Local Cartesian coordinate conversion based on Matlab

Mu Junjie,Deng Jianwei,Su Jiayi,Group 09

Abstract：In this experiment, we used matlab software to convert local Cartesian coordinates into ITRF coordinates, then converted ITRF coordinates into ETRF coordinates, and finally calibrated the corresponding unit longitude, latitude, and altitude.

Key words：MATLAB；Local Cartesian coordinate；ITRF；ETRF；

1.Introduction

A container ship is sailing in Genova's port. The ship is equipped with a GNSS receiver, which position is known in ITRF with a certain level of accuracy. The ship has a length of 300 meters and a width of 60. Known the considerable dimensions, a body reference frame is available for navigation purposes: it's origin O is known and placed in correspondence of the GNSS receiver. The X axis is oriented in the motion direction, Z axis is perpendicular to the ship plane and in the up direction, Y axis is oriented to complete the righthanded triad. The position of three points A,B,C of the ship is provided in the body frame.

2.Research Data

INPUT DATA:

- Origin *O* in ITRF

$$\underline{X}_{O\ (ITRF)} = \begin{bmatrix} 44°\ 23'\ 24.000" \\ 8°\ 56'\ 20.000" \\ 70.00\ m \end{bmatrix} \quad \sigma_0 = 10cm$$

- *A, B, C* in body frame (Local Level) with respect to the origin

$$\underline{X}_{A\ (LL)} = \begin{bmatrix} 0 \\ 30 \\ 0 \end{bmatrix} \quad \underline{X}_{B\ (LL)} = \begin{bmatrix} 0 \\ -30 \\ 0 \end{bmatrix} \quad \underline{X}_{C\ (LL)} = \begin{bmatrix} 200 \\ 0 \\ 0 \end{bmatrix}$$
$$\sigma_A = 2cm \qquad \sigma_B = 2cm \qquad \sigma_C = 10cm$$

- Components of the deflection angle in O (angle between the normal vector to the ellipsoid and the gravity vertical):

$$Local\ North = \xi = 10.23''$$
$$Local\ East = \eta = 9.5''$$

3.Workflow

3.1 Import the given data

```
%1 Import Data
Lat_0 = [44, 23, 24];
Lon_0 = [8, 56, 20];
H_0 = 70;
```

3.2 Convert the coordinates of the origin from ITRF Geodetic to Global Cartesian (X, Y, Z)

```
%2 Origin ITRF to Global Cartesian
a = 6378137;
f = 1/298.257222100882711243;
e = sqrt(f*(2 - f));
Rn = a / sqrt(1-e^2*sin(ToRad(Lat_0))^2);
Lambda0 = ToRad(Lon_0);
Fai0 = ToRad(Lat_0);
X0_GC = (Rn + H_0)*cos(Fai0)*cos(Lambda0);
Y0_GC = (Rn + H_0)*cos(Fai0)*sin(Lambda0);
Z0_GC = (Rn*(1-e^2)+H_0)*sin(Fai0);
P0 = [X0_GC;Y0_GC;Z0_GC];
```

3.3 Convert A,B,C from body frame (Local Level LL) to Local Cartesian

```
%3 LL TO LC
qsi = ToRad([0,0,10.23]);
eta =ToRad([0,0,9.5]);
alpha = ToRad([30, 27, 18]);
Rx_qsi = [1,0,0; 0,cos(qsi),-sin(qsi); 0,sin(qsi),cos(qsi)];
Ry_eta = [cos(eta),0,-sin(eta); 0,1,0; sin(eta),0,cos(eta)];
Rz_alpha = [cos(alpha),sin(alpha),0; -sin(alpha),cos(alpha),0; 0,0,1];
RLC2LL = Rz_alpha*Ry_eta*Rx_qsi;
RLL2LC =transpose(RLC2LL);
A = [0;30;0];
B = [0;-30;0];
C = [200;0;0];

A_LC = RLL2LC * A;
B_LC = RLL2LC * B;
C_LC = RLL2LC * C;
```

3.4 Convert A,B,C from Local Cartesian to ITRF Global Cartesian

```
%4 LC To ITRF GC
R0 = [-sin(Lambda0), cos(Lambda0), 0;
      -sin(Fai0)*cos(Lambda0), -sin(Fai0)*sin(Lambda0), cos(Fai0);
      cos(Fai0)*cos(Lambda0), cos(Fai0)*sin(Lambda0), sin(Fai0)];
A_GC = P0 + transpose(R0)*A_LC;
B_GC = P0 + transpose(R0)*B_LC;
C_GC = P0 + transpose(R0)*C_LC;
```

3.5 Convert through EPN website ITRF GC coordinates of A,B,C to ETRF GC coordinates at epoch 1st September 2022.

```
%5 ITRF GC to ETRF GC throuth EPN
ETRF_A = [4509854.8133, 709344.7333, 4439228.7611];
```

```
ETRF_B = [4509885.8305, 709380.3976, 4439191.8018];
ETRF_C = [4509773.4717, 709521.8569, 4439282.7125];
```

3.6 Convert ETRF GC to Geodetic for A,B,C (note: latitude and longitude must be expressed in sexagesimal).

```
%6 ETRF GC to Geodatic
A_Geo = ToGeo(ETRF_A);
```

3.7 Main function

```
function result = ToRad(input)
    a =input(1);
    b =input(2);
    c =input(3);
    result = (a+b/60+c/3600)*pi/180;
end
%Cartesian to Geodetic
function result = ToGeo(input)
    X = input(1);
    Y = input(2);
    Z = input(3);

    a = 6378137;
    b = 6356752.314;
    eb = sqrt(a^2/b^2 -1);
    r = sqrt(X^2+Y^2);
    e = sqrt(1 - b^2/a^2);

    Psi = atan2(Z/(1-e^2)^(1/r),0);
    Lambda = atan2(Y,X);
    Fai = atan2((Z+eb^2*b*sin(Psi)^3),(r-e^2*a*cos(Psi)^3));
    Fai1 = ToRad([44,23,24.8206]);
    Rn = a / sqrt(1-e^2*sin(Fai)^2);
    h = r/cos(Fai) -Rn;
    result = [ToSexagesimal(Fai/pi*180);
              ToSexagesimal(Fai1/pi*180);
              [0,0,h]];
end
function result = ToSexagesimal(input)
    degreesInt = fix(input);
    minutesFloat = (input - degreesInt) * 60;
    minutesInt = fix(minutesFloat);
    secondsFloat = (minutesFloat - minutesInt) * 60;
    result = [degreesInt, minutesInt, secondsFloat];
end
```

4.Output results

4.1 Local cartesian coordinates of points A, B, C in meters

Point A =

E: -15.206

N: 25.861

U: -0.001

Point B =

E: 15.206

N: -25.861

U: 0.001

Point C =

E: 172.406

N: 101.372

U: -0.013

4.2 ITRF global cartesian coordinates of points A,B,C

Point A =

X: 4509854.339

Y: 709345.362

Z: 4439229.142075994

Point B =

X: 4509885.357

Y: 709381.026

Z: 4439192.182868748

Point C =

X: 4509772.998

Y: 709522.485

Z: 4439283.093514732

4.3 ETRF cartesian coordinates of points A, B, C

Point A =

X: 4509854.8133

Y: 709344.7333

Z: 4439228.7611

Point B =

X: 4509885.8305

Y: 709380.3976

Z: 4439191.8018

Point C =

X: 4509773.4717

Y: 709521.8569

Z: 4439282.7125

4.4 ETRF geodetic coordinates of points A,B,C

```
Point A =
lat:  [44,  23,  24.8206]
lon:  [8,  56,  19.2816]
h:    [69.998]
Point B =
lat:  [44,  23,  23.1450]
lon:  [8,  56,  20.6556]
h:    [69.999]
Point C =
lat:  [44,  23,  27.2669]
lon:  [8,  56,  27.7582]
h:    [69.988]
```

4.5 Standard deviations of points A,B in East, North, Up in cm

```
E:  10.2
N:  10.2
U:  10.2
```

4.6 Standard deviations of points c in East, North, Up in cm

```
E:  14.1
N:  14.1
U:  14.1
```

5. Summary

In this experiment, the covariance approximation propagation method was mainly used in the process of converting LL to LC and LC to GC.

The reason for using covariance approximation in coordinate system transformations involves the concept of error propagation. In many applications we may need to take multiple measurements, each with a certain amount of error. When these measurements are combined or transformed into a different coordinate system, error propagation becomes critical.

During the coordinate system conversion process, The covariance matrix plays an important role here because it helps us estimate how the error propagates during the coordinate system transformation.The approximation of the covariance matrix is that the coordinate system transformation process, we can usually use the Jacobian matrix to approximate the propagation of the error. The Jacobian matrix describes a local linear approximation of a multivariable function. By multiplying the Jacobian matrix with the original covariance matrix, we can get an estimate of the error in the coordinate system transformation.

Overall,the application of the covariance approximation coordinate system transformations is to better understand how errors propagate so that these errors can be taken into account in the final results. This is particularly important for measurement and positioning systems that require high accuracy and reliability.