Department of Aeronautical and Aviation Engineering
航空及民航工程學系

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Satellite Navigation: Basics and Single Point Positioning

AAE4203 – Guidance and Navigation

Dr. Weisong Wen
Assistant Professor

Department of Aeronautical and Aviation Engineering
The Hong Kong Polytechnic University

Week 2

Outline

- Background of GPS
- GPS Overview
- Basic principle of the GNSS

Background of GNSS

Global Navigation Satellite System (GNSS)

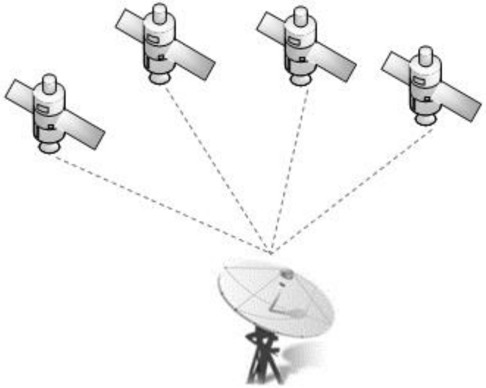
Global systems

- **US GPS**
- **Russia GLONASS**
- **China BDS**
- **EU Galileo**

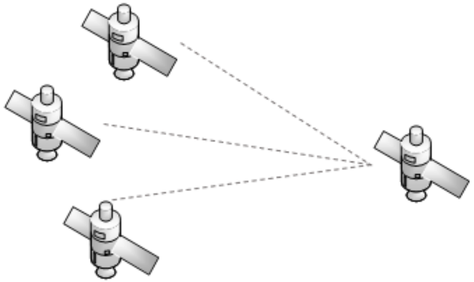
Regional systems

- **Japan QZSS**
- **India IRNSS**

Three Important Services of GNSS



Active Positioning (AP)



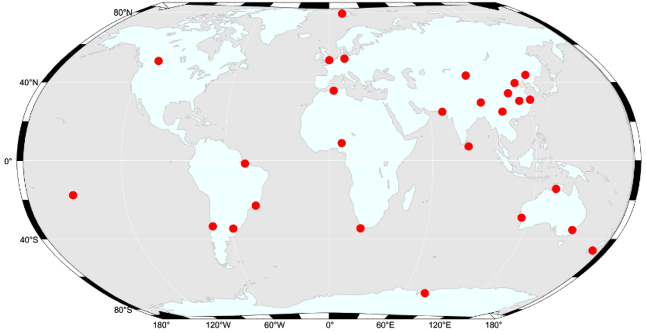
Inter-satellite link (ISL)

Short Message Communication (SMC)

Existing GNSS Data Sources

- GNSS Tracking Stations
  - Global
    - CDDIS
    - IGS center of WHU
    - iGMAS
  - Regional
    - Hong Kong CORS Network
    - EPN

IGS:500+ GNSS stations



iGMAS: 37 GNSS stations

Hong Kong: 19 GNSS stations

EPN: 400+ GNSS stations

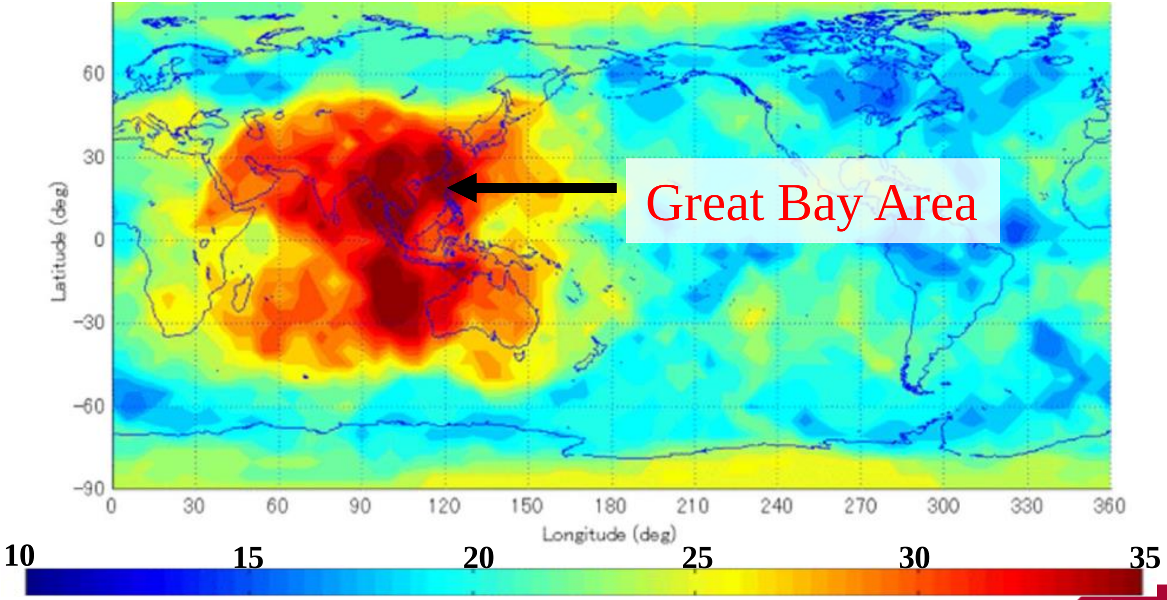
CDDIS: Crustal Dynamics Data Information System
IGS: International GNSS Service
iGMAS: integrated GNSS Monitoring and Assessment System
CORS: Continuously Operating Reference Station

Existing GNSS Frequency Resources

System	Owned GNSS frequency resources				
GPS	L1 1575.42 MHz	L2 1227.60 MHz	L5 1176.45 MHz		
GLONASS	L1 1598-1605 MHz	L2 1243-1249 MHz	L3 1202.025 MHz		
Galileo	E1 1575.42 MHz	E6 1278.75 MHz	E5b 1207.14 MHz	E5a 1176.45 MHz	
BDS-2	B1I 1561.10 MHz	B31 1268.52 MHz	B2I 1207.14 MHz		
BDS-3	B1C 1575.42 MHz	B1I 1561.10 MHz	B3I 1268.52 MHz	B2b 1207.14 MHz	B2a 1176.45 MHz
QZSS	L1 1575.42 MHz	LEX 1278.75 MHz	L2 1227.60 MHz	L5 1176.45 MHz	
IRNSS	S 2492.028 MHz	L5 1176.45 MHz			

New Era of GNSS

- GPS, GLONASS, Galileo and BDS
- Visible GNSS satellites with mask angle > 30º in 2020



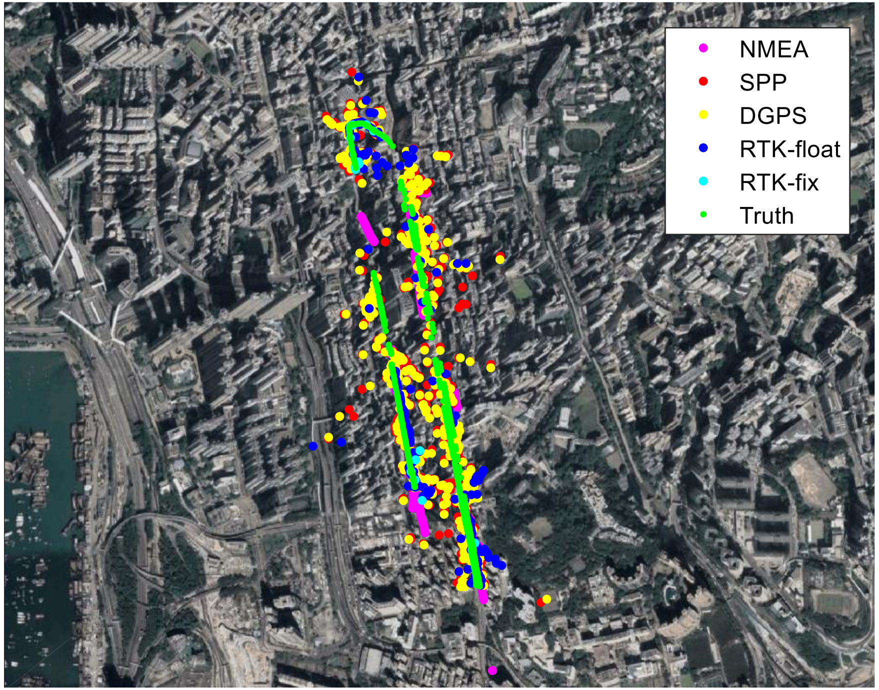
Application of the GNSS

GNSS in Autonomous Driving

Model					
Company Name	Waymo	Ford	Tesla	Nuro	Oxbotica
Positioning Solution	GNSS-RTK/INS/LiDAR/HD Map	GNSS-RTK/INS/LiDAR/HD Map	GNSS/INS/LiDAR/HD Map	GNSS-RTK/INS/LiDAR/HD Map	GNSS-RTK/INS/LiDAR/HD Map
Tested Scenario	Open area/sub-urban	Open area/sub-urban	Open area/sub-urban	Open area/sub-urban	Open area/sub-urban
Level of Automation	L5	L5	L2.5	L5	L5

ADAS*: Advanced Driving Assistant System

GNSS Performance in Urban Canyons



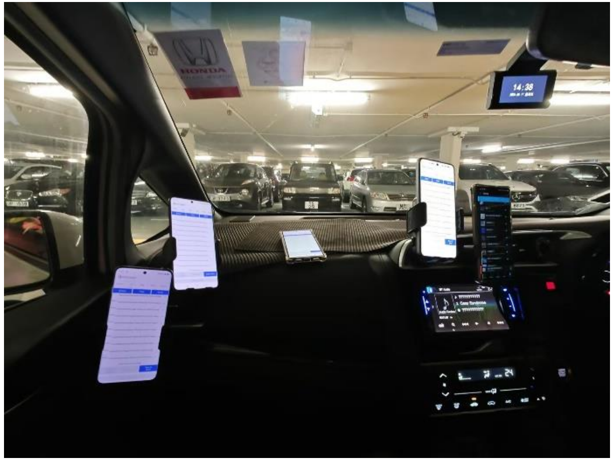
nSat	Full nSat	Ratio	HDOP	VDOP	XDOP	YDOP
8.50	21.70	39.03%	2.82	4.69	1.87	1.85

Type	Availability	2D Error	X Error	Y Error	2D STD	X STD	Y STD
NMEA	91.43%	84.74	78.75	16.71	85.11	87.68	16.01
SPP	76.94%	51.49	31.71	32.28	61.28	49.48	43.73
DGNSS	72.07%	45.87	28.91	28.62	57.41	46.54	39.74
RTK FLOAT	31.40%	28.91	14.69	21.66	44.59	30.13	35.10
RTK FIX	2.09%	10.10	5.88	7.12	12.87	9.89	9.25

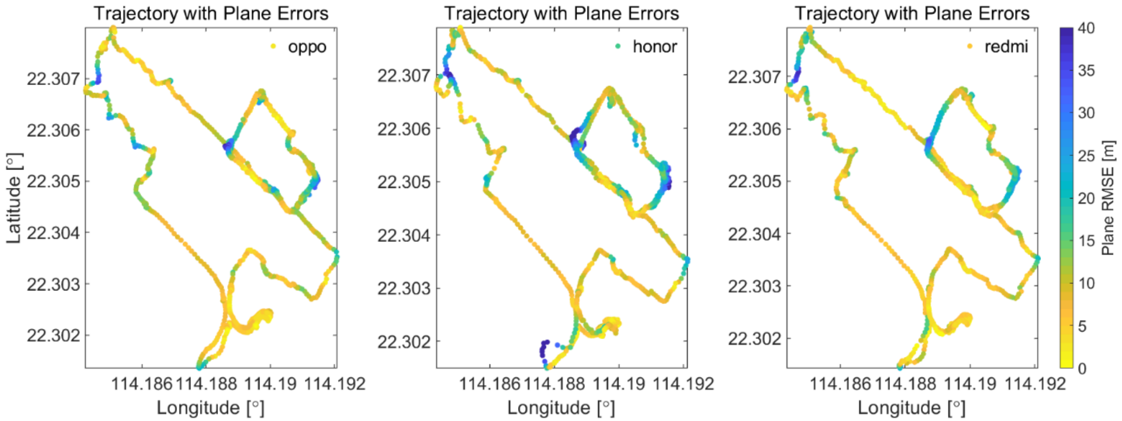
SPP: Single Point Positioning
RTK: Real-time Kinematic

GNSS in Smartphone Navigation

Model	OPPO K12	Honor X50	Redmi Note13Pro
Picture			
Company Name	OPPO	Huawei	Xiaomi
Price	1680.53 RMB	1238.05 RMB	1326.55 RMB
Positioning Solution	GNSS-RTK	GNSS-RTK	GNSS-RTK
Tested Scenario	Medium urban/harsh urban	Medium urban/harsh urban	Medium urban/harsh urban

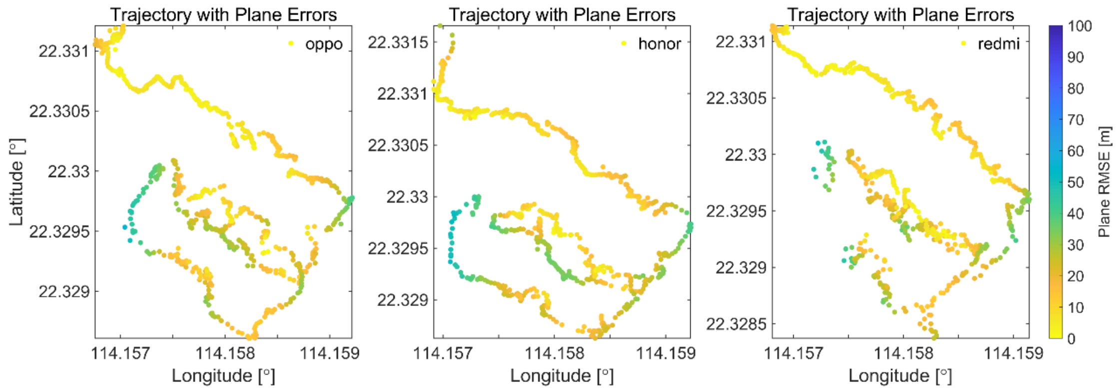


GNSS in Smartphone Navigation



Vehicle-mounted smartphone GNSS-RTK positioning results

Smartphone	Plane RMSE (m)	Plane STD (m)
Oppo	13.36	7.73
Honor	17.38	9.89
Redmi	13.23	7.50



Pedestrian-holding smartphone GNSS-RTK positioning results

Smartphone	Plane RMSE (m)	Plane STD (m)
Oppo	19.20	10.56
Honor	21.29	11.57
Redmi	18.10	10.43

Problem of GNSS in Urban Canyons

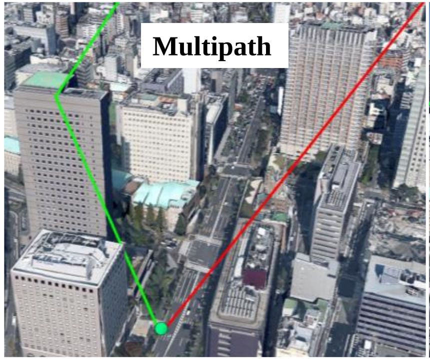
NLOS*: Non-line-of-sight

Problem 1: Poor GNSS measurements quality:

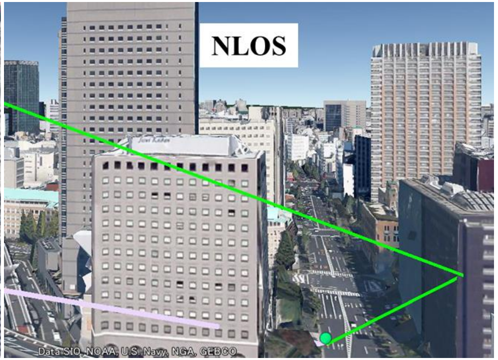
- NLOS receptions
- Multipath effects
- …

Problem 2: Poor satellite geometry:

- Limited satellite numbers
- Hard to resolve integer ambiguities successfully
- …

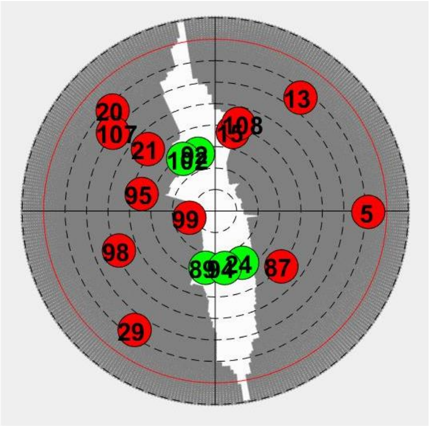


Hsu, 2016



Hsu, 2016

Poor GNSS measurements quality



Poor satellite geometry

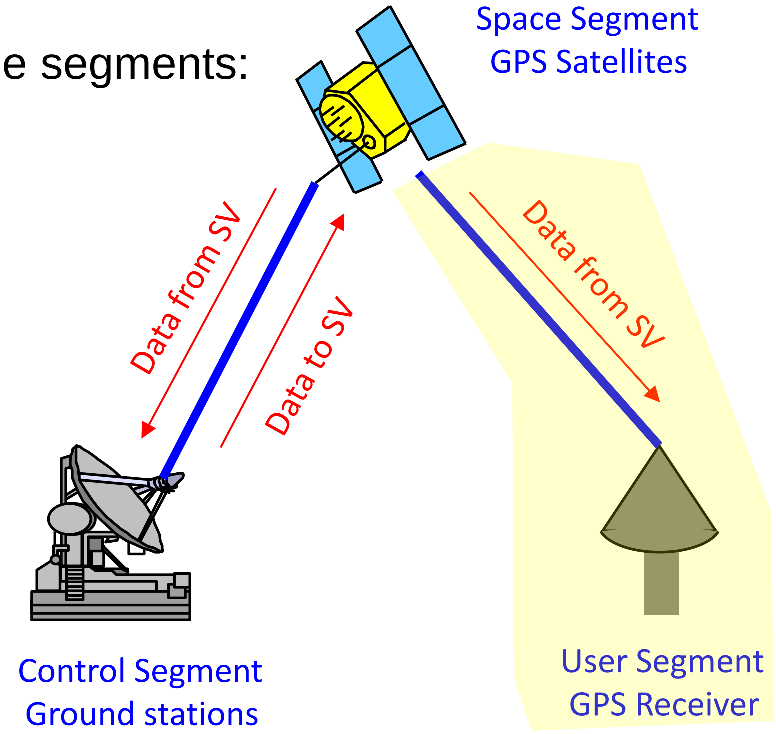
[1] J. Breßler, etc, "GNSS positioning in non-line-of-sight context—A survey," *ITSC 2016*.
[2] Hsu, Li-Ta, etc. "3D building model-based pedestrian positioning method using GPS/GLONASS/QZSS and its reliability calculation." *GPS solutions,* 2016

GPS Overview

System Configuration of GPS

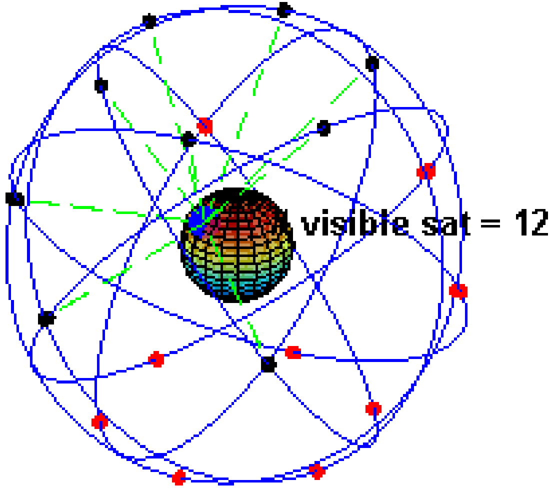
> Satellite navigation system is consisted of three segments:

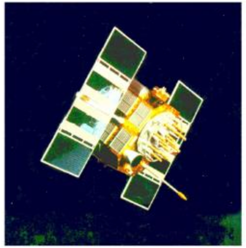
1. Control segment
   - Command infrequent small maneuvers to maintain orbit
   - Keep the synchronization of GPS time
2. Space segment (broadcasting)
   - 31+ medium earth orbit (MEO) satellites
   - 6 orbit planes
3. User segment
   - Antenna
   - A/D converter
   - Signal processing
   - **<u>Positioning algorithm</u>**



Space Segment

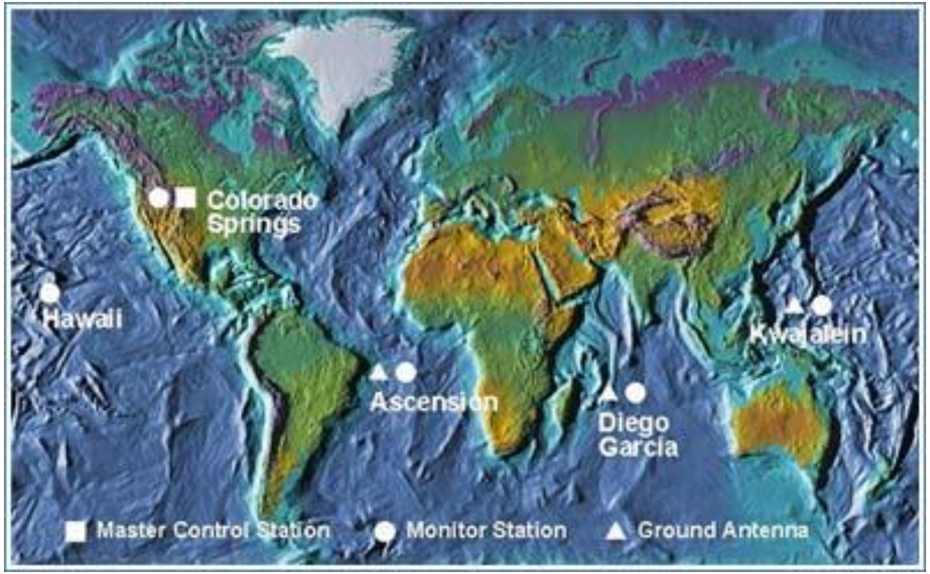
> Able to see 5 to 8 satellites at any point on the earth
> Each satellite has atomic clocks
> 32 satellites in 6 orbital planes (5-6 satellite per orbit)
> 20,200 km altitude, 55 degree inclination
> Two revolutions per sidereal day
> One sidereal day is 23 hours 56 minutes 4.091seconds
> SVs repeat more or less the same ground track on each day





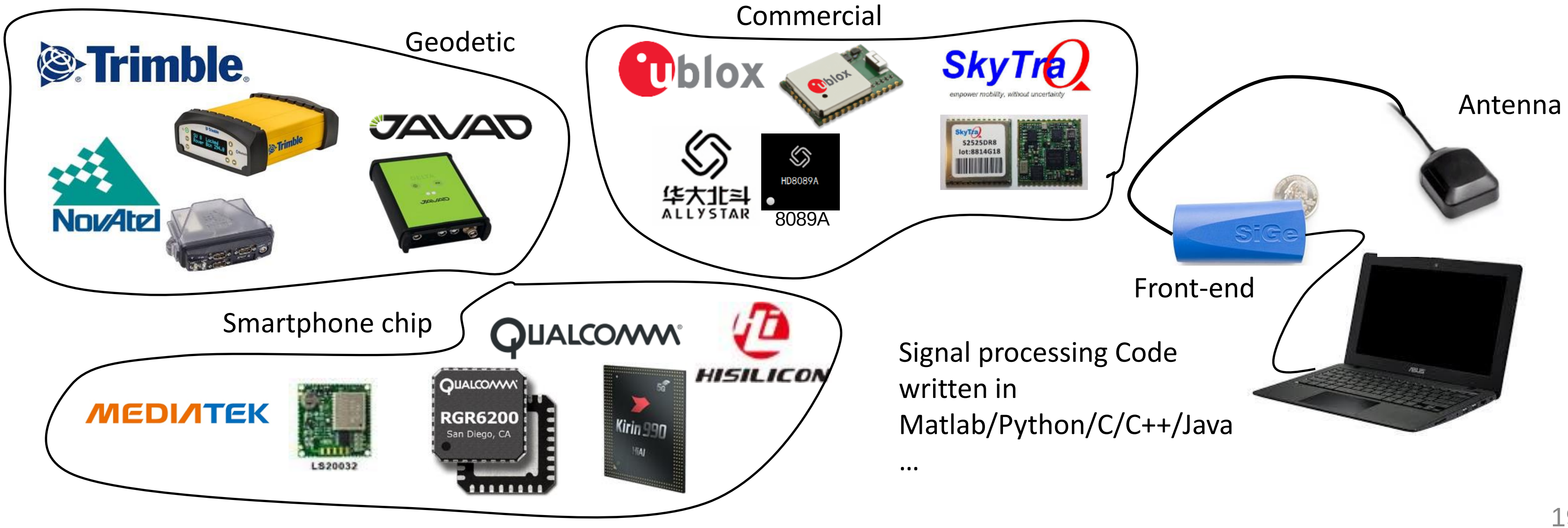
Control Segment

- Monitor stations measure signals from SVs and compute precise orbital and clock corrections data for each SV.
- Master Control station uploads orbital & clock data to SVs.

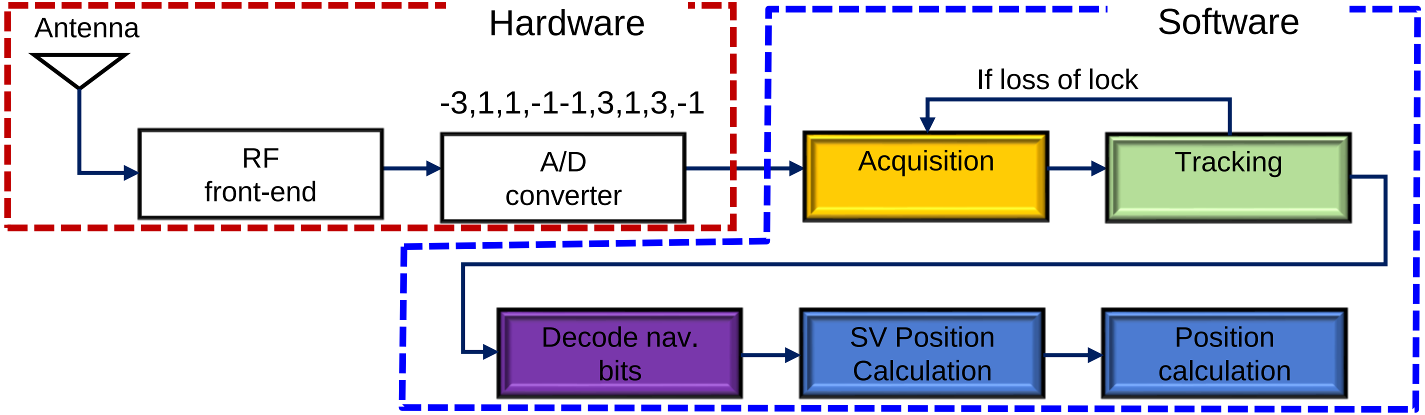


User Segment

> GPS receivers with quartz clocks can convert SV signals into position and time estimates and derive velocity.



Architecture of GPS Software Define Receiver



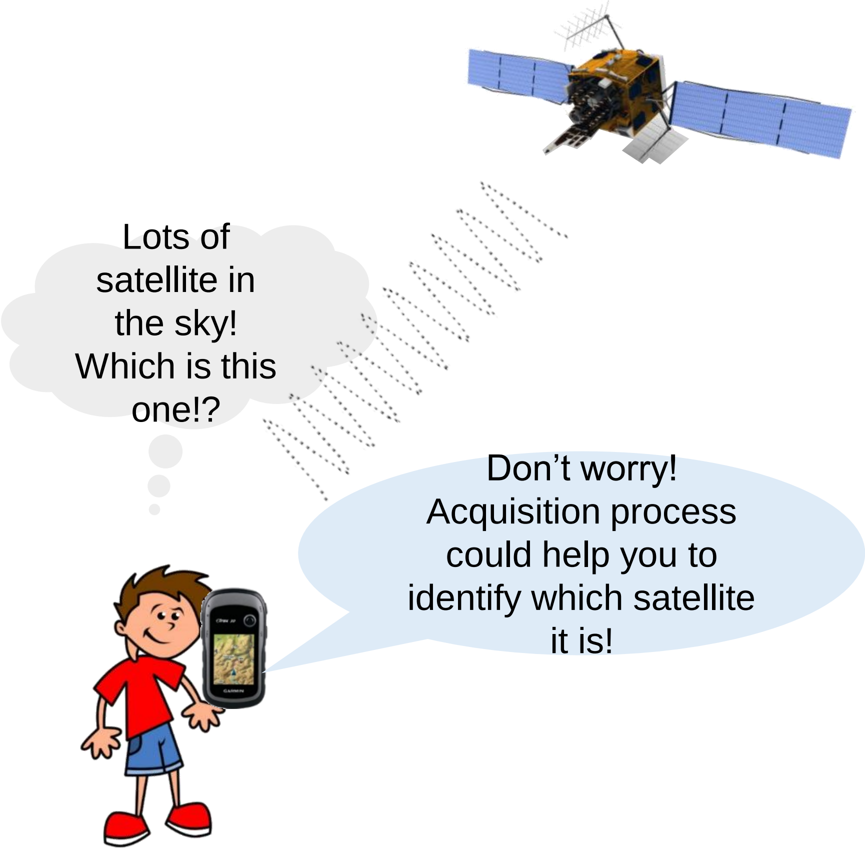
> **Acquisition:** to determine visible satellites, coarse values of carrier frequency, and code delay of received signals.

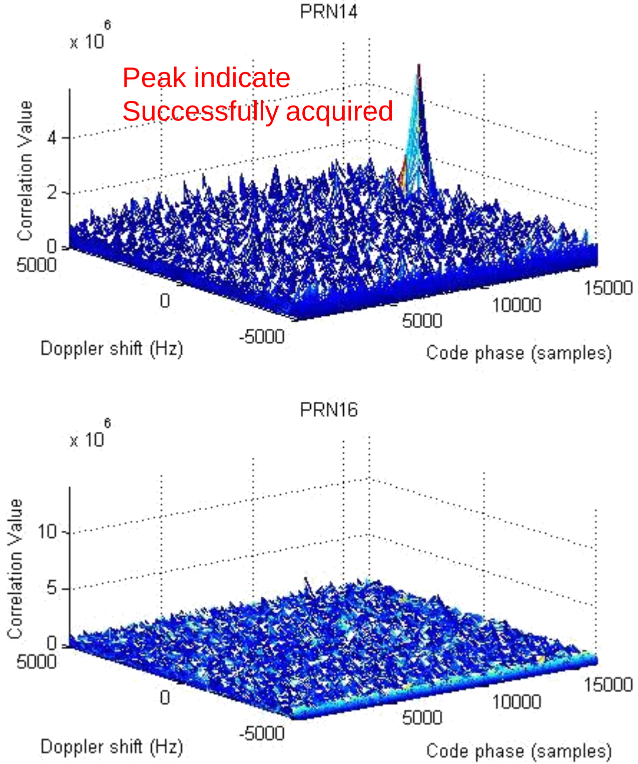
> **Tracking:** to refine these values and keep track and demodulate navigation data from satellites.

> **Navigation Data Decode:** to obtain Pseudorange, GPS time, Ephemeris, Almanac, and Klobuchar information.

> **User Positioning:** to calculate the receiver position via estimating technique.

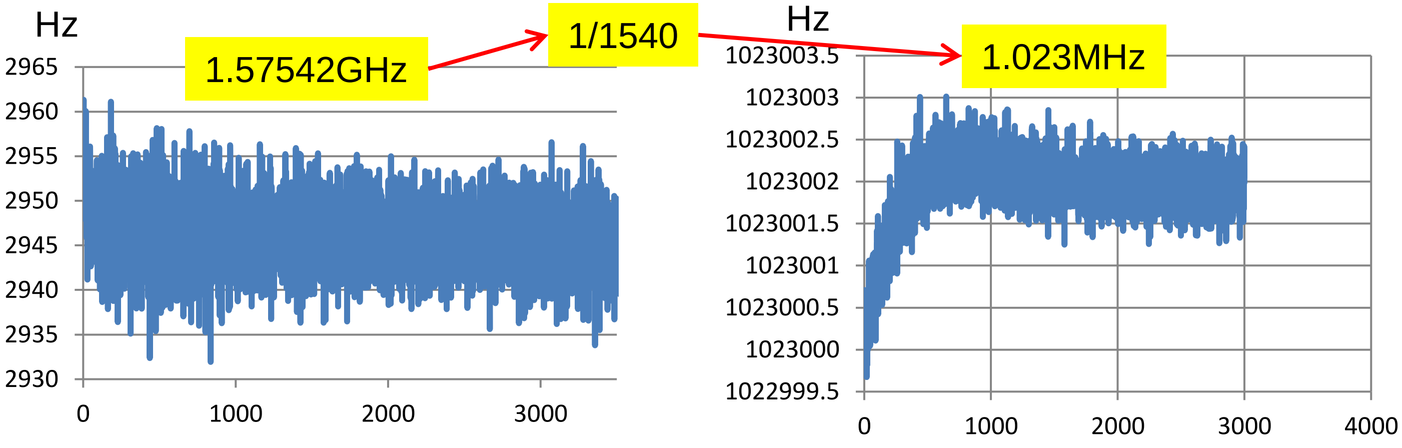
What is Signal Acquisition?





Purpose of Tracking

> Tracking is to continuously track the code-phase and Doppler frequency of GNSS signals. Loop filter is used in the tracking loop.



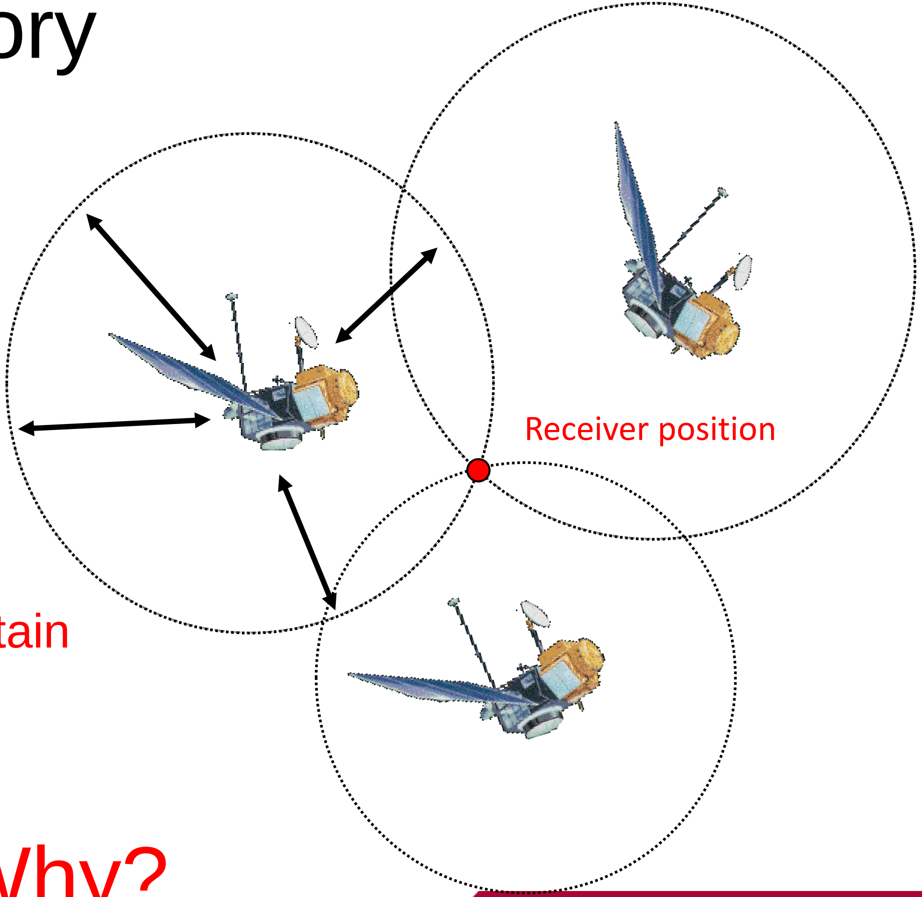
Doppler Frequency (ms)

Code Frequency (ms)

Basic principle of the GNSS

GNSS Positioning Theory

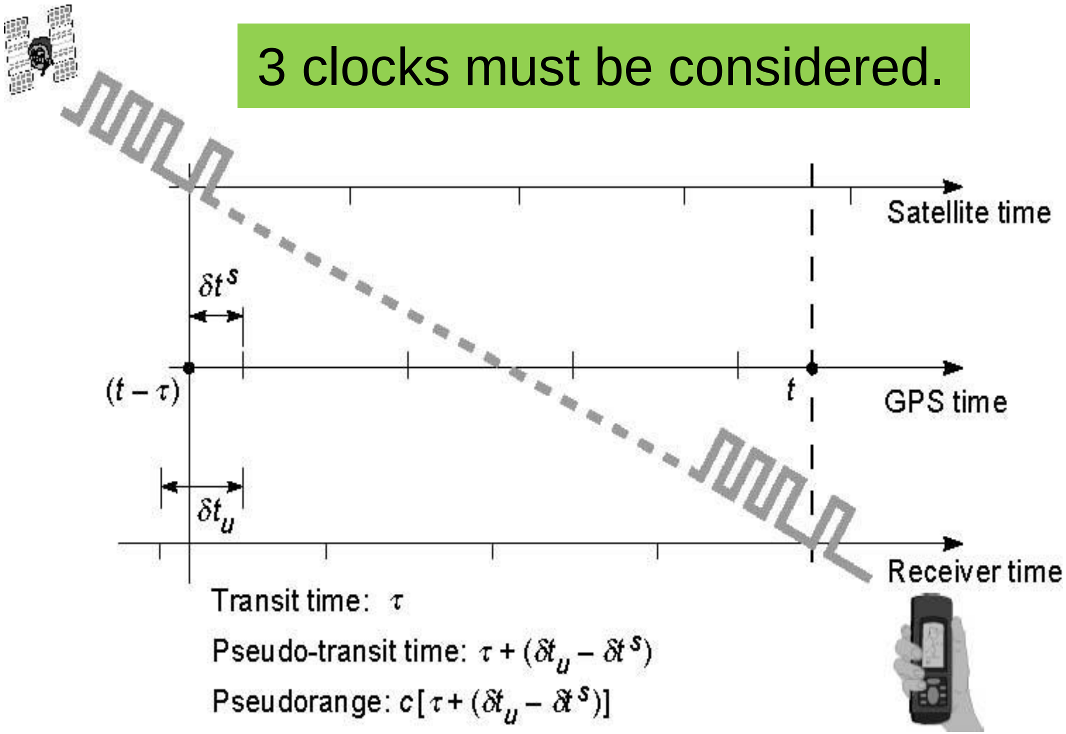
- GNSS Positioning is based on the triangulation method.
- Known information obtained from the signal processing
  - Position of satellites
  - Distance between satellites and receiver (Pseudoranges)
- How many satellites are required to obtain the 3-dimensional receiver position?
- To obtain the 3-dimensional receiver position, four satellites are required.



Why?

Pseudorange Measurement

3 clocks must be considered.



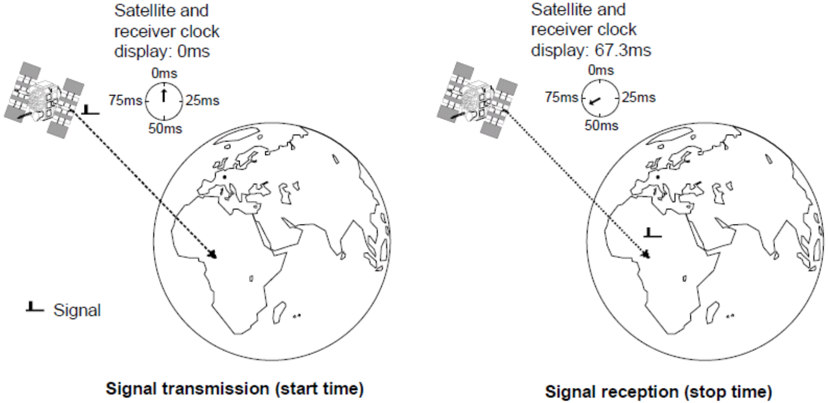
① 3 clocks are not synchronized.

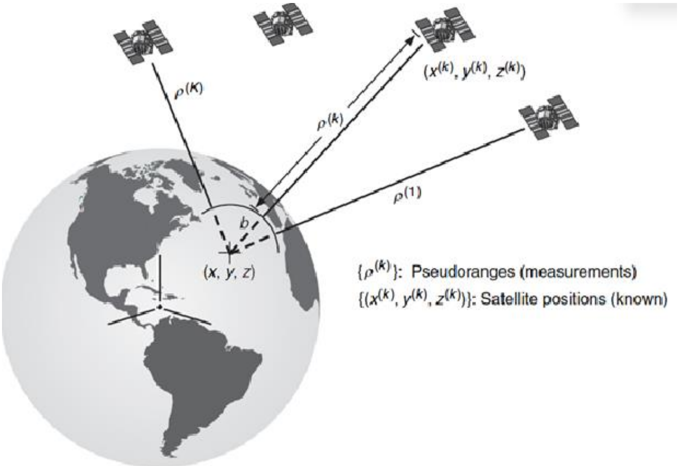
② Satellite clock error can be corrected using navigation message.

③ User clock error need to be estimated as an unknown parameter in the GNSS positioning.

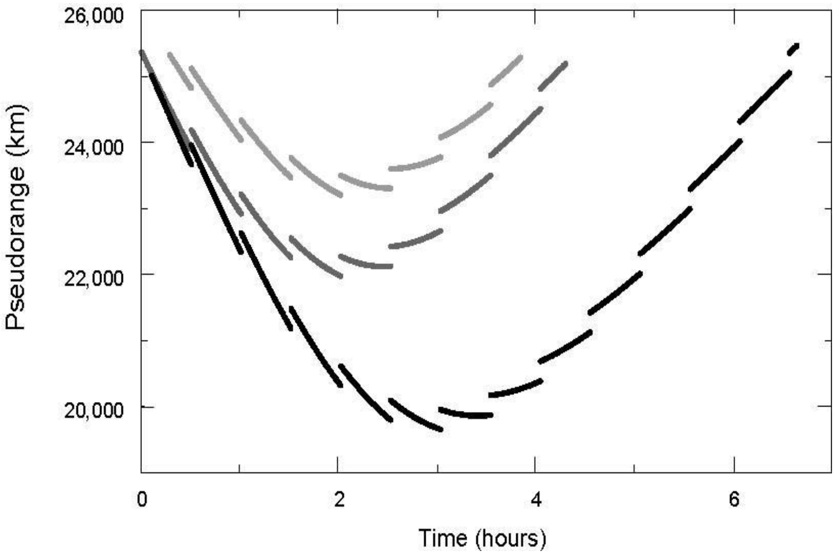
X, Y, Z, Receiver Clock Offset

- Satellite clock is corrected using navigation data.
- Fortunately, receiver clock offset is same for all satellites.
- unknown variables should be solved are X, Y, Z and receiver clock offset.
- Therefore, four satellites are required.





Real Pseudorange Measurements

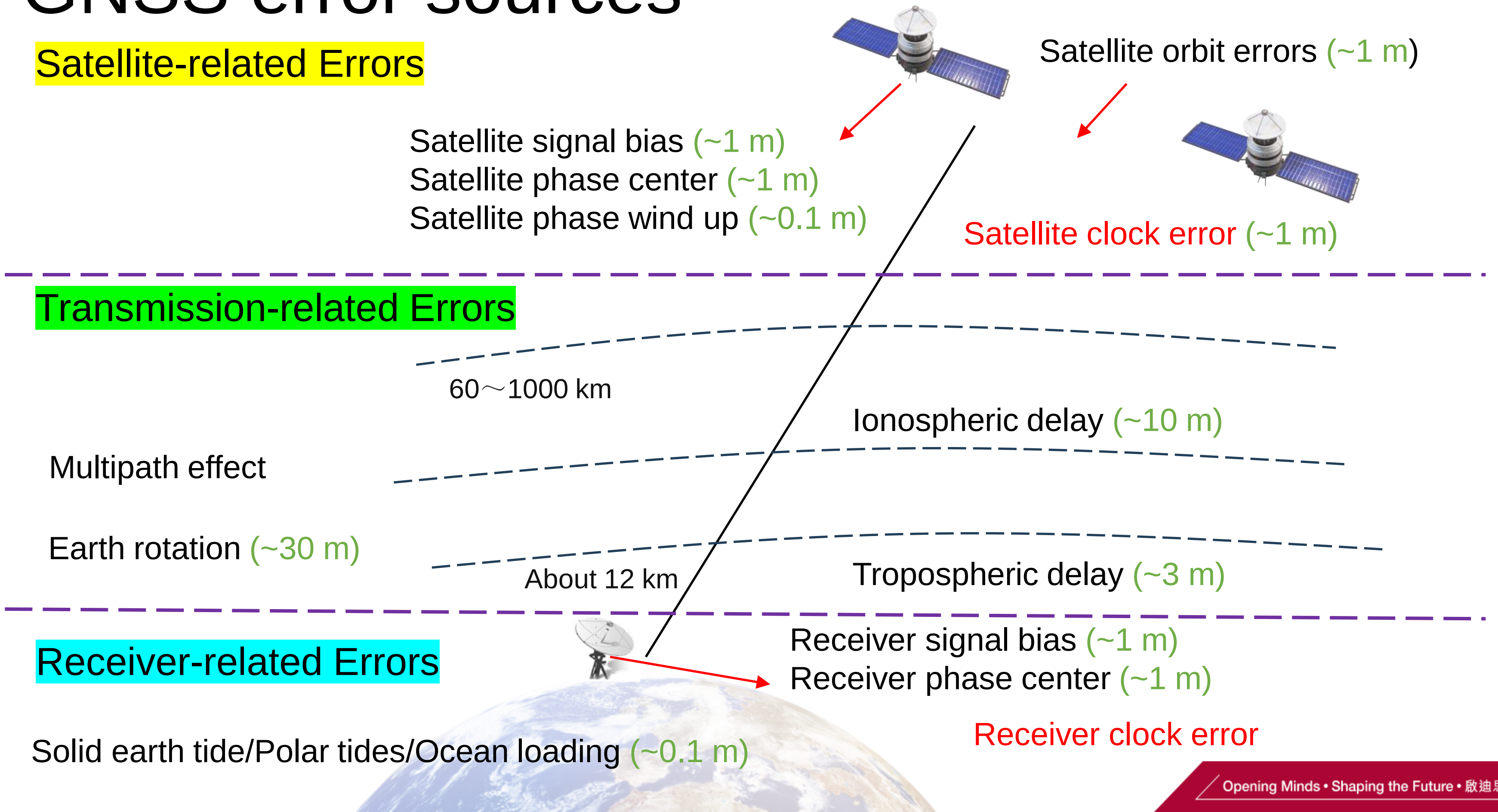


①The variations of pseudorange are mainly due to the satellite motion and earth rotation.

②Several gaps in all satellites are due to receiver clock offset.

③Receiver usually offset their own clock because the receiver clock error continues to increase.

GNSS error sources



Solve the GNSS Positioning Problem

Goal: Solve X, Y, Z!

- The pseudo-range is given by the time (τ) which takes for the satellite to the GNSS receiver multiplied by the speed of light in a vacuum. Since the clocks in the satellite and receiver are not synchronized with the GNSS system time, and there exist delays in propagation of the atmosphere, the measurement equation can be written as:

Range distance

Tropospheric Delay

Ionospheric Delay

$$P^{s}_{r,t} = \rho^{s}_{r,t} + \underbrace{c\left(\delta_{r,t} - \delta^{s}_{r,t}\right)}_{\text{Clock Errors}} + I^{s}_{r,t} + T^{s}_{r,t} + \varepsilon^{s}_{r,t}$$

Pseudorange

Multipath effect, NLOS receptions, receiver noise, etc.

Solve the GNSS Positioning Problem

Goal: Solve X, Y, Z!

Pseudorange

Receiver Clock Bias

Ionospheric Delay

Tropospheric Delay

$$P_{r,t}^{s} = \rho_{r,t}^{s} + c\left(\delta_{r,t} - \delta_{r,t}^{s}\right) + I_{r,t}^{s} + T_{r,t}^{s} + \varepsilon_{r,t}^{s}$$

$$\sqrt{(x_t^s - x_{r,t})^2 + (y_t^s - y_{r,t})^2 + (z_t^s - z_{r,t})^2}$$

Satellite Clock Bias

Multipath effects, receiver noise, etc.

Blue variables: Error source

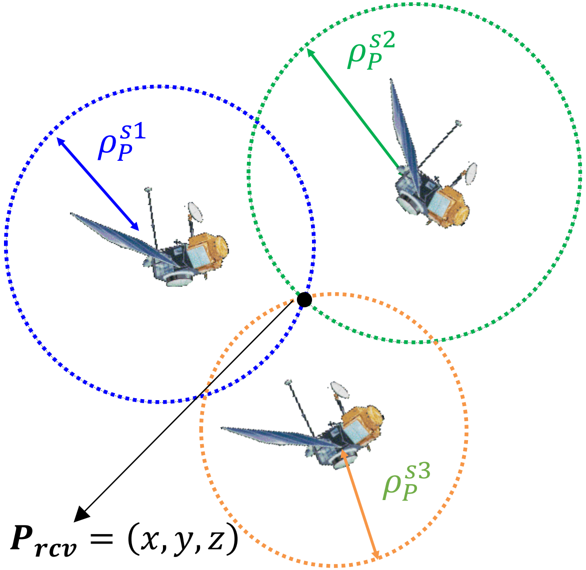
Red variables: Unknown

Black variables: Known

Satellite position $P_t^s = (x_t^s, y_t^s, z_t^s)$

Receiver position $P_{r,t} = (x_{r,t}, y_{r,t}, z_{r,t})$

Solve the GNSS Positioning Problem



Goal: Solve X, Y, Z!

Assume there are n pseudo-range measurements form n equations, these equations can be shown as：

$$\rho_{r,t}^{s1} = \sqrt{\left(x_t^{s1} - x_{r,t}\right)^2 + \left(y_t^{s1} - y_{r,t}\right)^2 + \left(z_t^{s1} - z_{r,t}\right)^2} + b$$

$$\rho_{r,t}^{s2} = \sqrt{\left(x_t^{s2} - x_{r,t}\right)^2 + \left(y_t^{s2} - y_{r,t}\right)^2 + \left(z_t^{s2} - z_{r,t}\right)^2} + b$$

$$\rho_{r,t}^{s3} = \sqrt{\left(x_t^{s3} - x_{r,t}\right)^2 + \left(y_t^{s3} - y_{r,t}\right)^2 + \left(z_t^{s3} - z_{r,t}\right)^2} + b$$

$$\vdots$$

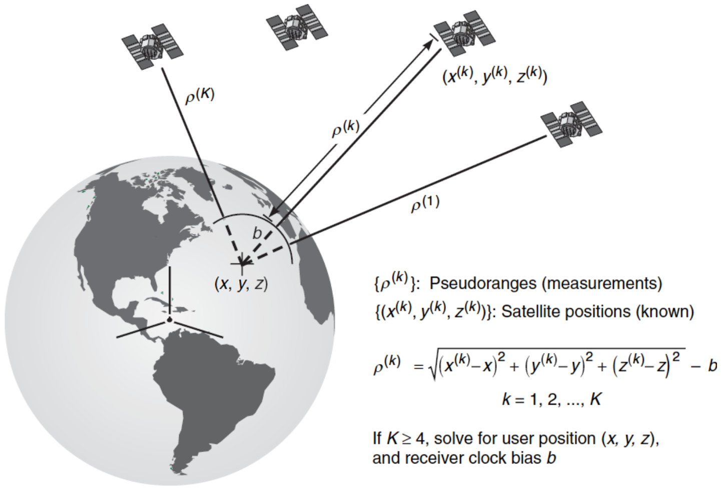
$$\rho_{r,t}^{sn} = \sqrt{\left(x_t^{sn} - x_{r,t}\right)^2 + \left(y^{sn} - y_{r,t}\right)^2 + \left(z^{sn} - z_{r,t}\right)^2} + b$$

Can we solve? How!?

YES! Mathematically, linearize the equations by Taylor Series Expansion at a point and then use Linear Least Square Estimation (LLSE) to solve the X, Y, Z.

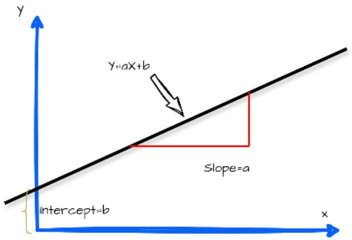
Why is LLSE needed?

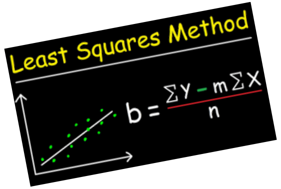
- If the number of equations is equal to the number of unknown variables, the matrix solving method can be used directly to calculate the unique solution.
- If the number of equations is greater than the number of unknown variables, LLSE can be adopted to calculate the optimal solution.

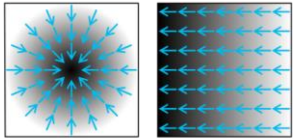


What is Linear Least Square Estimation (LLSE)

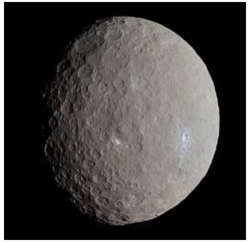
- **LLSE is the least squares approximation of linear functions to data.**
- There are two primary categories of least-squares method problems: Ordinary or linear least squares & Nonlinear least squares
- This is achieved by minimizing the sum of the squared residuals, which is the difference between the observed value and the predicted value based on the linear equation. The resulting linear equation is called the regression line or curve.





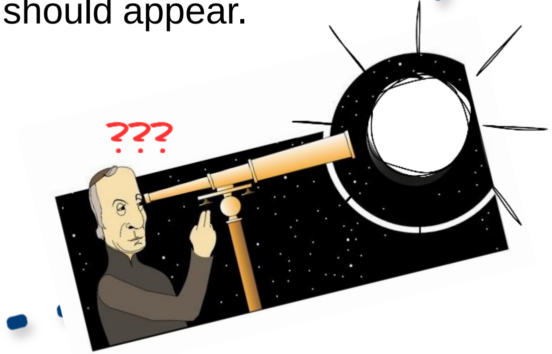


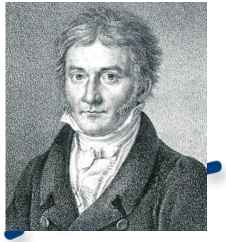
Background – Where is Ceres



1 January 1801, Italian astronomer Giuseppe Piazzi first find and named Ceres.

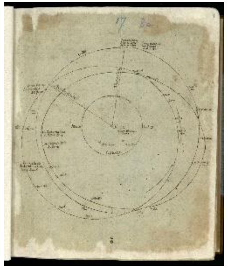
After 40 days of tracking, Piazzi lost his position as Ceres moved behind the sun, and couldn't find it when the ceres should appear.



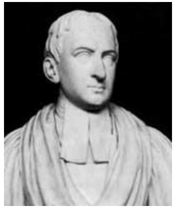


In over a month, **Gauss** developed the least square method, which can calculate the orbits of celestial bodies orbiting the sun based on a small number of observations.

The method he used was --- the **least square estimation (LSE)**



Background – Key contributors



Roger Cotes *1722*

➢ The combination of different observations as being the best estimate of the true value, errors decrease with aggregation rather than increase.

Tobias Mayer *1750*

➢ The combination of different observations taken under the same conditions contrary to simply trying one's best to observe and record a single observation accurately. The approach was known as the method of averages.

Pierre-Simon Laplace *1770s*

➢ An error minimization estimation method is defined, and Laplace specified the mathematical form of the probability density of the error and defined the method of estimating the error minimization

Assumptions of LLSE

Linearity

- The relationship between the dependent variable (response) and the independent variables (predictors) is linear, which can be expressed as:

$$Y = AX + \epsilon$$

where Y is the response vector, A is the design matrix, X is the vector of the estimated parameters.

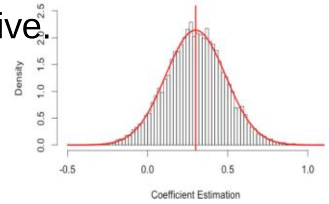
Independence

- Each observation is independent.
- This implies that the error terms ϵ are uncorrelated.
- In other words, the value of one observation does not influence or provide information about another observation.

Assumptions of LLSE

Normality of Errors

- The error terms ϵ, the differences between the actual and the predicted values, follow a normal distribution with a mean of zero.
- This means the errors are equally likely to be positive or negative.

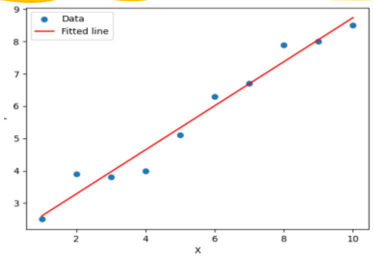


Homoscedasticity

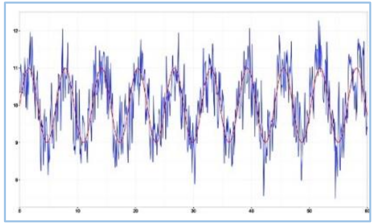
- The error terms ϵ have constant variance σ^2.
- This means that the variability in the response variable is the same across all levels of the independent variables.

Applications of LLSE

LLSE provides a way to model the relationship between variables, allowing us to understand the underlying patterns and trends in the data.

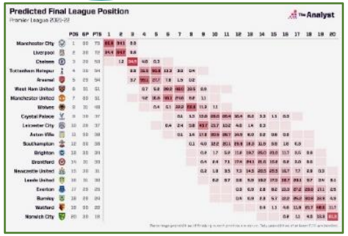


Data Fitting



In audio signal processing, LLSE can be used to smooth out noise from a recorded sound signal, making the true signal clearer.

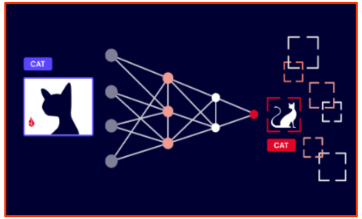
Noise filtering



By finding the best-fitting linear equation, LLSE enables us to make predictions about future values or outcomes based on the observed data.

Prediction

A deep learning frame contains two parts: training and inference. LLSE provides a theoretical framework for the training of inferring population parameters from data samples.



Inference

Example: Line Fitting use LLSE

- **Purpose**:

  Find a best fit line from a set of points which have a minimum sum of the squared residuals .

- **Input**:

  Data points' coordinate in 2-Dimension.

  $$\text{points} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

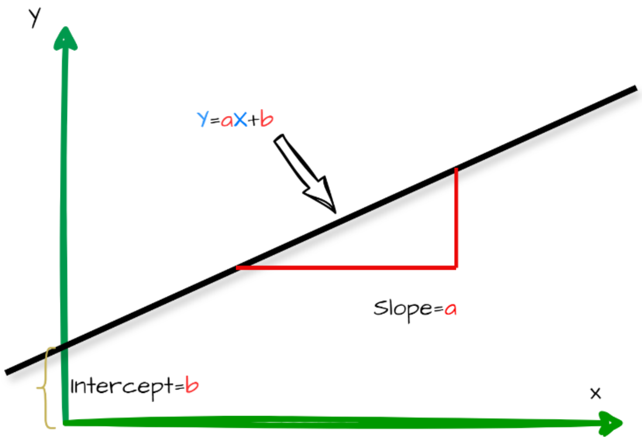
  Here we assume that all x's are independent variables, all y's are dependent ones.

- **Output**:

  Equation of best-fit line.

  $$y = a \times x + b$$

Notations:
a, b: Unknown variable (in red)
x, y: Known parameters (in blue)



Example: Line Fitting use LLSE

How to design target residual?

The given data points are to be minimized by the method of reducing residuals or offsets of each point from the line. We use the distance between target line function and data points as offsets.

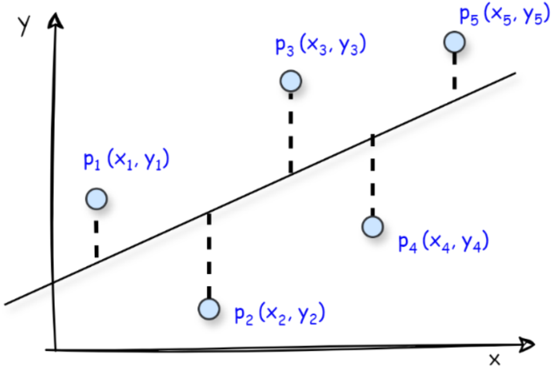
Residuals: the distance between target line function and data point

- For vertical offsets, the distance is:

$$d_i = (a * x_i + b) - y_i$$

Residuals: the distance between target line function and data point

Residual: $d_i - ((a * x_i + b) - y_i)$



vertical offsets

Example: Line Fitting use LLSE

How to build target residual?

- Here we use vertical offsets to design our residual.
- With the given five data points, we can write:

$$d_1 = (a * x_1 + b) - y_1$$
$$d_2 = (a * x_2 + b) - y_2$$
$$d_3 = (a * x_3 + b) - y_3$$
$$d_4 = (a * x_4 + b) - y_4$$
$$d_5 = (a * x_5 + b) - y_5$$

Example: Line Fitting use LLSE

How to build target residual?

- According to the property of least squares, the most suitable curve is represented by the property that the sum of squares of all deviations from a given value must be minimal, hence the residual can be built:

$$g(a,b) = residual_{vertical} = {d_1}^2 + {d_2}^2 + \cdots + {d_n}^2 = \sum_{i=0}^{n} (d_i)^2$$

Example: Line Fitting use LLSE

How to solve the problem (a and b are unknown)?

- Since our goal is to ***minimize*** the build residuals, the residual can be described as a function, The independent variables a, b are the parameters in line equation. The problem can be summarized into following function:

$$min(residual_{vertical})$$
$$= min({d_1}^2 + {d_2}^2 + \cdots + {d_n}^2) = min\sum_{i=0}^{n}(d_i)^2$$

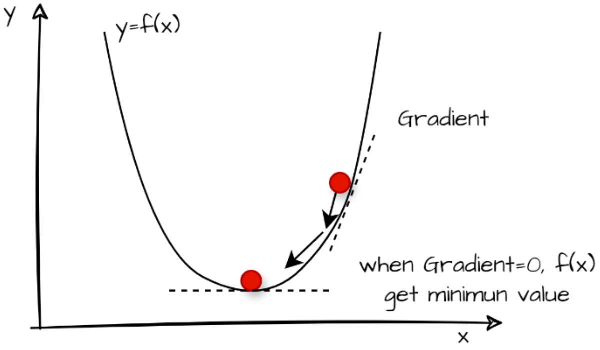
Example: Line Fitting use LLSE

How to solve the problem?

- The minimum of the sum of squares is found by setting the gradient to zero.

$$gradient: \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \dots\right)$$

- For example, In a quadratic equation with one unknown, the *gradient* equals to the *first derivative*, which represent the rate of change of y with respect to x.
- When the *gradient=0*, the function can get the minimum or the maximum value.



Example: Line Fitting use LLSE

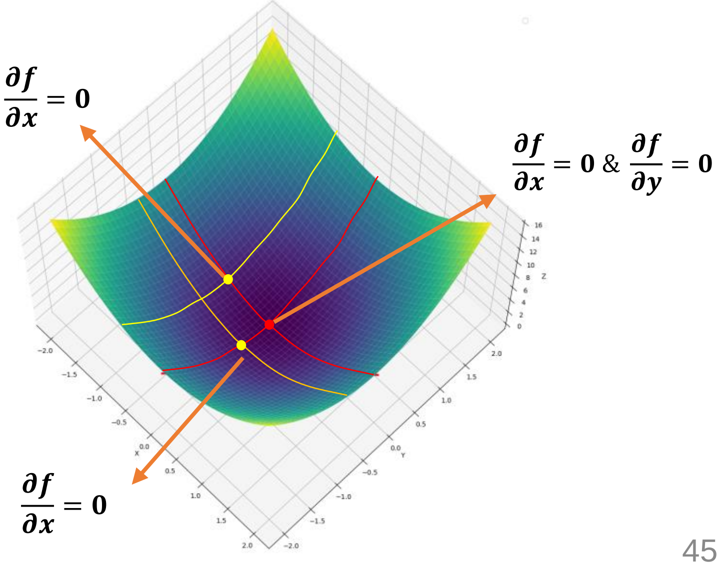
How to solve the problem?

- In the function of two variables, we need to calculate the partial derivative of each variables respectively:

$$gradient: \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

$$\frac{\partial f}{\partial x} = 0$$

$$\frac{\partial f}{\partial y} = 0$$



Example: Line Fitting use LLSE

How to solve the problem?

- In our line fitting case,

$$\begin{cases} \dfrac{\partial g(a,b)}{\partial a} = 2\sum_{i=0}^{n} d_i \dfrac{\partial d_i}{\partial a} = 2\sum_{i=0}^{n} \{[(a \times x_i + b) - y_i] \times x_i\} = 0 \\ \dfrac{\partial g(a,b)}{\partial b} = 2\sum_{i=0}^{n} d_i \dfrac{\partial d_i}{\partial b} = 2\sum_{i=0}^{n} \{[(a \times x_i + b) - y_i] \times 1\} = 0 \end{cases}$$

$$\Rightarrow y = a * x + b$$

Example: Line Fitting use LLSE

How to solve the problem?

- In our line fitting case,

$$\begin{cases} \dfrac{\partial g(a,b)}{\partial a} = 2\sum_{i=0}^{n} d_i \dfrac{\partial d_i}{\partial a} = 2\sum_{i=0}^{n}\{[(a \times x_i + b) - y_i] \times x_i\} = 0 \\ \dfrac{\partial g(a,b)}{\partial b} = 2\sum_{i=0}^{n} d_i \dfrac{\partial d_i}{\partial b} = 2\sum_{i=0}^{n}\{[(a \times x_i + b) - y_i] \times 1\} = 0 \end{cases}$$

By solve the two equation above, we can easily get the a, and b

Example: Line Fitting with Python Code

Environment Prerequisite

1. Install python3: Tutorial link.
2. Install numpy & scipy:

```
python3 -m pip install –U numpy scipy
```

3. If you are using Linux, install tkinter:

```
sudo apt-get install python3-tk
```

4. Download the example code:

```
git clone https://github.com/weisongwen/AAE4203-2425S1
```

If you have questions in install the python, please raise issue in GitHub.

Example: Line Fitting Code

Code analysis

The main function is the *fit_curve_LLSE(points)*

```
def fit_line_LLSE(points):
    # Step 1: Read sample data points (Xi,Yi), need to be converted to array (list) form.
    x = points[:, 0]
    y = points[:, 1]
    # Step 2: Set the initial value of the coefficient of the quadratic equation to be solved,
    # and the setting of the initial value will affect the convergence rate.
    initial_guess = [1, 1]
    # Step 3: Using linear least squares method to solve the coefficient of the line equation.
    # Pass in the residual and initial values.
    Para=leastsq(error_vertical,initial_guess,args=(x,y))
    # Step 4: The coefficients of the quadratic equation are obtained.
    k, b = Para[0]
    # Print the coefficients of the quadratic equation.
    print(“line function: y =",k," * x + ",b)
    # Return the coefficients of the quadratic equation for further plotting.
    return k, b
```

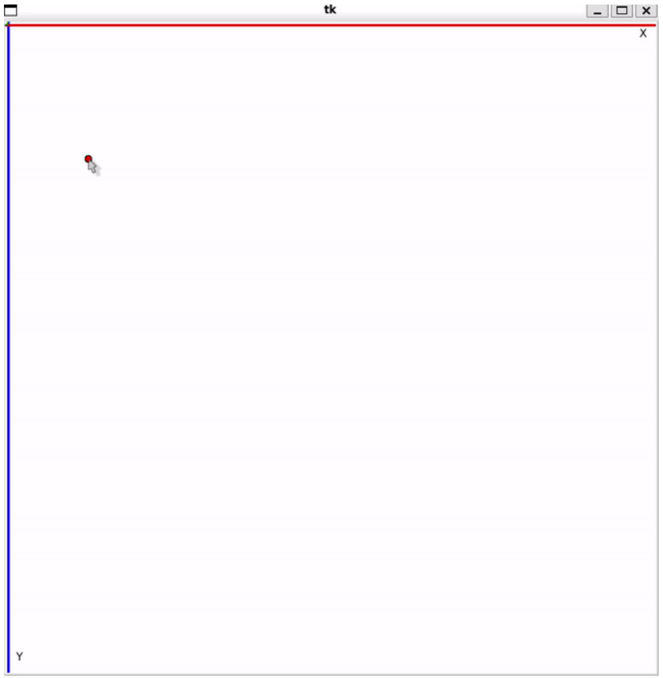
Example: Line Fitting Code

Usage

1. Enter the folder of curves fitting example code.
2. Execute the code.

```
python3 demo_line_fitting_LLSE.py
```

3. Select points in the window.



Example: Line Fitting use LSE

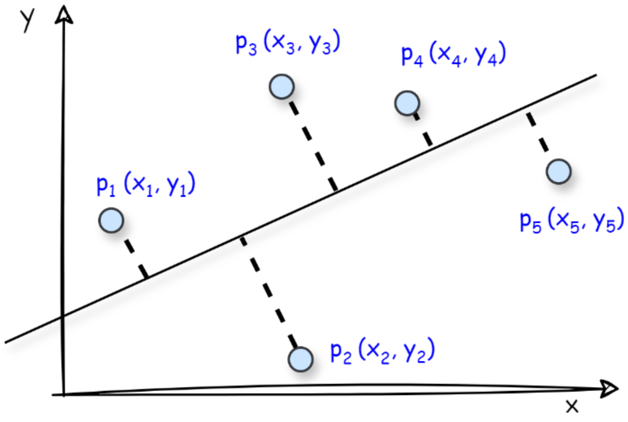
How to design target residual?

The given data points are to be minimized by the method of reducing residuals or offsets of each point from the line. We use the distance between target line function and data points as offsets.

- For perpendicular offsets, according to the Point-to-line distance formula, the distance is:

$$d_i = \frac{(a * x_i + b) - y_i}{\sqrt{a^2 + 1^2}}$$

- Here we use perpendicular offsets to design our residual. This residual is non-linear.



perpendicular offsets

Example: Line Fitting use LSE

How to build target residual?

- Here we use perpendicular offsets to design our residual. This residual is non-linear.
- because the absolute value function does not have continuous derivatives, minimizing d_n is not amenable to analytic solution. However, if the square of the perpendicular distances is minimized instead.

$$g(a,b) = residual_{perpendicular} = {d_1}^2 + {d_2}^2 + \cdots + {d_n}^2$$

$$= \sum_{i=0}^{n} (d_i)^2 = \sum_{i=0}^{n} \left(\frac{(a * x_i + b) - y_i}{\sqrt{a^2 + 1}} \right)^2$$

Example: Line Fitting use LSE

How to solve the problem (a and b are unknown)?

- Since our goal is to ***minimize*** the build residuals, the residual can be described as a function, The independent variables a, b are the parameters in line equation. The problem can be summarized into following function:

$$min\, g(a,b) = min(residual_{perpendicular})$$

$$= min({d_1}^2 + {d_2}^2 + \cdots + {d_n}^2) = min\sum_{i=0}^{n}(d_i)^2$$

Example: Line Fitting use LSE

How to solve the problem?

- In our line fitting case,

$$\begin{cases} \frac{\partial g(a,b)}{\partial a} = 2\sum_{i=0}^{n} d_i \frac{\partial d_i}{\partial a} = \frac{2}{a^2+1}\sum_{i=0}^{n}\{[(a \times x_i + b) - y_i] \times x_i\} + \sum_{i=0}^{n}\frac{[(a \times x_i + b) - y_i]^2(2a)}{(a^2+1)^2} = 0 \\ \frac{\partial g(a,b)}{\partial b} = 2\sum_{i=0}^{n} d_i \frac{\partial d_i}{\partial b} = \frac{2}{a^2+1}\sum_{i=0}^{n}\{[(a \times x_i + b) - y_i] \times 1\} = 0 \end{cases}$$

$$\Rightarrow y = a * x + b$$

For a reasonable number of noisy data points, the difference between vertical and perpendicular fits is quite small.

Example: Line Fitting Code

Code analysis

The difference between the previous line fitting code is how to design the residual: related function is the `error_perpendicular(p, x, y)`

```
# Line function: y = a * x + b
def error_ perpendicular(p, x, y):
    a = p[0]
    b = p[1]
    # perpendicular offsets
    return (a * x + b - y) / np.sqrt(a**2 + 1)
```

Then run the main function: `fit_line_LSE(points)`

```
def fit_line_LSE(points):
    x = points[:, 0]
    y = points[:, 1]
    initial_guess=[1,1]
    Para=leastsq(error_perpendicular,initial_guess,args=(x,y))
    k, b = Para[0]
    print("line function: y =",k," * x + ",b)
    return k, b
```

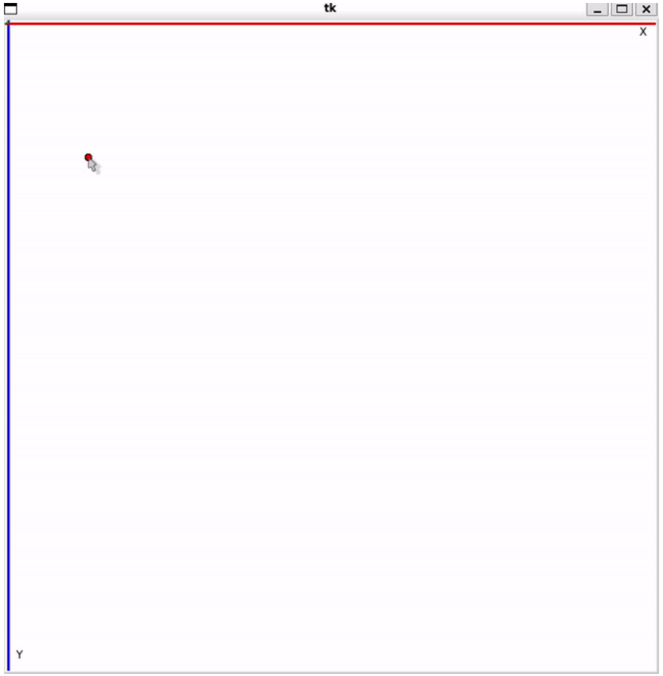
Example: Line Fitting Code

Usage

1. Enter the folder of curves fitting example code.
2. Execute the code.

```
python3 demo_line_fitting_LSE.py
```

3. Select points in the window.



Example: Curve Fitting use LSE

How to design target residual?

- Generalizing from a straight line (first degree polynomial) to a *k*th degree Polynomial:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

- Here we use vertical offsets to design our residual. This residual is given by:

$$residual = R^2 = \sum_{i=1}^{n} [(a_0 + a_1 x_i + a_2 {x_2}^2 + \cdots + a_k {x_i}^k) - y_i]^2$$

Example: Curve Fitting use LSE

How to solve the problem (a and b are unknown)?

- Since our goal is to ***minimize*** the build residuals, the residual can be described as a function, the independent variables *a*, and b are the parameters in the line equation. The problem can be summarized into the following function:

$$min(R^2) \ = min \sum_{i=1}^{n} [(a_0 + a_1 x_i + a_2 x_2{}^2 + \cdots + a_k x_i{}^k) - y_i]^2$$

Example: Curve Fitting use LSE

How to solve the problem?

- The Partial Derivatives are:

$$
\begin{cases}
\dfrac{\partial R^2}{\partial a_0} = 2\sum_{i=0}^{n}\left[\left(a_0 + a_1 x_i + a_2 x_2{}^2 + \cdots + a_k x_i{}^k\right) - y_i\right] = 0 \\
\dfrac{\partial R^2}{\partial a_1} = 2\sum_{i=0}^{n}\left[\left(a_0 + a_1 x_i + a_2 x_2{}^2 + \cdots + a_k x_i{}^k\right) - y_i\right]x = 0 \\
\qquad\qquad\qquad \dots\dots \\
\dfrac{\partial R^2}{\partial a_k} = 2\sum_{i=0}^{n}\left[\left(a_0 + a_1 x_i + a_2 x_2{}^2 + \cdots + a_k x_i{}^k\right) - y_i\right]x^k = 0
\end{cases}
$$

Example: Curve Fitting use LSE

How to solve the problem?

- These lead to the equations:

$$
\begin{cases}
a_0 n + a_1 \sum_{i=0}^{n} x_i + \cdots + a_k \sum_{i=0}^{n} {x_i}^k = \sum_{i=0}^{n} y_i \\
a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} {x_i}^2 + \cdots + a_k \sum_{i=0}^{n} {x_i}^{k+1} = \sum_{i=0}^{n} x_i y_i \\
\dots\dots \\
\dots\dots \\
a_0 \sum_{i=0}^{n} {x_i}^k + a_1 \sum_{i=0}^{n} {x_i}^{k+1} + \cdots + a_k \sum_{i=0}^{n} {x_i}^{2k} = \sum_{i=0}^{n} {x_i}^k y_i
\end{cases}
$$

Example: Curve Fitting use LSE

How to solve the problem?

- Put it in the matrix form:

$$\begin{bmatrix} n & \sum_{i=0}^{n} x_i & \cdots & \sum_{i=0}^{n} {x_i}^k \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} {x_i}^2 & \cdots & \sum_{i=0}^{n} {x_i}^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{n} {x_i}^k & \sum_{i=0}^{n} {x_i}^{k+1} & \cdots & \sum_{i=0}^{n} {x_i}^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Example: Curve Fitting use LSE

How to solve the problem?

- The matrix can be simplified as:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ {x_1}^k & {x_2}^k & \cdots & {x_n}^k \end{bmatrix} \begin{bmatrix} 1 & x_1 & \cdots & {x_1}^k \\ 1 & x_2 & \cdots & {x_2}^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & {x_n}^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ {x_1}^k & {x_2}^k & \cdots & {x_n}^k \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & x_1 & \cdots & {x_1}^k \\ 1 & x_2 & \cdots & {x_2}^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & {x_n}^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Example: Curve Fitting use LSE

How to solve the problem?

- In matrix notation:

$$A = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_1 & \cdots & {x_1}^k \\ 1 & x_2 & \cdots & {x_2}^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & {x_n}^k \end{bmatrix}, Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

The equation for a polynomial fit is given by:

$$Y = XA$$

This Matrix Equation can be solved numerically

$$X^T Y = X^T X A$$
$$A = (X^T X)^{-1} X^T Y$$

Example: Curve Fitting Code

Environment Prerequisite

1. Install python3: Tutorial link.
2. Install numpy & scipy:

```
python3 -m pip install –U numpy scipy
```

3. If you are using Linux, install tkinter:

```
sudo apt-get install python3-tk
```

4. Download the example code:

```
git clone https://github.com/weisongwen/AAE4203-2425S1
```

Example: Curve Fitting Code

Code analysis

The main function is the *fit_curve(points)*

```
def fit_curve(points):
    # Step 1: Read sample data points (Xi,Yi), need to be converted to array (list) form.
    x = points[:, 0]
    y = points[:, 1]
    # Step 2: Set the initial value of the coefficient of the quadratic equation to be solved,
    # and the setting of the initial value will affect the convergence rate.
    initial_guess = [1, 1, 1]
    # Step 3: Using least squares method to solve the coefficient of the quadratic equation.
    # Pass in the residual and initial values.
    coefficients, cov = scipy.optimize.leastsq(error_curve, initial_guess, args=(x,y))
    # Step 4: The coefficients of the quadratic equation are obtained.
    a, b, c = coefficients
    # Print the coefficients of the quadratic equation.
    print("curve function: y =",a," * x**2 + ",b ," * x + ",c)
    # Return the coefficients of the quadratic equation for further plotting.
    return a, b, c
```

Example: Curve Fitting Code

Usage

1. Enter the folder of curves fitting example code.
2. Execute the code.

```
python3 demo_curve_fitting.py
```

3. Select points in the window.
4. Click *Input points & Plot Curve* button on the right to visualize result. The curve function will be shown in the terminal.

Taylor Series Expansion

- Before using LLSE，the equations need to be linearized by Taylor Series Expansion at a point for non-linear equation.

- Ignoring second-order and higher-order error terms, the Taylor series expansion of the *n*-variable function $f(x^1, x^2, \ldots\ldots, x^n)$ at the point $(x_k^1, x_k^2, \ldots\ldots, x_k^n)$ is:

$$f(x^1, x^2, \ldots\ldots, x^n) = f(x_k^1, x_k^2, \ldots\ldots, x_k^n) + \sum_{i=1}^{n}(x^i - x_k^i) f'_{x^i}(x_k^1, x_k^2, \ldots\ldots, x_k^n) + o(n)$$

o(n) denotes the higher-order infinitesimal terms

Taylor Series Expansion

Practice:

Red variables: Unknown

Linearize the function $f(x_{r,t}, y_{r,t}, z_{r,t}) = \sqrt{(x_t^s - x_{r,t})^2 + (y_t^s - y_{r,t})^2 + (z_t^s - z_{r,t})^2} + b$ at the point (x_0, y_0, z_0) using Taylor series expansion.

$$f(x_{r,t}, y_{r,t}, z_{r,t})$$

$$\approx f(x_0, y_0, z_0) + (x_{r,t} - x_0)f'_{x_{r,t}}(x_0, y_0, z_0) + (y_{r,t} - y_0)f'_{y_{r,t}}(x_0, y_0, z_0) + (z_{r,t} - z_0)f'_{z_{r,t}}(x_0, y_0, z_0)$$

$$\approx \underline{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2} + b} + \boxed{(x_{r,t} - x_0)\frac{-(x^s - x_0)}{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}}}$$

$$+ \boxed{(y_{r,t} - y_0)\frac{-(y^s - y_0)}{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}}} + \boxed{(z_{r,t} - z_0)\frac{-(z^s - z_0)}{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}}}$$

Taylor Series Expansion

Red variables: Unknown

$$f(x_0, y_0, z_0) + (x_{r,t} - x_0) f'_{x_{r,t}}(x_0, y_0, z_0) + (y_{r,t} - y_0) f'_{y_{r,t}}(x_0, y_0, z_0) + (z_{r,t} - z_0) f'_{z_{r,t}}(x_0, y_0, z_0)$$

$$\approx \sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2} + b + (x_{r,t} - x_0) \frac{-(x^s - x_0)}{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}} + (y_{r,t} - y_0) \frac{-(y^s - y_0)}{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}} + (z_{r,t} - z_0) \frac{-(z^s - z_0)}{\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}}$$

Set $\rho_0 = \sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}$; $\Delta x = x_{r,t} - x_0$; $\Delta y = y_{r,t} - y_0$; $\Delta z = z_{r,t} - z_0$

Therefore, $f(x_{r,t}, y_{r,t}, z_{r,t}) \approx \rho_0 - \frac{(x^s - x_0)}{\rho_0} \Delta x - \frac{(y^s - y_0)}{\rho_0} \Delta y - \frac{(z^s - z_0)}{\rho_0} \Delta z + b$

Flowchart of GNSS positioning using Iterative LS

Step 1: Form the measurement function

Red variables: Unknown

- For a given pseudorange measurements $\rho_{r,t}^{s}$ which is received from satellite s at time epoch t, its measurement function can be written as follows:

$$P_{r,t}^{s} = \rho_{r,t}^{s} + c\left(\delta_{r,t} - \delta_{r,t}^{s}\right) + I_{r,t}^{s} + T_{r,t}^{s} + \varepsilon_{r,t}^{s} \quad (1)$$

- The distance between the satellite position P_t^s =(x_t^s, y_t^s, z_t^s) and receiver position $P_{r,t}$= ($x_{r,t}$, $y_{r,t}$, $z_{r,t}$) can be calculated by the distance formula, the function can be shown as

$$P_{r,t}^{s} = \sqrt{\left(x_t^s - x_{r,t}\right)^2 + \left(y_t^s - y_{r,t}\right)^2 + \left(z_t^s - z_{r,t}\right)^2} + c\delta_{r,t} + b \quad (2)$$

with b= -c$\delta_{r,t}^{s}$ $+I_{r,t}^{s} + T_{r,t}^{s}$.

Flowchart of GNSS positioning using Iterative LS

Red variables: Unknown

Step 2: Linearize the measurement function

- by using the Taylor series expansion at the approximate coordinates of the receiver position (x_0, y_0, z_0), the linearized measurement function can be obtained:

$$f(x_{r,t}, y_{r,t}, z_{r,t}) = \sqrt{(x_t^s - x_{r,t})^2 + (y_t^s - y_{r,t})^2 + (z_t^s - z_{r,t})^2} + c\delta_{r,t} + b$$

$$\approx \rho_0 - \frac{(x^s - x_0)}{\rho_0}\Delta x - \frac{(y^s - y_0)}{\rho_0}\Delta y - \frac{(z^s - z_0)}{\rho_0}\Delta z + c\delta_{r,t} + b \quad (3)$$

with ρ_0=$\sqrt{(x_t^s - x_0)^2 + (y_t^s - y_0)^2 + (z_t^s - z_0)^2}$; $\Delta x = x_{r,t} - x_0$; Δy=$y_{r,t} - y_0$; Δz=$z_{r,t} - z_0$.

Flowchart of GNSS positioning using Iterative LS

Step 3: Form the error functions

Red variables: Unknown

- Accordingly, the error equation v for a certain receiver and satellite can be written as:

$$\begin{aligned} v &= f\left(x_{r,t}, y_{r,t}, z_{r,t}\right) - P_{r,t}^{s} \\ &\approx \rho_0 - \frac{(x^s - x_0)}{\rho_0}\Delta x - \frac{(y^s - y_0)}{\rho_0}\Delta y - \frac{(z^s - z_0)}{\rho_0}\Delta z + c\delta_{r,t} + b - P_{r,t}^{s} \\ &= \left[-\frac{(x^s - x_0)}{\rho_0} \quad -\frac{(y^s - y_0)}{\rho_0} \quad -\frac{(z^s - z_0)}{\rho_0} \quad 1\right]\left[\Delta x \;\; \Delta y \;\; \Delta z \;\; c\delta_{r,t}\right]^{\mathrm{T}} - \Delta\rho \end{aligned} \quad (4)$$

with $\Delta\rho = P_{r,t}^{s} - \rho_0 - b$.

- Assume there are *n* measurements, the Eq. (4) can be expressed as:

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \frac{-(x^{s1} - x_0)}{\rho_0^1} & \frac{-(y^{s1} - y_0)}{\rho_0^1} & \frac{-(z^{s1} - z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2} - x_0)}{\rho_0^2} & \frac{-(y^{s2} - y_0)}{\rho_0^2} & \frac{-(z^{s1} - z_0)}{\rho_0^2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-(x^{sn} - x_0)}{\rho_0^n} & \frac{-(y^{sn} - y_0)}{\rho_0^n} & \frac{-(z^{sn} - z_0)}{\rho_0^n} & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \vdots \\ \Delta\rho_n \end{bmatrix}$$

$$\Big\downarrow$$

$$\boldsymbol{V} = \boldsymbol{G}\Delta\boldsymbol{p} - \Delta\boldsymbol{\rho}$$

$$\rho_{r,t}^{s1} = \sqrt{\left(x_t^{s1} - x_{r,t}\right)^2 + \left(y_t^{s1} - y_{r,t}\right)^2 + \left(z_t^{s1} - z_{r,t}\right)^2} + c\delta_{r,t} + \cdots$$

$$\rho_{r,t}^{s2} = \sqrt{\left(x_t^{s2} - x_{r,t}\right)^2 + \left(y_t^{s2} - y_{r,t}\right)^2 + \left(z_t^{s2} - z_{r,t}\right)^2} + c\delta_{r,t} + \cdots$$

$$\rho_{r,t}^{s3} = \sqrt{\left(x_t^{s3} - x_{r,t}\right)^2 + \left(y_t^{s3} - y_{r,t}\right)^2 + \left(z_t^{s3} - z_{r,t}\right)^2} + c\delta_{r,t} + \cdots$$

$$\vdots$$

$$\rho_{r,t}^{sn} = \sqrt{\left(x_t^{sn} - x_{r,t}\right)^2 + \left(y^{sn} - y_{r,t}\right)^2 + \left(z^{sn} - z_{r,t}\right)^2} + c\delta_{r,t} + \cdots$$

Flowchart of GNSS positioning using Iterative LS

Step 4: GNSS Positioning using Iterative LS

Red variables: Unknown

- Based on the principle of LS, the solution can be calculated as follows:

$$\Delta \boldsymbol{p} = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}\Delta\boldsymbol{\rho} \quad (5)$$

- Therefore, the receiver position can be obtained:

$$\boldsymbol{P}_{\mathbf{1}} = \boldsymbol{P}_{\mathbf{0}} + \Delta\boldsymbol{p} \quad (6)$$

- Since the satellite-to-earth distance is calculated using approximate coordinates, an iterative strategy is adopted to substitute the calculated coordinates $\boldsymbol{P}_{\boldsymbol{i}}$ as initial values and repeat steps 1-4 until the difference between the calculated receiver coordinates $\boldsymbol{P}_{\boldsymbol{i}+\mathbf{1}}$ and the previously calculated coordinates $\boldsymbol{P}_{\boldsymbol{i}}$ is within the threshold, thus obtaining the final receiver coordinates：

$$\boldsymbol{P}_{\boldsymbol{final}} = \boldsymbol{P}_{\boldsymbol{i}+\mathbf{1}} + \Delta\boldsymbol{p} \quad (7)$$

Flowchart of Iterative LS

Initialize $\boldsymbol{P_0} = [x_0\ y_0\ z_0]$

$\rho_0 = ||\boldsymbol{P}_0^s - \boldsymbol{P}_0||$

$\Delta\rho = P_{r,t}^s - \rho_0 - b$

$i = 0$

$$\boldsymbol{G} = \begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s1}-z_0)}{\rho_0^2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-(x^{sn}-x_0)}{\rho_0^n} & \frac{-(y^{sn}-y_0)}{\rho_0^n} & \frac{-(z^{sn}-z_0)}{\rho_0^n} & 1 \end{bmatrix}$$

$$\Delta\boldsymbol{p} = \left(\boldsymbol{G}^T\boldsymbol{G}\right)^{-1}\boldsymbol{G}^T\Delta\boldsymbol{\rho}$$

$\boldsymbol{P}_{i+1} = \boldsymbol{P}_i + \Delta\boldsymbol{p}$

$\|\Delta\boldsymbol{p}\| < 10^{-4}$

Y: $\boldsymbol{P}_{final} = \boldsymbol{P}_{i+1}$

N:

$i = i + 1$

$\rho_i = \|\boldsymbol{P}_i^s - \boldsymbol{P}_i\|$

$\Delta\rho = P_{r,t}^{sn} - \rho_i - b$

Satellite position P^s
Receiver position P
b= $-c\delta_{r,t}^s + I_{r,t}^s + T_{r,t}^s + \varepsilon_{r,t}^s$
$\Delta\boldsymbol{p}$ = $[\Delta x\ \Delta y\ \Delta z\ c\delta_{r,t}]$

Practice

(a) What is the linearization form of the observation function for the pseudorange measurement based on the first order of Taylor expansion? What is the linearization matrix **H**? (7 marks)

(b) What is the flow chart for the iteration process of the linear least square estimation to estimate the position of the GNSS receiver? (5 marks)

(c) Given the satellite position, satellite clock bias, ionospheric and tropospheric delays, and pseudorange measurement as below, what is the linearization matrix **H** given the zero initial guess of the receiver's position and receiver's clock bias? (8 marks)

Table 1 Illustration of the pseudorange measurements atmosphere error, clock bias terms, and satellite positions.

Satellite ID	Pseudorange ($\rho_{r,t}^{s}$) meters	Satellite clock bias ($c\delta_{r,t}^{s}$) meters	Ionospheric delay ($I_{r,t}^{s}$) meters	Tropospheric delay ($T_{r,t}^{s}$) meters	Satellite position ($p_{t,x}^{s}$) meters	Satellite position ($p_{t,y}^{s}$) meters	Satellite position ($p_{t,z}^{s}$) meters
10	21196662.1	198812.8	3.8639	3.24	-13186870.6	11385729.2	19672626.3
20	22222028.54	52245.17	4.6762	4.32	-7118031.6	23256076.0	-9700477.9
14	21431397.16	21575.56	3.614	3.07	-2303925.9	17164155.9	20120354.5
25	23928467.12	37173.51	5.9277	5.60	-15426414.5	2696509.3	22137570.3

Step 1: Form the measurement function

$$\begin{bmatrix} P_{r,t}^{s1} \\ P_{r,t}^{s2} \\ P_{r,t}^{s3} \\ P_{r,t}^{s4} \end{bmatrix} = \begin{bmatrix} 21196662.1 \\ 22222028.54 \\ 21431397.16 \\ 23928467.12 \end{bmatrix} \qquad P_0 = (0,0,0)^{\mathrm{T}}$$

Iteration time = 1

$$\begin{bmatrix} \rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1} \\ \rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2} \\ \rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3} \\ \rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3} \end{bmatrix} = \begin{bmatrix} \sqrt{(-13186870.6-0)^2+(11385729.2-0)^2+(19672626.3-0)^2} - 198812.8 + 3.8639 + 3.24 \\ \sqrt{(-7118031.6-0)^2+(23256076.0-0)^2+(-9700477.9-0)^2} - 52245.17 + 4.6762 + 4.32 \\ \sqrt{(-2303925.9-0)^2+(17164155.9-0)^2+(20120354.5-0)^2} - 21575.56 + 3.614 + 3.07 \\ \sqrt{(-15426414.5-0)^2+(2696509.3-0)^2+(22137570.3-0)^2} - 37173.51 + 5.9277 + 5.60 \end{bmatrix} = \begin{bmatrix} 26079333.72 \\ 26131933.01 \\ 26525464.62 \\ 27079575.38 \end{bmatrix}$$

Step 2&3: Linearize the function and form the error functions

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s2}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s3}-x_0)}{\rho_0^2} & \frac{-(y^{s3}-y_0)}{\rho_0^2} & \frac{-(z^{s3}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s4}-x_0)}{\rho_0^n} & \frac{-(y^{s4}-y_0)}{\rho_0^n} & \frac{-(z^{s4}-z_0)}{\rho_0^n} & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1}) \\ P_{r,t}^{s2} - (\rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3}) \end{bmatrix} = \underbrace{\begin{bmatrix} 0.50 & -0.43 & -0.75 & 1 \\ 0.27 & -0.89 & 0.37 & 1 \\ 0.09 & -0.65 & -0.76 & 1 \\ 0.57 & -0.10 & -0.82 & 1 \end{bmatrix}}_{\boldsymbol{G}} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \underbrace{\begin{bmatrix} -4882671.62 \\ -3909904.48 \\ -5094067.46 \\ -3151108.27 \end{bmatrix}}_{\boldsymbol{\Delta\rho}}$$

Step 4: GNSS Positioning using Iterative LS

$$\boldsymbol{\Delta p} = (\boldsymbol{G^T G})^{-1} \boldsymbol{G^T \Delta\rho} = \begin{bmatrix} -2809135.94 \\ 6332896.23 \\ 2866177.34 \\ 1416617.18 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} \qquad \boldsymbol{P_{i+1}} = \boldsymbol{P_i} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -2809135.94 \\ 6332896.22 \\ 2866177.34 \end{bmatrix}$$

$$\|[\Delta x\ \Delta y\ \Delta z]\| > 10^{-4}$$

continue

Step 1: Form the measurement function

$$\begin{bmatrix} P_{r,t}^{s1} \\ P_{r,t}^{s2} \\ P_{r,t}^{s3} \\ P_{r,t}^{s4} \end{bmatrix} = \begin{bmatrix} 21196662.1 \\ 22222028.54 \\ 21431397.16 \\ 23928467.12 \end{bmatrix} \qquad \boldsymbol{P_0} = \boldsymbol{P_1} = \begin{bmatrix} -2809135.94 \\ 6332896.22 \\ 2866177.34 \end{bmatrix}$$

Iteration time = 2

$$\begin{bmatrix} \rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1} \\ \rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2} \\ \rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3} \\ \rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3} \end{bmatrix} = \begin{bmatrix} \sqrt{(-13186870.6 - (-2809135.94))^2 + (11385729.2 - 6332896.22)^2 + (19672626.3 - 2866177.34)^2} - 198812.8 + 3.8639 + 3.24 \\ \sqrt{(-7118031.6 - (-2809135.94))^2 + (23256076.0 - 6332896.22)^2 + (-9700477.9 - 2866177.34)^2} - 52245.17 + 4.6762 + 4.32 \\ \sqrt{(-2303925.9 - (-2809135.94))^2 + (17164155.9 - 6332896.22)^2 + (20120354.5 - 2866177.34)^2} - 21575.56 + 3.614 + 3.07 \\ \sqrt{(-15426414.5 - (-2809135.94))^2 + (2696509.3 - 6332896.22)^2 + (22137570.3 - 2866177.34)^2} - 37173.51 + 5.9277 + 5.60 \end{bmatrix} = \begin{bmatrix} 20189554.32 \\ 21462442.98 \\ 20356803.30 \\ 23282478.12 \end{bmatrix}$$

Step 2&3: Linearize the function and form the error functions

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s2}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s3}-x_0)}{\rho_0^2} & \frac{-(y^{s3}-y_0)}{\rho_0^2} & \frac{-(z^{s3}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s4}-x_0)}{\rho_0^n} & \frac{-(y^{s4}-y_0)}{\rho_0^n} & \frac{-(z^{s4}-z_0)}{\rho_0^n} & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1}) \\ P_{r,t}^{s2} - (\rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3}) \end{bmatrix} = \begin{bmatrix} 0.51 & -0.25 & -0.82 & 1 \\ 0.20 & -0.79 & 0.58 & 1 \\ -0.02 & -0.53 & -0.85 & 1 \\ 0.54 & 0.16 & -0.83 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} 1007107.77 \\ 759585.55 \\ 1074593.85 \\ 645988.99 \end{bmatrix}$$

Step 4: GNSS Positioning using Iterative LS

$$\Delta \boldsymbol{p} = (\boldsymbol{G}^T \boldsymbol{G})^{-1} \boldsymbol{G}^T \Delta \boldsymbol{\rho} = \begin{bmatrix} 385044.55 \\ -927249.63 \\ -446042.86 \\ 213639.13 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} \qquad \boldsymbol{P_{i+1}} = \boldsymbol{P_i} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -2424091.38 \\ 5405646.58 \\ 2420134.47 \end{bmatrix} \qquad \|[\Delta x\ \Delta y\ \Delta z]\| > 10^{-4}$$

continue

Step 1: Form the measurement function

$$\begin{bmatrix} P_{r,t}^{s1} \\ P_{r,t}^{s2} \\ P_{r,t}^{s3} \\ P_{r,t}^{s4} \end{bmatrix} = \begin{bmatrix} 21196662.1 \\ 22222028.54 \\ 21431397.16 \\ 23928467.12 \end{bmatrix} \qquad \boldsymbol{P_0} = \boldsymbol{P_2} = \begin{bmatrix} -2424091.38 \\ 5405646.58 \\ 2420134.47 \end{bmatrix}$$

Iteration time = 3

$$\begin{bmatrix} \rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1} \\ \rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2} \\ \rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3} \\ \rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3} \end{bmatrix} = \begin{bmatrix} \sqrt{(-13186870.6-(-2424091.38))^2 + (11385729.2 - 5405646.58)^2 + (19672626.3 - 2420134.47)^2} - 198812.8 + 3.8639 + 3.24 \\ \sqrt{(-7118031.6-(-2424091.38))^2 + (23256076.0 - 5405646.58)^2 + (-9700477.9 - 2420134.47)^2} - 52245.17 + 4.6762 + 4.32 \\ \sqrt{(-2303925.9-(-2424091.38))^2 + (17164155.9 - 5405646.58)^2 + (20120354.5 - 2420134.47)^2} - 21575.56 + 3.614 + 3.07 \\ \sqrt{(-15426414.5-(-2424091.38))^2 + (2696509.3 - 5405646.58)^2 + (22137570.3 - 2420134.47)^2} - 37173.51 + 5.9277 + 5.60 \end{bmatrix} = \begin{bmatrix} 20996648.50 \\ 22028980.94 \\ 21228719.82 \\ 23736291.84 \end{bmatrix}$$

Step 2&3: Linearize the function and form the error functions

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s2}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s3}-x_0)}{\rho_0^2} & \frac{-(y^{s3}-y_0)}{\rho_0^2} & \frac{-(z^{s3}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s4}-x_0)}{\rho_0^n} & \frac{-(y^{s4}-y_0)}{\rho_0^n} & \frac{-(z^{s4}-z_0)}{\rho_0^n} & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1}) \\ P_{r,t}^{s2} - (\rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3}) \end{bmatrix} = \begin{bmatrix} 0.51 & -0.25 & -0.82 & 1.0 \\ 0.20 & -0.79 & 0.58 & 1.0 \\ -0.02 & -0.53 & -0.85 & 1.0 \\ 0.54 & 0.16 & -0.83 & 1.0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} 200013.59 \\ 193047.59 \\ 202677.33 \\ 192175.27 \end{bmatrix}$$

Step 4: GNSS Positioning using Iterative LS

$$\Delta \boldsymbol{p} = \left(\boldsymbol{G^T G}\right)^{-1} \boldsymbol{G^T} \Delta \boldsymbol{\rho} = \begin{bmatrix} 6270.73 \\ -20868.27 \\ -20868.27 \\ 181327.85 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} \qquad \boldsymbol{P_{i+1}} = \boldsymbol{P_i} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -2417820.64 \\ 5384778.31 \\ 2408323.52 \end{bmatrix}$$

$$\|[\Delta x\ \Delta y\ \Delta z]\| > 10^{-4}$$

continue

Step 1: Form the measurement function

$$\begin{bmatrix} P_{r,t}^{s1} \\ P_{r,t}^{s2} \\ P_{r,t}^{s3} \\ P_{r,t}^{s4} \end{bmatrix} = \begin{bmatrix} 21196662.1 \\ 22222028.54 \\ 21431397.16 \\ 23928467.12 \end{bmatrix} \qquad \boldsymbol{P_0} = \boldsymbol{P_3} = \begin{bmatrix} -2417820.64 \\ 5384778.31 \\ 2408323.52 \end{bmatrix}$$

Iteration time = 4

$$\begin{bmatrix} \rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1} \\ \rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2} \\ \rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3} \\ \rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3} \end{bmatrix} = \begin{bmatrix} \sqrt{(-13186870.6 - (-2417820.64))^2 + (11385729.2 - 5384778.31)^2 + (19672626.3 - 2408323.52)^2} - 198812.8 + 3.8639 + 3.24 \\ \sqrt{(-7118031.6 - (-2417820.64))^2 + (23256076.0 - 5384778.31)^2 + (-9700477.9 - 2408323.52)^2} - 52245.17 + 4.6762 + 4.32 \\ \sqrt{(-2303925.9 - (-2417820.64))^2 + (17164155.9 - 5384778.31)^2 + (20120354.5 - 2408323.52)^2} - 21575.56 + 3.614 + 3.07 \\ \sqrt{(-15426414.5 - (-2417820.64))^2 + (2696509.3 - 5384778.31)^2 + (22137570.3 - 2408323.52)^2} - 37173.51 + 5.9277 + 5.60 \end{bmatrix} = \begin{bmatrix} 21015340.49 \\ 22040711.48 \\ 21250073.03 \\ 23747149.70 \end{bmatrix}$$

Step 2&3: Linearize the function and form the error functions

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s2}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s3}-x_0)}{\rho_0^2} & \frac{-(y^{s3}-y_0)}{\rho_0^2} & \frac{-(z^{s3}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s4}-x_0)}{\rho_0^n} & \frac{-(y^{s4}-y_0)}{\rho_0^n} & \frac{-(z^{s4}-z_0)}{\rho_0^n} & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1}) \\ P_{r,t}^{s2} - (\rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3}) \end{bmatrix} = \begin{bmatrix} 0.50 & -0.28 & -0.81 & 1 \\ 0.21 & -0.80 & 0.54 & 1 \\ -0.005 & -0.55 & -0.83 & 1 \\ 0.54 & 0.11 & -0.82 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} 181321.60 \\ 181317.05 \\ 181324.12 \\ 181317.41 \end{bmatrix}$$

Step 4: GNSS Positioning using Iterative LS

$$\Delta \boldsymbol{p} = (\boldsymbol{G}^T \boldsymbol{G})^{-1} \boldsymbol{G}^T \Delta \boldsymbol{\rho} = \begin{bmatrix} 1.15 \\ -10.99 \\ -7.33 \\ 181311.94 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} \qquad \boldsymbol{P_{i+1}} = \boldsymbol{P_i} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -2417819.49 \\ 5384767.32 \\ 2408316.19 \end{bmatrix} \qquad \|[\Delta x \; \Delta y \; \Delta z]\| > 10^{-4}$$

continue

Step 1: Form the measurement function

$$\begin{bmatrix} P_{r,t}^{s1} \\ P_{r,t}^{s2} \\ P_{r,t}^{s3} \\ P_{r,t}^{s4} \end{bmatrix} = \begin{bmatrix} 21196662.1 \\ 22222028.54 \\ 21431397.16 \\ 23928467.12 \end{bmatrix} \qquad \boldsymbol{P_0} = \boldsymbol{P_4} = \begin{bmatrix} -2417819.49 \\ 5384767.32 \\ 2408316.19 \end{bmatrix}$$

Iteration time = 5

$$\begin{bmatrix} \rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1} \\ \rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2} \\ \rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3} \\ \rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3} \end{bmatrix} = \begin{bmatrix} \sqrt{(-13186870.6 - (-2417819.49))^2 + (11385729.2 - 5384767.32)^2 + (19672626.3 - 2408316.19)^2} - 198812.8 + 3.8639 + 3.24 \\ \sqrt{(-7118031.6 - (-2417819.49))^2 + (23256076.0 - 5384767.32)^2 + (-9700477.9 - 2408316.19)^2} - 52245.17 + 4.6762 + 4.32 \\ \sqrt{(-2303925.9 - (-2417819.49))^2 + (17164155.9 - 5384767.32)^2 + (20120354.5 - 2408316.19)^2} - 21575.56 + 3.614 + 3.07 \\ \sqrt{(-15426414.5 - (-2417819.49))^2 + (2696509.3 - 5384767.32)^2 + (22137570.3 - 2408316.19)^2} - 37173.51 + 5.9277 + 5.60 \end{bmatrix} = \begin{bmatrix} 21015350.15 \\ 22040716.59 \\ 21250085.21 \\ 23747155.17 \end{bmatrix}$$

Step 2&3: Linearize the function and form the error functions

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s2}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s3}-x_0)}{\rho_0^2} & \frac{-(y^{s3}-y_0)}{\rho_0^2} & \frac{-(z^{s3}-z_0)}{\rho_0^2} & 1 \\ \frac{-(x^{s4}-x_0)}{\rho_0^n} & \frac{-(y^{s4}-y_0)}{\rho_0^n} & \frac{-(z^{s4}-z_0)}{\rho_0^n} & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s1} + I_{r,t}^{s1} + T_{r,t}^{s1}) \\ P_{r,t}^{s2} - (\rho_0^{s2} - c\delta_{r,t}^{s2} + I_{r,t}^{s2} + T_{r,t}^{s2}) \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s3} + I_{r,t}^{s3} + T_{r,t}^{s3} \\ P_{r,t}^{s1} - (\rho_0^{s1} - c\delta_{r,t}^{s4} + I_{r,t}^{s3} + T_{r,t}^{s3}) \end{bmatrix} = \begin{bmatrix} 0.50 & -0.28 & -0.81 & 1 \\ 0.21 & -0.80 & 0.54 & 1 \\ -0.005 & -0.55 & -0.83 & 1 \\ 0.54 & 0.11 & -0.82 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} - \begin{bmatrix} 181321.60 \\ 181317.05 \\ 181324.12 \\ 181317.41 \end{bmatrix}$$

Step 4: GNSS Positioning using Iterative LS

$$\Delta \boldsymbol{p} = (\boldsymbol{G^T G})^{-1} \boldsymbol{G^T} \Delta \boldsymbol{\rho} = \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \\ 181311.94 \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_{r,t} \end{bmatrix} \qquad \boldsymbol{P_{i+1}} = \boldsymbol{P_i} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -2417819.49 \\ 5384767.32 \\ 2408316.19 \end{bmatrix} \qquad \|[\Delta x\ \Delta y\ \Delta z]\| < 10^{-4}$$

end

Practice

Code implementation:

1. Download the example code: Code

   The code is in the ~/Sample_Codes/gnss_position/LLSE_GNSS.py

2. How to use it:

   On Windows, run it directly

   In Linux, run "python3 LLSE_GNSS.py" in the terminal

Practice

Code analysis :

1. Import data and set the initial value

```
# Import the satellite position Unit: meter
satellite_positions = np.array([
    [-13186870.6, 11385729.2, 19672626.3],
    [-7118031.6, 23256076.0, -9700477.9],
    [-2303925.9, 17164155.9, 20120354.5],
    [-15426414.5, 2696509.3, 22137570.3]
])
# Import the pseudoranges measurement Unit: meter
pseudoranges_meas = np.array([21196662.1, 22222028.54, 21431397.16, 23928467.12])
# Import the satellite clock bias(δ^s_{r,t}) Unit: meter
satellite_clock_bias = np.array([198812.8, 52245.17, 21575.56, 37173.51])
# Import the ionospheric delay Unit: meter
ionospheric_delay = np.array([3.8639, 4.6762, 3.614, 5.9277])
# Import the tropospheric delay Unit: meter
tropospheric_delay = np.array([3.24, 4.32, 3.07, 5.60])
# Set initial receiver position
receiver_position = np.array([0.0, 0.0, 0.0])
```

Practice

Code analysis :

2. The receiver position is obtained by least square method function

```python
"""Calculate solution of receiver position by least squares, iterate a maximum of 20 times until the condition is met
Parameters:
satellite_positions - A three-dimensional array of satellite positions
receiver_position - The receiver positions
satellite_clock_bias - The value of the satellite clock bias
ionospheric_delay - The value of the ionospheric delay
tropospheric_delay - The value of tropospheric delay """
def least_squares_solution(satellite_positions, receiver_position, pseudoranges_meas,satellite_clock_bias, 
ionospheric_delay, tropospheric_delay):
# iterate a maximum of 20 times until the condition is met
  for j in range(20):
    # Calculate the pseudorange
    estimated_distances = np.linalg.norm(satellite_positions - receiver_position, axis=1)
    pseudoranges = estimated_distances-satellite_clock_bias+ionospheric_delay+tropospheric_delay
    # Calculate the difference between the measured pseudorange and the calculated pseudorange
    pseudoranges_diff = pseudoranges_meas – pseudoranges


……
```

estimated_distances: $\rho_i = \|\boldsymbol{P}_i^s - \boldsymbol{P}_i\|$

pseudoranges_diff: $\Delta\rho = P_{r,t}^s - \rho_i - b$

$= P_{r,t}^s - \rho_i + c\delta_{r,t}^s - I_{r,t}^s - T_{r,t}^s$

Practice

Code analysis :

2. The receiver position is obtained by least square method function

```
def least_squares_solution(satellite_positions, receiver_position, pseudoranges_meas,satellite_clock_bias,
ionospheric_delay, tropospheric_delay):
    for j in range(20):
    ......
    # Calculate the matrix G
        G = np.zeros((len(satellite_positions), 4))
        for i in range(len(satellite_positions)):
            p_i = satellite_positions[i] - receiver_position
            r_i = np.linalg.norm(p_i)
            G[i, :3] = -p_i / r_i
            G[i, 3] = 1.0

    # Solve using least square method
        #delta_p = np.linalg.inv(G.T @ G) @ G.T @ pseudoranges_diff
        delta_p = np.linalg.lstsq(G, pseudoranges_diff, rcond=None)[0]
        receiver_position += delta_p[:3]

        if np.linalg.norm(delta_p[:3]) < 1e-4:
            break
    return receiver_position
```

$$G = \begin{bmatrix} \frac{(x^{s1}-x_i)}{\rho_i^1} & \frac{(y^{s1}-y_i)}{\rho_i^1} & \frac{(z^{s1}-z_i)}{\rho_i^1} & 1 \\ \frac{(x^{s2}-x_i)}{\rho_i^2} & \frac{(y^{s2}-y_i)}{\rho_i^2} & \frac{(z^{s2}-z_i)}{\rho_i^2} & 1 \\ \frac{(x^{s3}-x_i)}{\rho_i^3} & \frac{(y^{s3}-y_i)}{\rho_i^3} & \frac{(z^{s3}-z_i)}{\rho_i^3} & 1 \\ \frac{(x^{s4}-x_i)}{\rho_i^4} & \frac{(y^{s4}-y_i)}{\rho_i^4} & \frac{(z^{s4}-z_i)}{\rho_i^4} & 1 \end{bmatrix}$$

$$\Delta \boldsymbol{p} = \left(\boldsymbol{G}^T \boldsymbol{G}\right)^{-1} \boldsymbol{G}^T \Delta \boldsymbol{\rho}$$

$$\|[\Delta x\ \Delta y\ \Delta z]\| < 10^{-4}$$

Practice

Code analysis :

3. After data import and function definition, use the function

estimated_position = least_squares_solution(satellite_positions, receiver_position, pseudoranges_meas,satellite_clock_bias, ionospheric_delay, tropospheric_delay)

Run the LLSE_GNSS.py, can get the result:

```
Iteration time =  4
estimated_distances =  [21214155.85449962 22092952.77220106 21271654.09439831 23784317.16070073]
pseudoranges_diff =  [181311.94160039 181311.94159894 181311.94160169 181311.94159926]
G =  [[ 0.50763515 -0.28287536 -0.81381085  1.        ]
 [ 0.21274712 -0.80891445  0.54808401  1.        ]
 [-0.00535424 -0.55375988 -0.83265919  1.        ]
 [ 0.54694002  0.1130265  -0.82950685  1.        ]]
delta_p =  [-9.52671809e-07 -2.83129299e-06 -2.36111283e-06]
Estimated Receiver Position: [-2417819.49115683  5384767.32183912  2408316.19588408]
Estimated Receiver Clock Bias: 181311.9415981472
```

This is the GNSS position result obtained by the least square method.

Receiver position

$P_{r,t} = (x_{r,t}, y_{r,t}, z_{r,t})$

Receiver Clock Bias $c\delta_{r,t}$

Q&A

Thank you for your attention ☺

Q&A

Dr. Weisong Wen

If you have any questions or inquiries, please feel free to contact me.

Email: welson.wen@polyu.edu.hk

Linear Least Square Estimation Formulation via Matrix

Matrix form of least squares: $\boldsymbol{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$

Taking 5 pieces of data as an example, the linear regression model constructed is as follows:

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \epsilon_2 \\
y_3 &= \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \epsilon_3 \\
y_4 &= \beta_0 + \beta_1 x_{41} + \beta_2 x_{42} + \epsilon_4 \\
y_5 &= \beta_0 + \beta_1 x_{51} + \beta_2 x_{52} + \epsilon_5
\end{aligned}
$$

In matrix form: $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$

$$
\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ 1 & x_{41} & x_{42} \\ 1 & x_{51} & x_{52} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix}
$$

Linear Least Square Estimation Formulation via Matrix

The idea of the least squares method is to find $\boldsymbol{\beta}$ such that the sum of the squared errors $\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon}$ is minimized.

$$\min_{\boldsymbol{\beta}} \boldsymbol{\epsilon}^\top\boldsymbol{\epsilon}$$

$$\begin{aligned}
\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon} &= (\boldsymbol{y}-\boldsymbol{X\beta})^T(\boldsymbol{y}-\boldsymbol{X\beta}) \\
&= (\boldsymbol{y}-\boldsymbol{X\beta})^T\boldsymbol{y}-(\boldsymbol{y}-\boldsymbol{X\beta})^T\boldsymbol{X\beta} \\
&= \boldsymbol{y}^\top\boldsymbol{y}-(\boldsymbol{X\beta})^T\boldsymbol{y}-\boldsymbol{y}^\top\boldsymbol{X\beta}+(\boldsymbol{X\beta})^T\boldsymbol{X\beta} \\
&= \boldsymbol{y}^\top\boldsymbol{y}-\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y}-\boldsymbol{y}^\top\boldsymbol{X\beta}+\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X\beta}
\end{aligned}$$

The minimum point required for least squares is usually at the point where the partial derivative is 0, so

$$\frac{\partial(\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon})}{\partial\boldsymbol{\beta}}=\frac{\partial(\boldsymbol{y}^\top\boldsymbol{y})}{\partial\boldsymbol{\beta}}-\frac{\partial(\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y})}{\partial\boldsymbol{\beta}}-\frac{\partial(\boldsymbol{y}^\top\boldsymbol{X\beta})}{\partial\boldsymbol{\beta}}+\frac{\partial(\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X\beta})}{\partial\boldsymbol{\beta}}=0$$

Linear Least Square Estimation of Matrix

$$\frac{\partial(\boldsymbol{x}^\top \boldsymbol{a})}{\partial \boldsymbol{x}} = \frac{\partial(\boldsymbol{a}^\top \boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{a}$$

Prove:

$$\frac{\partial(\boldsymbol{x}^\top \boldsymbol{a})}{\partial \boldsymbol{x}} = \frac{\partial(\boldsymbol{a}^\top \boldsymbol{x})}{\partial \boldsymbol{x}}$$

$$= \frac{\partial(a_1x_1 + a_2x_2 + \cdots + a_nx_n)}{\partial \boldsymbol{x}}$$

$$= \begin{bmatrix} \frac{\partial(a_1x_1 + a_2x_2 + \cdots + a_nx_n)}{\partial x_1} \\ \frac{\partial(a_1x_1 + a_2x_2 + \cdots + a_nx_n)}{\partial x_2} \\ \vdots \\ \frac{\partial(a_1x_1 + a_2x_2 + \cdots + a_nx_n)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \boldsymbol{a}$$

Linear Least Square Estimation of Matrix

$$\frac{\partial(\boldsymbol{x}^\top\boldsymbol{x})}{\partial\boldsymbol{x}} = 2\boldsymbol{x}$$

Prove: $$\frac{\partial(\boldsymbol{x}^\top\boldsymbol{x})}{\partial\boldsymbol{x}} = \frac{\partial(x_1^2+x_2^2+\cdots+x_n^2)}{\partial\boldsymbol{x}}$$

$$= \begin{bmatrix} \frac{\partial(x_1^2+x_2^2+\cdots+x_n^2)}{\partial x_1} \\ \frac{\partial(x_1^2+x_2^2+\cdots+x_n^2)}{\partial x_2} \\ \vdots \\ \frac{\partial(x_1^2+x_2^2+\cdots+x_n^2)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \vdots \\ 2x_n \end{bmatrix} = 2\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = 2\boldsymbol{x}$$

Linear Least Square Estimation of Matrix

$$\frac{\partial(\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^\top \boldsymbol{x}$$

Prove:

$$\frac{\partial(\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial\begin{pmatrix} a_{11}x_1x_1 + a_{12}x_1x_2 + \cdots + a_{1n}x_1x_n \\ +a_{21}x_2x_1 + a_{22}x_2x_2 + \cdots + a_{2n}x_2x_n \\ +a_{n1}x_nx_1 + a_{n2}x_nx_2 + \cdots + a_{nn}x_nx_n \end{pmatrix}}{\partial \boldsymbol{x}}$$

$$= \begin{bmatrix} (a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n) + (a_{11}x_1 + a_{21}x_2 + \cdots + a_{n1}x_n) \\ (a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n) + (a_{12}x_1 + a_{22}x_2 + \cdots + a_{n2}x_n) \\ \vdots \\ (a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n) + (a_{1n}x_1 + a_{2n}x_2 + \cdots + a_{nn}x_n) \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \end{bmatrix} + \begin{bmatrix} a_{11}x_1 + a_{21}x_2 + \cdots + a_{n1}x_n \\ a_{12}x_1 + a_{22}x_2 + \cdots + a_{n2}x_n \\ \vdots \\ a_{1n}x_1 + a_{2n}x_2 + \cdots + a_{nn}x_n \end{bmatrix}$$

Linear Least Square Estimation of Matrix

$$\frac{\partial(\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^\top \boldsymbol{x}$$

Prove:

$$\frac{\partial(\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial\begin{pmatrix} a_{11}x_1x_1 + a_{12}x_1x_2 + \cdots + a_{1n}x_1x_n \\ +a_{21}x_2x_1 + a_{22}x_2x_2 + \cdots + a_{2n}x_2x_n \\ +a_{n1}x_nx_1 + a_{n2}x_nx_2 + \cdots + a_{nn}x_nx_n \end{pmatrix}}{\partial \boldsymbol{x}}$$

$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \end{bmatrix} + \begin{bmatrix} a_{11}x_1 + a_{21}x_2 + \cdots + a_{n1}x_n \\ a_{12}x_1 + a_{22}x_2 + \cdots + a_{n2}x_n \\ \vdots \\ a_{1n}x_1 + a_{2n}x_2 + \cdots + a_{nn}x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^\top \boldsymbol{x}$$

Linear Least Square Estimation of Matrix

$$\frac{\partial(\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon})}{\partial\boldsymbol{\beta}} = \frac{\partial(\boldsymbol{y}^\top\boldsymbol{y})}{\partial\boldsymbol{\beta}} - \frac{\partial(\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y})}{\partial\boldsymbol{\beta}} - \frac{\partial(\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} + \frac{\partial(\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \mathbf{0}$$

$$= 0 - \boldsymbol{X}^\top\boldsymbol{y} - \boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta}$$

$$= 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta} - 2\boldsymbol{X}^\top\boldsymbol{y} \quad = \mathbf{0}$$

$$\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^\top\boldsymbol{y}$$

$$\boldsymbol{\beta} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

Tip: $\frac{\partial(\boldsymbol{y}^\top\boldsymbol{y})}{\partial\boldsymbol{\beta}} = 0$ $\quad$ $\frac{\partial(\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y})}{\partial\boldsymbol{\beta}} = \boldsymbol{X}^\top\boldsymbol{y}$ $\quad$ $\frac{\partial(\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = (\boldsymbol{y}^\top\boldsymbol{X})^\top = \boldsymbol{X}^\top\boldsymbol{y}$

$$\frac{\partial(\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta} + (\boldsymbol{X}^\top\boldsymbol{X})^\top\boldsymbol{\beta} = 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta}$$

Visualization of GNSS Positioning using LLSE

$\boldsymbol{V} = \boldsymbol{G}\Delta\boldsymbol{p} - \Delta\boldsymbol{\rho}$

According to the Least Squares

$$\Delta\boldsymbol{p} = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}\Delta\boldsymbol{\rho}$$

$$\boldsymbol{P} = \boldsymbol{P}_{\mathbf{0}} + \Delta\boldsymbol{p}$$

$\Delta\boldsymbol{p}$=[$\Delta x\ \Delta y\ \Delta z\ c\delta_{r,t}$]

$\boldsymbol{V} = [v_1, v_2, \ldots\ldots, v_n]$

$\Delta\boldsymbol{\rho}$= $[\Delta\rho_1, \Delta\rho_2, \ldots\ldots, \Delta\rho_n]$

$$\boldsymbol{G}=\begin{bmatrix} \frac{-(x^{s1}-x_0)}{\rho_0^1} & \frac{-(y^{s1}-y_0)}{\rho_0^1} & \frac{-(z^{s1}-z_0)}{\rho_0^1} & 1 \\ \frac{-(x^{s2}-x_0)}{\rho_0^2} & \frac{-(y^{s2}-y_0)}{\rho_0^2} & \frac{-(z^{s1}-z_0)}{\rho_0^2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-(x^{sn}-x_0)}{\rho_0^n} & \frac{-(y^{sn}-y_0)}{\rho_0^n} & \frac{-(z^{sn}-z_0)}{\rho_0^n} & 1 \end{bmatrix}$$

$\boldsymbol{P}_{\mathbf{0}}$=[$x_0\ y_0\ z_0$]