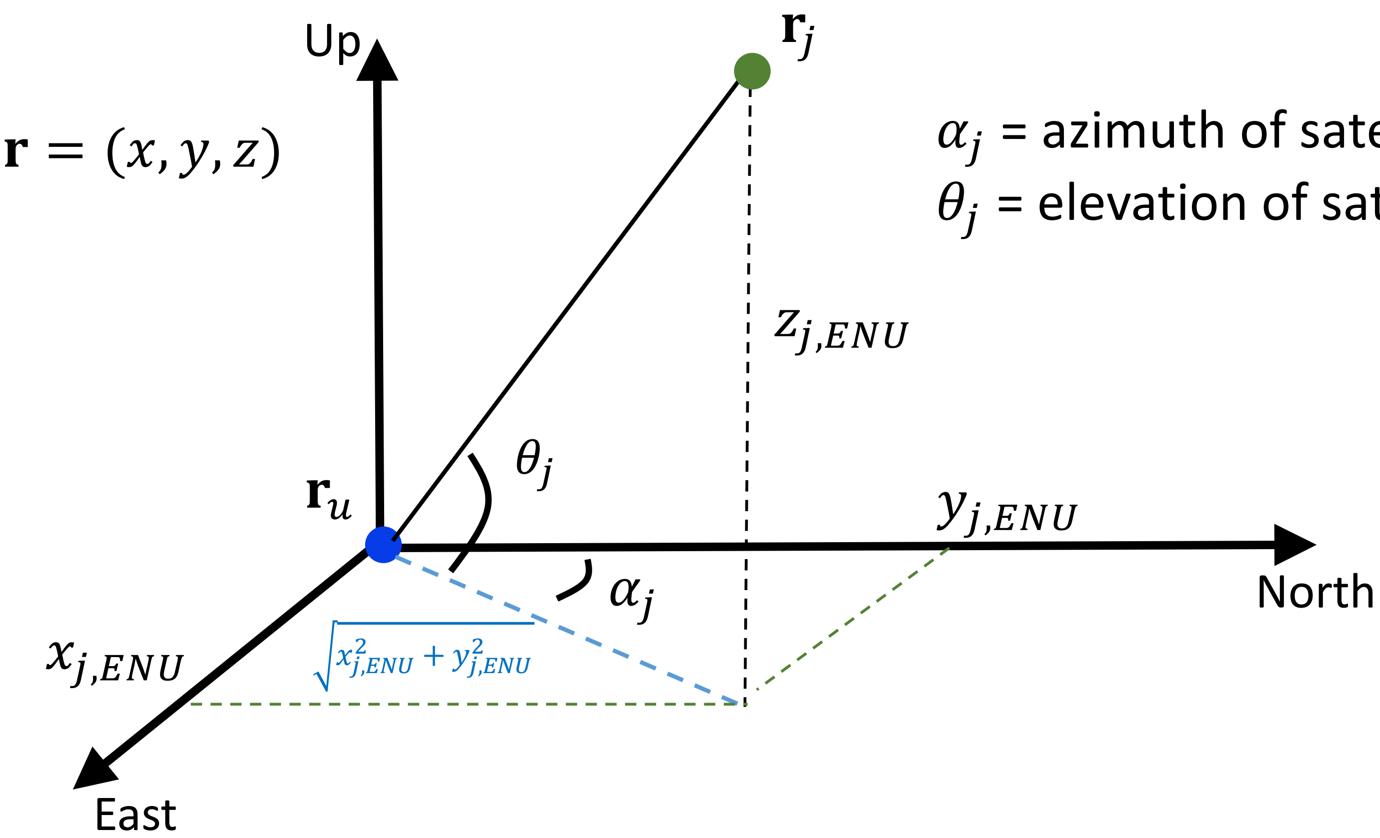


Satellite Position Calculation – ‘Ephcal’

- (svpos, svclock, ecode) = ephcal(transmit_time, ephem, svid)
- Subroutine to calculate the position of a satellite in ECEF.
- Inputs:
  - **transmit_time**: time at which the satellite signal was transmitted
  - **ephem**: the ephemeris parameters (parameters that describe the satellite’s orbit)
  - **svid**: identification of the satellite
- Outputs:
  - **svpos**: satellite position in ECEF (3-by-1 vector)
  - **svclock**: satellite clock error (w.r.t. GPS time)
  - **ecode**: erorr code
    - ecode = 0: no detectable errors, ecode = 1: semi-major axis <= 0.0, ecode = 2: eccentric anomaly did not converge, ecode = 3: no ephemeris for this satellite

Technische Universität Berlin
Flugführung und Luftverkehr

Satellite Elevation and Azimuth



α_j = azimuth of satellite 'j'
θ_j = elevation of satellite 'j'

$$\alpha_j = atan\left(\frac{x_{j,ENU}}{y_{j,ENU}}\right) \qquad \theta_j = atan\left(\frac{z_{j,ENU}}{\sqrt{x_{j,ENU}^2 + y_{j,ENU}^2}}\right)$$

Use 'atan2'

Technische Universität Berlin
Flugführung und Luftverkehr

Part 1

- Use the *template* provided on ISIS
  - comp_svpos_template.py
- Load the Ephemeris data from file '**ephem.dat**'
- For all satellites in the data set compute the **azimuth** and **elevation** and print the values

Time of Transmission (TOT) vs Reception (TOR)

- Typically, you have the time-of-reception, NOT the time-of-transmission. The time of transmission can then be estimated using:

$$TOT_j \approx TOR - \frac{PR_j}{c} - svclock_j$$

From measurement

From Ephcal

→ So, how do we typically go about solving this?

Time of Transmission (TOT) vs Reception (TOR)

$$TOT_j \approx TOR - \frac{PR_j}{c}$$

(svpos, $svclock_j$, ecode) = ephcal(TOT_j, ephem, svid)

Check for abs($svclock_j$) < 0.01)

$$TOT_j \approx TOR - \frac{PR_j}{c} - svclock_j$$

(svpos, $svclock_j$, ecode) = ephcal(TOT_j, ephem, svid)

Part 2

- Use the *template* provided on ISIS
  - comp_tot_svpos_template.m/py
- Load the pseudoranges and corresponding satellite ID’s from ‘**pr.dat**’
- Load the Ephemeris data from file ‘**ephem.dat**’
- For all satellites in the pr.dat data set
  - Implement time of transmission correction from the previous slide
  - compute the azimuth and elevation

Earth Rotation Correction – Sagnac Effect

- While the GNSS signal travels from the satellite to the user on the Earth (receiver), the Earth rotates. As the ECEF frame rotates with the Earth, the position at time-of-transmission must be adjusted to reflect the position at the time-of-reception.

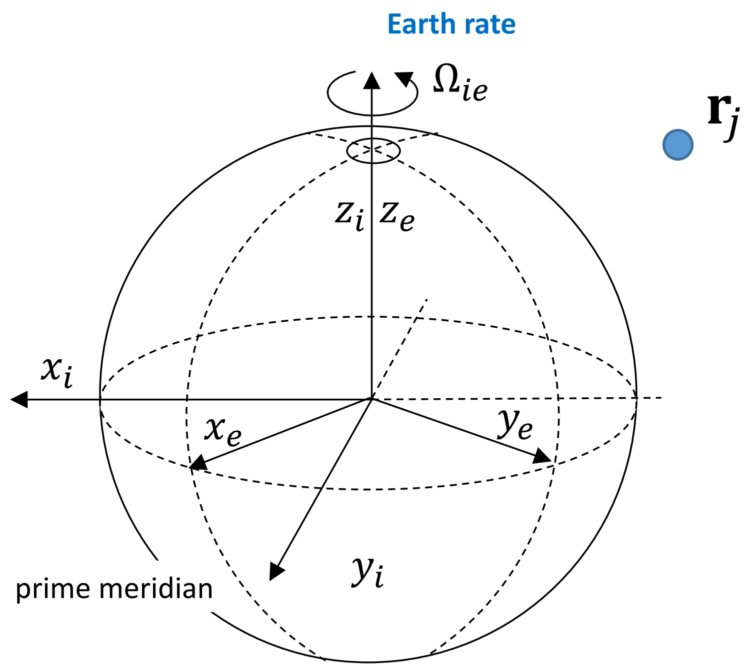
$$\Delta t = travel\ time = \frac{PR_j}{c} + svclock_j$$

Angle the Earth has rotated during signal travel time:

$$\Delta t \cdot \boldsymbol{\omega}_{ie} = \begin{bmatrix} 0 \\ 0 \\ \Delta t \cdot \Omega_{ie} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \alpha \end{bmatrix}$$

ECEF correction (add to ECEF coordinates):

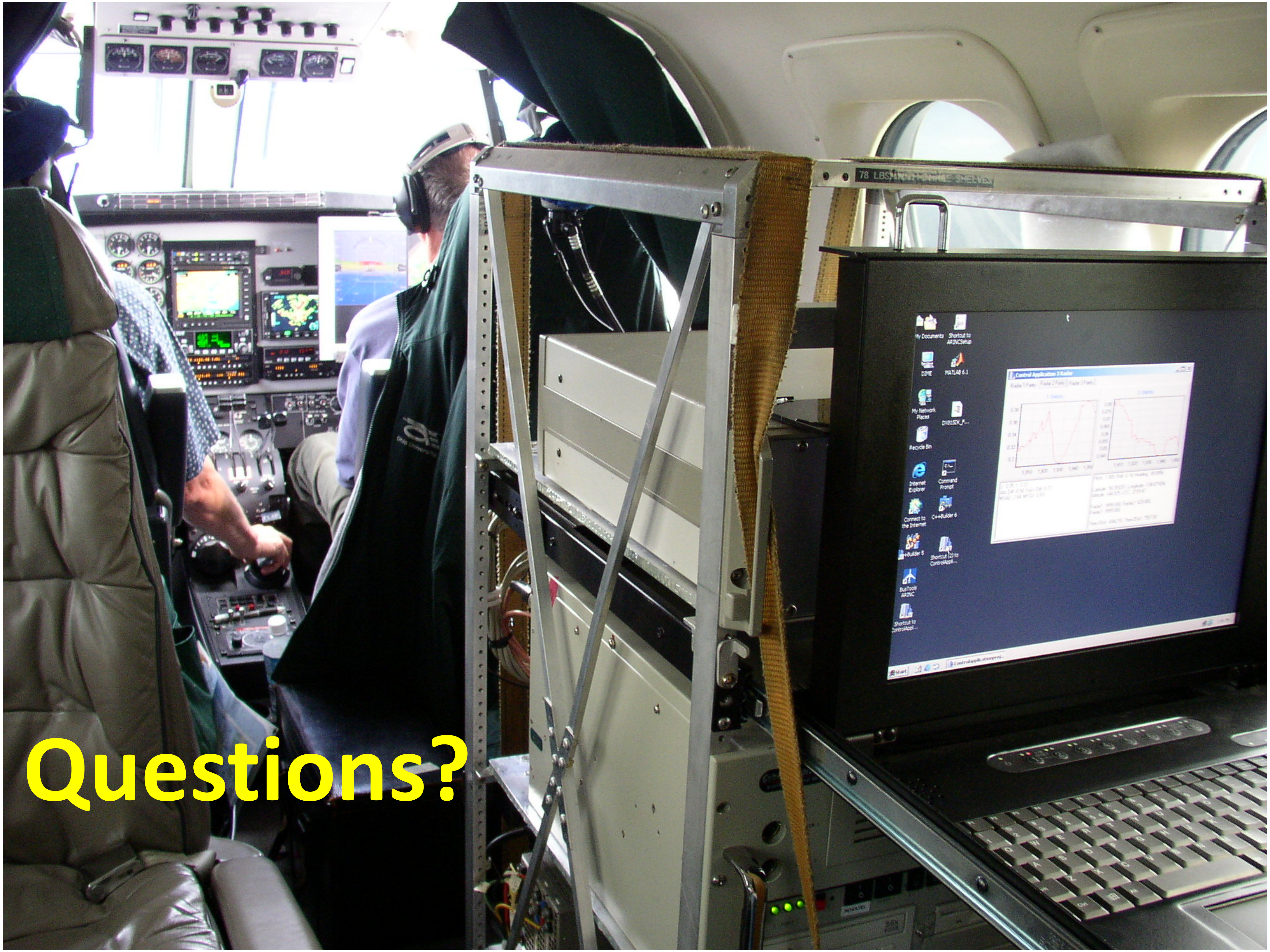
$$\Delta \mathbf{r}_{j,ECEF} = \Delta t \cdot \boldsymbol{\omega}_{ie} \times \mathbf{r}_{j,ECEF} = \begin{bmatrix} \alpha \cdot y_{j,ECEF} \\ -\alpha \cdot x_{j,ECEF} \\ 0 \end{bmatrix}$$

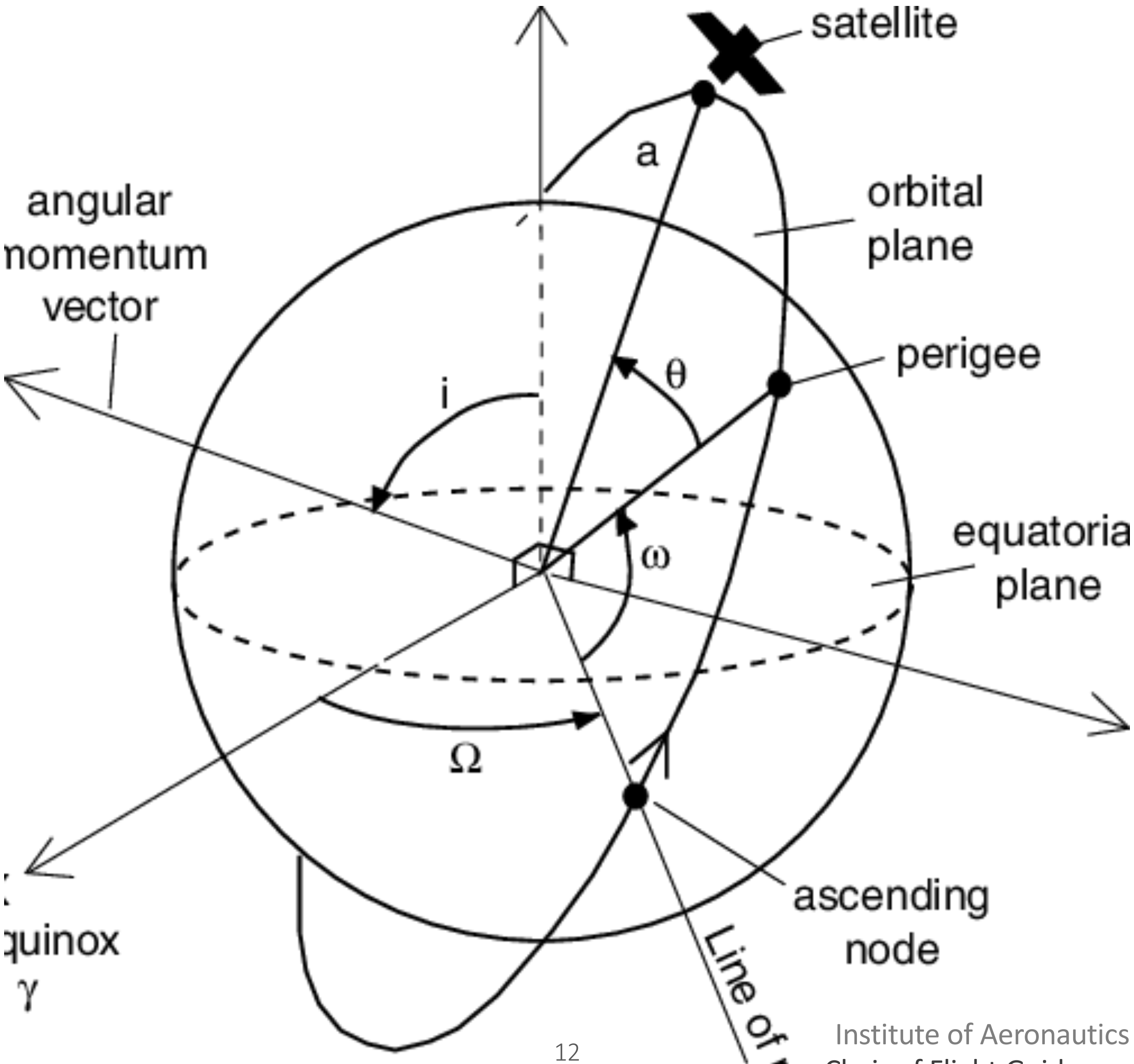


Ω_{ie} = eanvel in code

Part 3

- Build upon Part 2
- Compute the correction and add to the satellite position
- Compute the azimuth and elevation


Questions?


satellite
a
orbital
plane
angular
nomentum
vector
perigee
θ
i
ω
equatoria
plane
Ω
ascending
node
Line of
quinox
γ