1 Introduction

The purpose of this lab is to gain experience in geodetic GNSS data processing and data analysis. To do this, RINEX files of GNSS observation data for six stations around the world on two adjacent was obtained. Then, this data was edited and processed using NRCan’s Precise Point Positioning service to include 48 sets. All six stations have a dataset with 24 hours and 30 minutes of data, processed in both kinematic and static modes, over two days. With all processed datasets obtained, the effects of various processing parameters including processing type and data arc will be analyzed by comparing the estimated station coordinates with published reference coordinates.

2 Methodology

**2.1 Obtaining Data**

Files containing GNSS observation data for all six stations on two adjacent days were first obtained from the SOPAC & CSRC Garner GPS Archive in .crx format. Three stations are located in Canada, and the other stations are located in China, India, and the United States. To convert to a usable RINEX format, the provided *crx2nrx.exe* executable file was used. Then, these twelve files were duplicated and edited to included data arcs of both 24 hours and 30 minutes. Next, the NRCan CSRS-PPP service was used to process each dataset in both kinematic and static Precise Point Positioning processing modes.

**2.2 Description of Acquired Data**

Prior to editing the RINEX files or processing the data with the NRCan CSRS-PPP service, the obtained RINEX files contain raw observation data for satellites in both the GPS and GLONASS constellations. Each file contains satellites’ broadcast ephemeris’ at epochs over 24 hours, as well as a variety of parameters used for point positioning.

**2.3 Data Processing**

Once files were duplicated and edited to contain both 24-hour and 30-minute data arcs, the NRCan CSRS-PPP service applies the method of relative positioning to provide precise positioning data, including the estimated latitude, longitude, and height of a specific receiver (station). In the kinematic processing mode, the receiver (station) is treated as if it is in motion, and a greater number of satellites over multiple epochs are required to compute the station position estimate. In the static processing mode, stations are treated as if they remain in the same position, and fewer satellites are required over a single epoch to compute an estimate of the station position.

3 Data Analysis and Discussion

**3.1 Analysis Overview**

Once datasets were obtained, all analyses were completed using MATLAB, and provided code was used to read each RINEX file. Desired plots included the number of satellites and DOP as functions of time, a time series showing the evolution of horizontal and vertical error, histograms including the vertical and horizontal RMSE for specific results, and histograms including the time it took specific solutions to reach 5 cm of vertical and horizontal error.

Though the provided *readPos.m* was capable of reading a RINEX file and returning various types of data including time epochs, GDOP at epochs, and ECEF coordinate at all epochs, additional data was required. The *readPos.m* file was edited to return error in East, North, Up instead of ECEF XYZ coordinates, both horizontal and vertical error at each epoch, total horizontal RMSE, total vertical RMSE, and the time it took any solution to reach 5cm of horizontal/ vertical error. Additionally, this function was used to develop plots of GDOP and number of satellites as a function of time. The edited *readPos.m* file will be described further in *Data Processing for Individual Files* and can be found in Appendix 1.0.

Since *readPos.m* obtains data from individual RINEX files, another file called *main.m* was developed for analysis purposes. Here, all RINEX files were first organized by station, data arc length, processing mode, day of acquisition, and various combinations of the mentioned categories. By organizing files in this way, vectors could be developed that included, for example, the horizontal RMSE for all datasets with a 30-minute data arc, or the vertical RMSE for all datasets processed in static mode. These vectors were then used to develop histograms to analyze the effects of data arc length and processing mode on position estimates. In addition, *main.m* was developed to plot overlapped time series of horizontal or vertical error in specific combinations of datasets for further analysis. The *main.m* function will be described further in the *Effects of Data Arc Length and Processing Mode on Position Estimates* section and can be found in Appendix 2.0.

**3.2 Data Processing for Individual Files**

In order to draw meaningful conclusions from the datasets, the first step was to modify the provided *readPos.m* file to output variables relating to the quality of the data in each individual RINEX file. These variables include time epochs in a usable format, position error, and summations of position error. Detailed descriptions are provided in the following four sections, along with brief sections of *readPos.m*.

**3.2.1 DOP and Number of Satellites per Epoch**

To determine the GDOP at station receiver locations at each epoch for a specified file, the *readPos.m* function iterates line by line and stores GDOP per epoch in a data structure called *solutions*:

solutions.GDOP(epoch\_index, 1) = str2double(split\_line(index\_GDOP));

Next, the number of available satellites at each epoch is read from the RINEX file and stored in *solutions*:

solutions.num\_sat(epoch\_index, 1) = str2double(split\_line(index\_numSat));

Finally, since time is read from the RINEX file in Hours, Minutes, and Seconds (HMS), these values were converted to decimal hours and stored in a single vector called *decimalHour* in *solutions*:

hms = split\_line{index\_HMS};

solutions.time(epoch\_index, :) = [str2double(hms(1:2)) str2double(hms(4:5)) str2double(hms(7:11))];

% decimal hours (used for time series instead of just HMS)

solutions.decimalHour(epoch\_index, :) = [str2double(hms(1:2)) + str2double(hms(4:5))/60 + str2double(hms(7:11))/3600];

**3.2.2 Horizontal and Vertical Error per Epoch**

To determine horizontal and vertical error at each epoch, the estimated latitude, longitude, and height of the corresponding station are first read from the RINEX file. Once obtained, the provided *llh2XYZ.m*  function (Appendix 3.0) is used to convert from latitude, longitude, and height to ECEF XYZ coordinates. These llh and XYZ coordinates are then stored in *solutions*:

[X, Y, Z] = llh2XYZ(lat\_degree, lon\_degree, height);

solutions.llh(epoch\_index, :) = [lat\_degree lon\_degree height];

solutions.ECEF(epoch\_index, :) = [X, Y, Z];

However, since the error is computed using East/North/Up, a function called *XYZ2enu* (Appendix 4.0) was written to transform between the two systems. *XYZ2enu* makes use of the following equation:

This function was then called to store East/North/Up differences in a matrix called *enu* in *solutions*:

% enu relative to reference station

[e, n, u] = XYZ2enu(stations(station,1), stations(station,2), stations(station,3), X, Y, Z, lat\_degree, lon\_degree);

solutions.enu(epoch\_index, :) = [e, n, u];

Finally, horizontal and vertical error are computed as:

These values are computed at each epoch and stored in *solutions* as follows:

% horizontal error

hor\_error = sqrt(e^2 + n^2);

solutions.hor\_error(epoch\_index, :) = hor\_error;

% vertical error

vert\_error = abs(u);

solutions.vert\_error(epoch\_index, :) = vert\_error;

**3.2.3 Total Horizontal and Vertical RMSE**

To compute the total horizontal and vertical RMSE for a singular dataset, the sum of each error over all available epochs was first required. To compute this, error values were initialized outside of the epoch-by-epoch loop:

hdiff = 0;

vdiff = 0;

Within the epoch-by-epoch loop, each value was iteratively updated:

% sum of differences used for horizontal RMSE

hdiff = hdiff + hor\_error^2;

% sum of differences used for vertical RMSE

vdiff = vdiff + vert\_error^2;

Finally, the total horizontal and vertical RMSE for a single dataset can be computed and stored in *solutions*:

% horizontal root mean square error

h\_RMSE = sqrt(hdiff/epoch\_index);

solutions.h\_RMSE = h\_RMSE;

% vertical root mean square error

v\_RMSE = sqrt(vdiff/epoch\_index);

solutions.v\_RMSE = v\_RMSE;

**3.2.4 Computation of Time to Reach 5cm Horizontal and Vertical Error**

When determining the time that it takes a solution to reach 5cm of error, our first approach was to determine the last epoch at which the solution rose over 5cm of error, and then use the following epoch. However, some noisy datasets spike to 5cm at random times and this approach gave inaccurate results.

Our second approach was to determine the first epoch at which the error was under 5cm. For horizontal error, this was completed by first initializing the 5cm error epoch to zero and then looping through all horizontal errors to find the desired epoch. However, if the solution never reached 5cm accuracy, the 5cm epoch was stored as NaN. This process can be seen in the following code:

time to reach 5cm horizontal error

solutions.h\_error\_5cm = 0;

terminate = 1;

k = 1;

while terminate == 1

if solutions.hor\_error(k) <= 0.05

solutions.h\_error\_5cm = solutions.decimalHour(k);

terminate = -1;

elseif k == length(solutions.hor\_error)

terminate = -1;

solutions.h\_error\_5cm = NaN;

else

k = k+1;

end

end

For vertical error, the process was identical:

% time to reach 5cm vertical error

solutions.v\_error\_5cm = 0;

terminate = 1;

k = 1;

while terminate == 1

if solutions.vert\_error(k) <= 0.05

solutions.v\_error\_5cm = solutions.decimalHour(k);

terminate = -1;

elseif k == length(solutions.vert\_error)

terminate = -1;

solutions.v\_error\_5cm = NaN;

end

k = k+1;

end

**3.3 Number of Satellites and GDOP as a Function of Time**

For each RINEX file, the GDOP and number of satellites were plotted against time using the *GDOP, num\_sat,* and *decimalHour* matrices stored in the *solutions* data structure:

% \*\*\*\*\*\*\*\*\*\* GDOP and Number of Satellites \*\*\*\*\*\*

figure('Position', [50 50 1000 600])

title(sprintf('GDOP and Number of Satellites vs Time for %s', path\_file), 'Interpreter', 'none');

yyaxis right

plot(solutions.decimalHour, solutions.GDOP);

ylabel('GDOP'); ylim([0 5])

yyaxis left

plot(solutions.decimalHour, solutions.num\_sat);

ylabel('Number of Satellites'); ylim([0 25])

grid on; xlabel('Time (Hours)')

Plots for a 24-hour and 30-minute data arc of ALGO00CAN can be seen in figures 1 and 2:

Chart

Description automatically generated

Figure - GDOP and Number of Satellites vs Time for a 24-Hour Data Arc of ALGO00CAN

Chart, line chart

Description automatically generated

Figure - GDOP and Number of Satellites vs Time for a 30-Minute Data Arc of ALGO00CAN

This inverse relationship between the number of satellites and GDOP can be seen for all datasets. The relationship is very symmetrical for the 30-minute data arc, however the noise in the 24-hour data arc can be attributed to other factors including the spread of the available satellites. The same noise can be seen in Figure 3 below.

A picture containing chart

Description automatically generated

Figure - GDOP and Number of Satellites vs Time for a 24-Hour Data Arc of WHIT00CAN

**3.4 Data Processing for Multiple Files**

**3.4.1 Effects of Data Arc Length and Processing Mode on Position Estimates**

In order to analyze relationships between various datasets, a function called *main.m* was developed. All RINEX files are first organized by station, data arc length, processing mode, and day of acquisition. These combinations allowed us to compute vectors of all horizontal and vertical RMSE or time to reach 5cm of error for all 24-hour data, 30-minute data, kinematic data, and static data. Complete code can be found in Appendix 2.0, but the following example shows how horizontal and vertical RMSE vectors are computed for all 24 sets of 24-hour data:

% \*\*\*\*\*\* horizontal and vertical RMSE for 24 hour data arc \*\*\*\*\*\*\*\*\*\*\*

% j counts through the stations numbers (1 to 6) to be used in readPos.m

% i iterates through each element of the 24H vector

h\_RMSE\_24hr = zeros(24,1);

v\_RMSE\_24hr = zeros(24,1);

j = 1;

for i = 1:24

solutions = readPos(sprintf('%s', station24H{i}), j);

h\_RMSE\_24hr(i) = solutions.h\_RMSE;

v\_RMSE\_24hr(i) = solutions.v\_RMSE;

j = j+1;

if j > 6

j = 1;

end

end

Next, all RINEX files were organized into the following eight categories:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Day 2 | | | | Day 3 | | | |
| 24 Hour | | 30 Minute | | 24 Hour | | 30 Minute | |
| Kinematic | Static | Kinematic | Static | Kinematic | Static | Kinematic | Static |

This categorization allowed us to develop overlapping histograms comparing RMSE and time to reach 5cm of error for specific cases, such as horizontal RMSE of kinematic data vs static data, or time to reach 5cm or vertical error for a 24-hour data arc vs a 30-minute data arc. Additionally, this organization allowed us to plot time series of the evolution of horizontal and vertical error for very specific scenarios. For example, the following portion of *main.m* computes plots of vertical and horizontal error as a function of time for all 24-hour static data on Day 2:

24H STATIC DAY 2

all\_hor\_error24S2 = [];

all\_vert\_error24S2 = [];

hour = [];

length = [];

sizes = [];

for i=1:6

solutions = readPos(sprintf('%s', station24S2{i}), i);

hor\_error = solutions.hor\_error;

vert\_error = solutions.vert\_error;

hour = solutions.decimalHour;

if i==5

hor\_error(numel(hor\_error)+1) = 0;

hour(numel(hour)+1) = hour(numel(hour));

end

if i==5

vert\_error(numel(vert\_error)+1) = 0;

end

length = numel(hor\_error);

sizes = [sizes, length];

all\_hor\_error24S2 = [all\_hor\_error24S2, hor\_error];

all\_vert\_error24S2 = [all\_vert\_error24S2, vert\_error];

subplot(2,1,1)

plot(hour, all\_hor\_error24S2(:,i));

hold on

subplot(2,1,2)

plot(hour, all\_vert\_error24S2(:,i));

hold on

end

subplot(2,1,1)

title('24 Hour STATIC Day 2 Horizontal Error');

xlabel('Hours');

ylabel('Horizontal Error (m)');

legend('ALGO00CAN','DUBO00CAN','WHIT00CAN','GOLD00USA','WUHN00CHN','IISC00IND')

subplot(2,1,2)

title('24 Hour STATIC Day 2 Vertical Error');

xlabel('Hours');

ylabel('Vertical Error (m)');

legend('ALGO00CAN','DUBO00CAN','WHIT00CAN','GOLD00USA','WUHN00CHN','IISC00IND')

**3.5 Effect of Data Arc on RMSE and Time to Reach 5cm Error**

**3.6 Effect of Processing Mode on RMSE and Time to Reach 5cm Error**