Levenberg-Marquardt Filter for Orbit Estimation

a project presented to
The Faculty of the Department of Aerospace Engineering
San José State University

in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

By

Robert Ziegler

May 2019

approved by

Prof. Jeanine Hunter
Faculty Advisor

The Designated Project Advisor(s) Approves the Project Titled

LEVENBERG-MARQUARDT FILTER FOR ORBIT ESTIMATION

by

Robert Ziegler

APPROVED FOR THE DEPARTMENT OF AEROSPACE ENGINEERING

SAN JOSÉ STATE UNIVERSITY

May 2019

Prof. Jeanine Hunter	Department of Aerospace Engineering	Advisor

ABSTRACT

LEVENBERG-MARQUARDT FILTER FOR ORBIT ESTIMATION

By Robert Ziegler

This paper tests the Levenberg-Marquardt method of least-squares as it is applied to orbit estimation using noisy Doppler data. Doppler data used in the analysis is simulated by calculating range rate at multiple points along the path of a satellite at times the satellite would pass over a real ground station. The paper begins by discussing how real Doppler data would be used for orbit estimation. Next, a reference frame used for the analysis is defined. Then, the methods used to acquire simulated data are outlined. Finally, the Levenberg-Marquardt least-squares algorithm is discussed in detail, and the results of the experiment are analyzed. It is determined that the Levenberg-Marquardt method of least-squares is an excellent filter for providing a “best estimate” of a state vector, and thus the orbit, of a satellite using Doppler data.

Acknowledgments

When it came time to begin a project to fulfill the Master of Science in Aerospace Engineering requirements, my desire was to work on a challenging problem that would push my limits as an engineer. This desire was fulfilled when Professor Jeanine Hunter became my project advisor. As I have progressed through this project (hitting every wall that could possibly be found), Professor Hunter has given me guidance and has shown me an extreme amount of patience, and for this she has my sincerest gratitude.

Table of Contents

List of Symbols

Symbol	Definition	Units (SI)
a	semimajor axis length	ft (m)
c	Speed of light	ft/s (m/s)
d	Day	
e	Eccentricity	
F	Force	lbs (N)
f	Vector of residuals	
$f_{()}$	frequency	Hz
f_{obl}	Oblateness of the earth	
Δf	Doppler shift	Hz
H	Altitude	ft (m)
i	inclination	deg or rad
J_0	Julian day at 0 hours UT	days
JD	Julian day	days
J	Jacobian matrix	
K	Constant factor of frequency transposure	
m	Month	
N	Doppler cycle count	cycles
q	Vector of Marquardt solutions	ft, ft/s (m, m/s)
r	Satellite position wrt Earth center	ft (m)
R	Ground site position wrt Earth center	ft (m)
S	Sum of squares of residuals	
t	Time	s
T_0		
UT	Universal time	
v	Velocity of satellite wrt Earth center	ft/s (m/s)
V_{wave}	Velocity of the wave of a radio signal	ft/s (m/s)
V_{source}	Velocity of the source of a radio signal	ft/s (m/s)
X, x	State Vector	ft, ft/s (m, m/s)
y	Year	
Greek Symbols		
α	Right ascension	deg or rad
δ	Declination	deg or rad
ε	Error	
θ	True anomaly	deg or rad
λ	Marquardt parameter	

λ_f	Wavelength	ft (m)

$\dot{\rho}$	Topocentric range-rate	ft/s (m/s)
σ	Sidereal time	deg
τ	Orbital period	s
φ	Longitude	deg or rad
ω	Argument of perigee	deg or rad
Ω	Right ascension of the ascending node	deg or rad
Subscripts		
$( )_c$	Geocentric	
$( )_d$	Geodetic	
$( )_E$	Earth	
$( )_{eq}$	Equatorial	
$( )_g$	Ground	
$( )_{pol}$	Polar	
$( )_r$	Receiver	
$( )_s$	Satellite	
$( )_t$	Transmitter	
Acronyms		
DE	Doppler Effect	
ECF	Earth-Centered, Fixed	
GPS	Global Positioning System	
GNM	Gauss-Newton Method	
LMA	Levenberg-Marquardt Algorithm	
LS	Least Squares	
OASIS	Orbit Analysis and Simulation Software	
OD	Orbit Determination	
PRARE	Precise Range and Range-rate Equipment	
RF	Radio Frequency	
SDM	Steepest Descent Method	
TCA	Time of Closest Approach	
TDRS	Tracking and Data Relay Satellites	
TLE	Two-Line Element	
VLBI	Very Long Baseline Interferometry	

1. INTRODUCTION

Orbit determination is a tool used by engineers, scientists, and hobbyists to understand the trajectory of objects travelling through space. Using orbital mechanics and observations of the position and velocity of the satellite, observers can predict the position of the object over time. While this prediction was problematic for the first satellite in space, technology now allows precise tracking of satellites using enhanced physical models and computing power. Presently, anyone with access to the internet can either look up the ephemeris of their desired satellite or download sophisticated software that can approximate and display the trajectory of many orbiting satellites; however, a different approach is proposed in this report.

As more satellites are introduced into earth orbit, novel approaches to orbit determination are required. While satellites can send ephemeris data and computer software can approximate the position of the satellite over time, there must be alternative methods which do not rely on such technology. The subject of this report is the derivation of the orbital elements of a satellite using the relative velocity between the satellite and a ground station. This relative velocity will be measured through examination of radio frequencies from a satellite received at a ground station. Using a Damped Least-Squares (DLS) algorithm, noisy data received from an Earth-orbiting satellite will be corrected to find a "best estimate" for the true orbit of the satellite. Figure 1 illustrates how noisy measurement data can lead to an incorrect orbit. As seen in Figure 1, $\mathbf{x}$ variables represent state vectors along the reference orbit, and $\hat{\boldsymbol{x}}$ variables represent state vectors along an estimated orbit. The noisy data will be curve-fit to a reference orbit using the prescribed DLS algorithm.

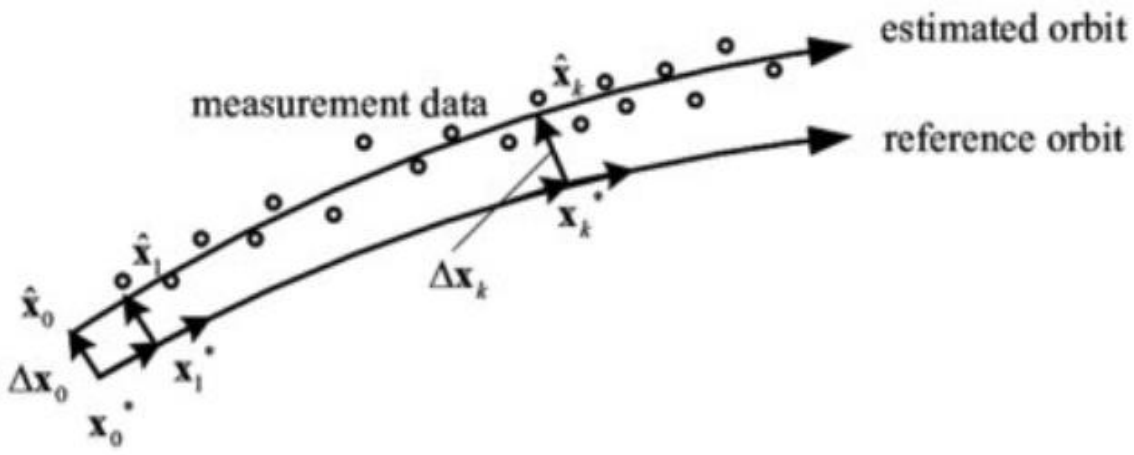

Figure 1.1 Damped Least Squares Estimation [1].

1.1 DOPPLER ESTIMATION

Many approaches have been taken to use Doppler data to predict the orbit of a satellite. Some of the simpler methods assume a circular orbit, while the more complex variations are flexible with orbital parameters but require more information on the orbital history of the satellite. In this section, texts and articles which discuss the Doppler effect DE, and how DE of a received signal pertains to orbit determination.

As new equipment and software are introduced into space systems, testing these novelties in the field is critical. The analysis of the Precise Range and Range-rate Equipment (PRARE), a satellite tracking system, and how this system aids in precise orbit determination, presented by Bordi [2], is an example of such testing.

Acting as a supplement to laser tracking, the PRARE system provides range and Doppler data of a satellite by sending two modulated signals, in X-band and S-band, respectively. This data is collected and compared at PRARE ground stations where observers use the time delay between X- and S-band signals to determine the ionospheric delay. Included in the signals are time data which is

used to acquire ephemeris data and allow tracking of future passes of the satellite. After processing, the X-band signal is modulated and sent back to the PRARE space segment where range ρ measurements are calculated using the two-way signal time [2]:

$$\rho = \frac{1}{2}c(\Delta t + \Delta t_{corr}) - \Delta\rho_{trop} - \Delta\rho_{iono} + \varepsilon \tag{1.1}$$

where Δt is the time measurement between transmission and reception of the signal, $\Delta\rho$ terms are corrections made for atmospheric delays, and ε is the errors made in observation. Due to the relative velocity between the satellite and ground stations, Doppler data is also collected during the signal exchange [2].

The PRARE system analyzes the Doppler frequency shift which results from the relative velocity between a satellite and a ground station. According to Bordi [2], this change in frequency is how PRARE calculates range-rate of the satellite. This method requires the measurement of two frequencies. The first measured frequency is of the received signal at the ground station per:

$$f_{r_g} = f_{t_s} - \frac{\dot{\rho_d}}{\lambda} = f_{t_s}\left(1 - \frac{\dot{\rho_d}}{c}\right), \tag{1.2}$$

where f is frequency, $\dot{\rho}$ is the range-rate of the satellite with respect to the ground station, λ is the wavelength of the signal, and the subscripts, r_g and t_s, represent signal received by the ground station and the signal transmitted by the PRARE space segment, respectively. The second necessary frequency is measured on the satellite per:

$$f_{r_s} = f_{t_g}\left(1 - \frac{\dot{\rho_u}}{c}\right) = K \cdot f_{t_s}\left(1 - \frac{\dot{\rho_d}}{c}\right) \cdot \left(1 - \frac{\dot{\rho_u}}{c}\right), \tag{1.3}$$

where K is the constant factor of frequency transposure, and subscripts r_s and t_g represent the signals received by the satellite and transmitted by the ground station, respectively. The Doppler count, which is a measure of cycles in the signal, is found by integrating the differences between the signals found in Eqns. (1.1.2) and (1.1.3). Average range-rate is then calculated from the start and end of each integration interval, as in [2]:

$$\frac{\Delta\rho}{\Delta t} = \frac{1}{2}\left(\frac{N + N_{corr}}{\Delta t}\right)\frac{c}{f_{ref}} + \varepsilon, \tag{1.4}$$

where Δρ is the difference between ranges in the integration interval, N is the Doppler cycle count, and f_{ref} is the function

$$f_{ref} = K \cdot f_{t_s}. \tag{1.5}$$

Throughout the remainder of Bordi's [2] research, errors induced by various sources are discussed, and the methods used to correct these errors are explained. The analysis [2] relies on sophisticated hardware, but in other cases, orbit history of the satellite is used to predict the future orbit of the object.

Using knowledge of the orbital history of a satellite, the study produced by Amiri and Mehdipour [3] presents a method to accurately measure the Doppler shift, regardless of the orbit of the satellite. Using known values for the position of the satellite P_s and ground transceiver P_g, a relationship

between relative velocity v_t and Doppler shift can be calculated (annotated from [3]):

$$\Delta f = \frac{f_0 v_t}{c} \tag{1.6}$$

where

$$v_t = \frac{d(P_s - P_g)}{dt} \tag{1.7}$$

in spherical coordinates, and f_0 is the carrier frequency. To find v_t, velocities of the satellite and ground station are studied in ECEF coordinates, and perturbing forces are analyzed.

Finding a value for v_t requires an orbit generator with corrections for the following perturbing forces R, S, and W (annotated from [3]):

$$\boldsymbol{\gamma}_p = R\boldsymbol{q_r} + S\boldsymbol{q}_\Theta + W\boldsymbol{q_z} \tag{1.8}$$

$$K = -1.5 \cdot \mu \cdot J_2 \cdot R_E^2 \cdot \frac{1}{r^4} \tag{1.9}$$

$$R = K(1 - 3\sin^2(\omega + \nu) \cdot \sin^2(i)) \tag{1.10}$$

$$S = K \cdot \sin(2(\omega + \nu) \cdot \sin^2(i)) \tag{1.11}$$

$$W = K \cdot \sin(\omega + \nu) \cdot \sin(2i) \tag{1.12}$$

Gauss planetary equations can now be used to illustrate how orbit parameters of a satellite are influenced by these outside forces (annotated from [3]):

$$\frac{da}{dt} = \frac{2}{n\sqrt{1-e^2}}(eR\sin(\theta) + (1 + e\sin\theta)\cdot S) \tag{1.13}$$

$$\frac{de}{dt} = \frac{\sqrt{1-e^2}}{na}(R\sin\theta + (\cos E + \cos\theta)\cdot S) \tag{1.14}$$

$$\frac{di}{dt} = \frac{1}{na\sqrt{1-e^2}}\cdot\frac{r}{a}\cdot\cos(\theta + \omega)\cdot W \tag{1.15}$$

$$\frac{d\Omega}{dt} = \frac{1}{na\sqrt{1-e^2}}\cdot\frac{r}{a}\cdot\frac{\sin(\theta + \omega)}{\sin i}\cdot W \tag{1.16}$$

$$\frac{d\omega}{dt} = \frac{\sqrt{1-e^2}}{nae}\cdot\left(-R\cos\theta + \left(1 + \frac{1}{1+e\cos\theta}\right)\cdot S\sin\theta - \frac{d\Omega}{dt}\cdot\cos i\right) \tag{1.17}$$

$$\frac{dM}{dt} = n + \frac{1-e^2}{nae}\left(R\left(\frac{-2e}{1+e\cos\theta} + \cos\theta\right) - S\left(1 + \frac{1}{1+e\cos\theta}\right)\cdot\sin\theta\right) \tag{1.18}$$

Equation (1.13 – 1.18) decide if there is a known perturbing force vector. Analytical calculations will determine the orbital parameter rate of change. Using these forces with an orbit generator allows the derivation of v_t, and ultimately Δf, after a series of transformations to ECEF coordinates. In the next article, Doppler estimation is used for satellite identification.

In February 2012, seven 1U CubeSats and two larger satellites were launched into orbit as part of a

student science endeavor. In orbit, the proximity of the satellites made individual identification difficult. To combat this issue, respective orbits were calculated by Marcin and Grzegorz [4] using the Doppler signal of the satellites.

Unlike the previous methods, the measurements taken in by Marcin and Grzegorz [4] rely on Software Defined Radio (SDR) and a reference signal generator, as well as computer software which displayed orbital positions of the nine satellites. Over the course of five days, the satellite group passed over the ground station seven times, which allowed the collection of thirty downlink frequency measurements. With an assumed satellite frequency, Orbiton software is used to discern the desired satellite from the group. The next paper derives position over time of satellites using inclined circular orbits.

In his paper, Tabakovics's [5] objective is to find $\dot{\rho}$ using trajectory coordinates of the satellite. As in the previous article, measurements are taken at a ground station receiver. Using longitude and latitude coordinates in the orbital plane, the trajectory of the satellite in circular inclined orbit can be stated as (annotated from [5]):

$$r_s = R_E + Altitude \tag{1.19a}$$

$$\phi = 0\,; \tag{1.19b}$$

$$\lambda = \omega_s \cdot t + \phi_0 \tag{1.19c}$$

where R_E is the mean radius of the earth, R_E, ϕ and λ are longitude and latitude, respectively, and ω_s is the angular velocity of the satellite. For azimuth and elevation measurements at ground stations, the coordinates from Eqns. (1.4.7) are transformed to the geocentric equatorial plane Earth

rotating coordinate system [5]:

$$r_s = r_s \tag{1.20a}$$

$$\phi_{ER} = \sin^{-1}(\sin i \cdot \sin(\omega_s \cdot t + \phi_0)) \tag{1.19b}$$

$$\lambda_{ER} = \tan^{-1}[\cos i \cdot \tan(\omega_s \cdot t + \phi_0)] + \Omega - \theta_G - \omega_s \cdot t + k\pi \tag{1.20c}$$

Range-rate of a satellite can be found by differentiating the distance of the satellite [5]:

$$r = [(X_s - X_g)^2 + (Y_s + Y_g)^2 + (Z_s - Z_g)^2]^{\frac{1}{2}}$$

$$= \sqrt{R_E^2 + r_S^2 - 2R_E r_S \cos\gamma} \tag{1.21}$$

where subscripts s and g represent satellite and ground station, respectively, and γ is the geocentric angle between the ground station and the satellite. Introducing an expression for maximum elevation δ_m and differentiating the position vector, relative velocity can be obtained [5]:

$$\dot{\rho} = \frac{dr}{dt}$$

$$= \frac{R_E r_S \sin[\,\delta_m + \sin^{-1}(\frac{R_E}{R_E + r_S} \cdot \cos\delta_m)]\sin[(\omega_S - \omega_E \cdot \cos i)t](\omega_S - \omega_E \cdot \cos i)}{\sqrt{R_E^2 + R_E^2 - 2R_E r_S \sin[\delta_m + \sin^{-1}\left(\frac{R_E}{R_E + r_S}\cos\delta_m\right)]\cos[(\omega_S - \omega_E \cdot \cos i)t]}} \quad (1.20)$$

Doppler shift can now be found using $\dot{\rho}$ [5]:

$$\Delta f = \dot{\rho} \cdot \left(-\frac{f_t}{c}\right) \quad (1.21)$$

where f_t is the transmitted frequency of the downlink signal of the satellite, and c is the speed of light. Using Doppler measurements to improve global positioning system (GPS) performance is the topic of the next article.

As aircraft maneuver through the air, the Doppler shift induced on received signals can be much greater than observed values at a ground station. In their investigation, Agostino, Manzino, and Marucco [6] use a Kalman filter estimator to improve GPS tracking of aircraft using Doppler measurements. In this process, a precise ephemeris of a satellite is used, along with the inherent Doppler shift to calculate the velocity of the aircraft. It is determined that using this Kalman filter estimation reduces errors caused by noisy measurements.

In another experiment, Ialongo [7] uses a cycle counter to read two-way Doppler measurements and produce range rate of a satellite. This method feeds an input frequency f_i into a counter, where

$$f_i = \frac{5}{4} f_t - f_r \quad (1.22)$$

An expression for range rate $\dot{\rho}$ is derived from the two-way signal as [7]:

$$\dot{\rho} = \frac{RRN - F(\theta, \phi)\frac{1}{2}}{1 - \left[RRN - F(\theta, \phi)\frac{c}{2}\right] \cdot \frac{1}{c}} \tag{1.23}$$

where

$$RRN = \frac{cN_2}{2048N_2 + 26240N_1} \tag{1.24}$$

N_1 and N_2 are cycle counts, and

$$F(\theta, \phi) = \frac{(R_t v_t + \rho_t v_t \sin\theta - \rho_t v_s \cos\phi)}{R_t v_t} \tag{1.25}$$

Here, ρ is the distance travelled by the signal.

In this section, methods of measuring the Doppler effect and range rate of a satellite were discussed. Now, applications to orbit determination will be examined.

1.2 ORBIT DETERMINATION FROM DOPPLER DATA

Having covered a variety of methods used to measure the Doppler shift inherent in satellite communication, orbit determination schemes which use this data are presented in this section, starting with a simple circular orbit-based algorithm.

The experiment presented by Schuch [8] uses observations of the orbital period τ of a satellite to estimate a circular orbit. Using Doppler measurements, a Time of Closest Approach (TCA) is determined by finding when the received frequency from a satellite is equal to the transmitted

frequency, that is, when there is no Doppler shift present. Noting the TCA, a second pass is evaluated, and a first estimation of τ can be made. The error incurred from the rotation of the earth is corrected by repeating this process for two successive descending passes. The time elapsed between these passes is an integer multiple of τ. This integer can be calculated with the equation (annotated from [8]):

$$n = int\left(\frac{n\tau}{\tau_{est}}\right) \tag{1.26}$$

where τ_{est} is the first estimation of orbital period. Because the orbit in this example is assumed to be circular, Keplerian elements, which are detailed in Chapter 2, can be found from [8]:

$$\tau = \sqrt{\frac{4\pi^2 r^3}{GM}}. \tag{1.27}$$

The system presented by Kirschner, et. al [9] uses six tracking measurements consisting of a combination of range and Doppler data. A homotopy continuation method is used to solve a set of nonlinear equations. This method can be used when little or nothing is known about the orbit. Range and Doppler data are processed in Tracking and Data Relay Satellites (TDRSs). Using preliminary orbit determination, this process is precise enough to determine trajectory of future passes. The homotopy continuation method uses a mapping parameter to step through a solution curve in seven-dimensional space, which produces a set of orbital parameters [9].

In another study, Izsak [10] extracted Doppler measurements simultaneously from three ground stations. Using the Doppler data from these three stations, Izsak [10] found radial velocity of the

satellite with

$$\dot{\rho} = c\left(1 - \frac{f_r}{f_t}\right) \tag{1.28}$$

If the distance between the satellite and each of the ground stations is known at two different times, the Keplerian orbit can be formed through methods which will be discussed in the next chapter. Moving to a higher earth orbit, an experiment performed by Estefan [11] uses differenced Doppler for elliptical orbiters. A method of orbit determination for high-orbit elliptical satellites, Very Long Baseline Interferometry (VLBI), is under investigation [11] for its ability to improve orbit accuracy. The problem with this process is its high cost. Termed "quasi-VLBI," an alternative differenced (two-way minus three-way) Doppler is proposed. While data measured with differenced Doppler is not as accurate as seen with VLBI, Doppler and range data can be supplied much faster for navigation purposes [11].

Differenced Doppler first relies on extracting range measurements from Orbit Analysis and Simulation Software (OASIS), which was developed at the Jet Propulsion Laboratory (JPL). Taking the time derivative of this range, range rate can be measured. Using downlink signals measured at three ground stations over the same period, a differencing of the signals is collected, as in the equation [11]:

$$\begin{aligned} \Delta\dot{\rho}(\Delta\rho) &= [\dot{\rho_{1u}}(\rho_{1u}) + \dot{\rho}_{1d}(\rho_{1d})] - [\dot{\rho_{1u}}(\rho_{1u}) + \dot{\rho}_{2d}(\rho_{2d})] \\ &= \dot{\rho_{1u}}(\rho_{1u}) - \dot{\rho}_{2d}(\rho_{2d}) \end{aligned} \tag{1.31}$$

where the subscripts u and d represent direction of the signal (uplink/downlink) and the numerical

subscripts represent different ground stations.

Guier and Weiffenbach [12] use the entirety of a Doppler curve to obtain orbital elements in their article. While many Doppler-based orbit determination schemes include an intermediary process, steps can be taken to maximize Doppler data by directly calculating the six orbital elements from the frequency shift curve. Additional elements, totaling eight, are extracted to account for errors, such as refraction from the ionosphere. Although computational cost is higher using this single-pass method, its results have shown that such calculations are possible. The final article reviewed in the present chapter concerns the use of the Doppler effect in GPS measurements.

The derivation of the GPS relativistic Doppler effects is given by Zhang, et. al [13]. In the GPS observation system, additional changes in frequency are present. These shifts are caused by gravity potential from the geoid shape of the earth, the gravity field of the earth, and the orbital eccentricity of the satellite. To correct the relativistic effects, a special relativity term is added to the equation for received frequency, which will not be included in this paper as these corrections are not desired for the present analysis.

1.3 RESEARCH OBJECTIVES

Sophisticated software has enabled aerospace companies to track satellites with accurate measure. For civilian satellite enthusiasts, there is also satellite tracking software, although these programs take a loss on accuracy. To mitigate this loss, observations of frequency shift from the transmitted radio signal of a satellite can be used to determine the orbital elements of the orbit.

The primary objective of this report is to define an algorithm which uses range rate of a satellite with respect to a ground station to determine the orbital elements of the satellite. To do this, measurements of the signal received from the satellite will be taken as the satellite travels overhead. The collection of these frequency data points will form a Doppler curve which will be used to

calculate the range rate. A least-squares algorithm will then be introduced to an orbit generator for final orbit determination.

In the next chapter, data will be extracted from the described orbit propagator. The data pulled will have contain the mentioned noise needing correction. The LMA will be used to eliminate the noisy data, and an orbit will be reproduced using the range-rate between the simulated orbiting object and a simulated ground station.

2. ORBITAL DETERMINATION FROM GROUND SITE OBSERVATIONS

To establish the trajectory of an object through space, six independent state parameters are necessary. For example, the orbit of a satellite about the earth can be determined by the cartesian state vector, which gives x, y, and z components for the radial and velocity vectors. For this experiment, observable data will be calculated from the orbit propagated from a state vector. One of these observables, range rate, will be calculated using the Doppler signal from the satellites under investigation. For comparison, additional observables will be included in alternative test cases for this orbit determination problem.

The present chapter discusses theory relevant to orbital determination using ground site observations. The first part of this chapter discusses the Julian Date system and sidereal time. These methods of timekeeping simplify later calculations. Next, the Earth-Centered Inertial reference frame is converted to a frame local to the surface of Earth. Then, the theory of orbital determination using the described independent quantities is discussed. Finally, because the trajectory of an object travelling through space can be altered by outside forces, orbital perturbations are briefly discussed.

2.1 TIME MANAGEMENT

As with most problems involving kinetics, time is a necessary component when determining the orbit of a satellite with observational data. Unlike the ubiquitous solar time, which tracks the movement of the Sun through the sky, universal time (UT) monitors the passage of the Sun through the meridian of Greenwich, London, where terrestrial longitude is defined as zero degrees. Measuring westward from the Greenwich meridian to the local meridian, local standard time in calculated by adding one hour per each time zone passed.

Like UT, local sidereal time is measured with respect to the meridian at Greenwich, London. Greenwich sidereal time is the time, in degrees, elapsed since the Greenwich meridian travelled through the vernal equinox multiplied by a factor of fifteen. Local sidereal time is calculated by adding longitude ϕ of the ground site to Greenwich sidereal time σ_G, as in the equation [14]

$$\sigma = \sigma_G + \phi \tag{2.29}$$

This Julian Date system is used to calculate Greenwich sidereal time.

2.1.2 Julian Date System

Just one year after the Gregorian calendar was introduced to the world [15], French-Italian astronomer and historian Joseph Scaliger proposed using the Julian period, later to be named the Julian Date system, facilitate time calculations [16]. Scaliger proposed that rather than having both BC and AD eras, there should be a temporal point of origin from which all time can be measured without the need for positive and negative dates. Because the earliest historical records of the time dated to the year 4713 BCE, Scaliger declared noon UT on January 1 of this year to be the beginning of the Julian Date system.

To calculate the Julian day JD, it is necessary to find the Julian day number $J0$ at 0 hours UT [14]:

$$J_0 = 367y - INT\left\{\frac{7[y + INT(m+9)]}{4}\right\} + INT\left(\frac{275m}{9}\right) + d + 1{,}721{,}013.5 \tag{2.30}$$

where y is a year between 1901 and 2099, m is the numerical month, and d is the day of the month. Then,

$$JD = J_0 + \frac{UT}{24} \tag{2.31}$$

The next series of calculations are based on the current Julian epoch, which was noon on January 1, 2000. Termed J_{2000}, this epoch contains 2,451,545 Julian days. Additionally, because there are 365.25 days in a Julian year, it is no surprise that a Julian century has 36,525 days. Time, therefore, can be found with

$$T_0 = \frac{J_0 - 2{,}451{,}545}{36{,}525} \tag{2.32}$$

where T_0 is measured in Julian centuries between the Julian day $J0$ and $J2000$. The dimensionless time T_0 can be used to find Greenwich sidereal time σ_{G0} at 0 hours UT with

$$\sigma_0 = \sigma_{G0} + 360.9856724 \cdot \frac{UT}{24} \tag{2.33}$$

It is important to know that sidereal time must be in the range $0 \leq \sigma \leq 360$. If a value is found outside of this range, an integer multiple of 360 must be added to or subtracted from the value to make it meet this criterion.

2.2 DEFINING THE REFERENCE FRAME

For this experiment, measurements will be taken using an orbit simulated in MatLab. The "truth" orbit will be propagated using a state vector calculated from the Two-Line Element (TLE) of each satellite. From this simulation, frequency shift shall be estimated based on the introduced orbit, as

well as a simulated ground station.

2.2.1 Topocentric Coordinate System

Due to the oblateness of the earth, a line emanating from the center of the earth will only be tangent to the surface at the equator and the poles; therefore, the topocentric coordinate system defined in this section will be used instead of the Earth-Centered, Inertial (ECI) frame, with directional vectors $\boldsymbol{ec_x}$, $\boldsymbol{ec_y}$, and $\boldsymbol{ec_z}$, to facilitate later calculations. Referring to Figure 2.1, The topocentric coordinate system is centered at ground site S, which is a distance R from the center of the earth EC_0. At the site, the East North Up (ENU), or Topocentric Horizon, coordinate system is defined, where $\boldsymbol{enu_x}$, $\boldsymbol{enu_y}$, and $\boldsymbol{enu_z}$ are directional vectors pointing North, East, and Zenith, respectfully [17].
As exaggerated in Figure 2.2, the shape of a cross-section of the earth is taken as an ellipse. In reality, this cross-section is an ellipsoid, but alterations can be made to equations for changes from the terrestrial elevation [17]. The angle between $\boldsymbol{ec_y}$ and **R**, is defined as geocentric latitude λ_c. Because **R** is not parallel to Zenith, a secondary position vector $\boldsymbol{R_\lambda}$ perpendicular to the tangent at S is defined, with its origin ED_0 located on the polar axis. The angle created between $\boldsymbol{R_\lambda}$ and the $\boldsymbol{ec_y'}$ is called the geodetic latitude λ_d. The terrestrial ellipse can be defined by semimajor R_{eq} and semiminor R_p axes [17].

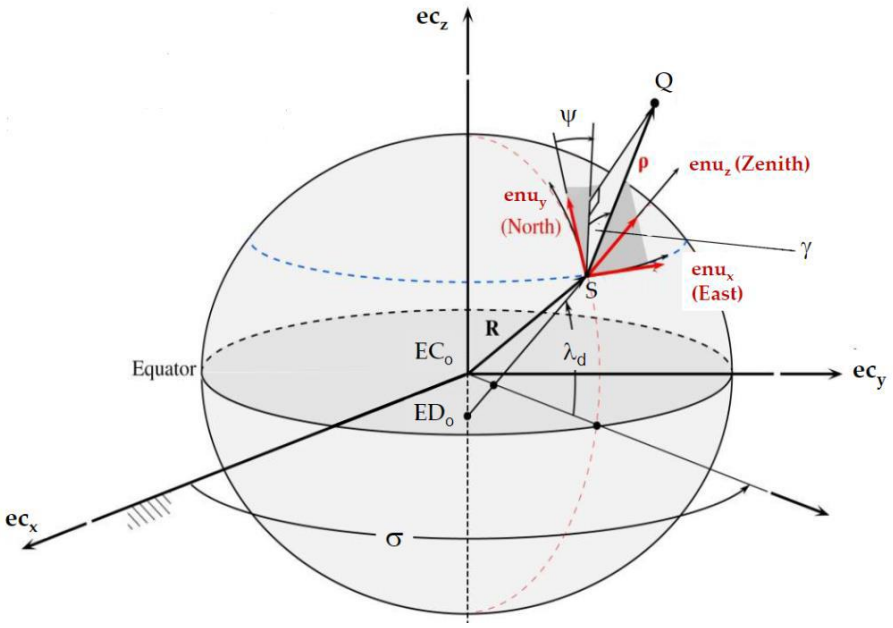

Figure 2.1 Topocentric Coordinate System ([14] as adapted by [17]).

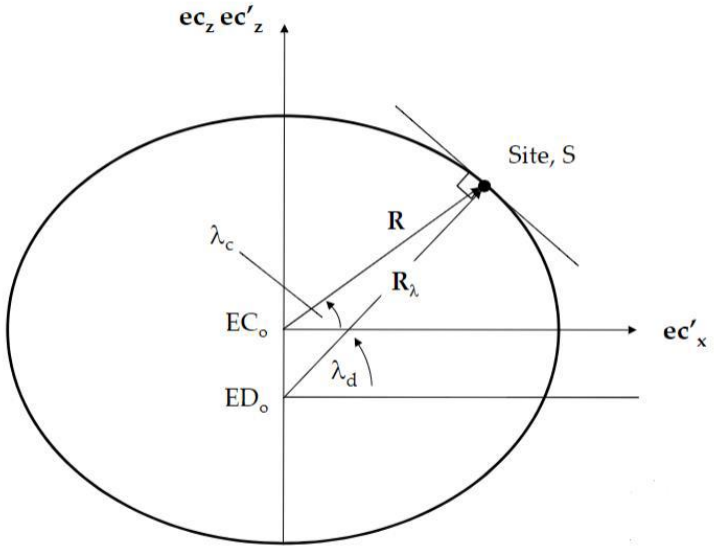

Figure 2.2 Cross-section of the earth [17].

The oblateness and eccentricity of the earth are defined, respectively, by [14]:

$$f_{obl} = \frac{R_{eq} - R_p}{R_{eq}} = 1 - \sqrt{1 - e^2} \tag{2.34}$$

where the relation between e and f_{obl} is

$$e = \sqrt{2f_{obl} - f_{obl}^2} \tag{2.35}$$

The distance from EC_0 to ED_0 is $R_\lambda e^2 \sin^2 \lambda$, where R_λ is defined as [14]:

$$R_\lambda = \frac{R_{eq}}{\sqrt{1 - e^2 \sin^2 \lambda}} = \frac{R_{eq}}{\sqrt{1 - (2f_{obl} - f_{obl}^2) \sin^2 \lambda}} \tag{2.36}$$

The position of S with respect to EC_0 can now be defined as [14]:

$$\begin{aligned} R = {} & (R_\lambda + H) \cos \lambda \cos \sigma \, \boldsymbol{ec}_x + (R_\lambda + H) \cos \lambda \sin \sigma \, \boldsymbol{ec}_y \\ & + [(1 - f)^2 R_\lambda + H] \sin \lambda \, \boldsymbol{ec}_z \end{aligned} \tag{2.37}$$

where H refers to height of the ground station with respect to the reference ellipsoid.

2.3 THE DOPPLER EFFECT

The Doppler effect is a phenomenon that increases or decreases the observed frequency of a wave due to the relative velocity between the source of the wave and the observer. This frequency shift is most recognizable in the siren of a passing ambulance, but it can also occur in light and radio signal reception. Measuring the frequency of stellar light, astronomers can determine if a star is moving

with respect to the earth. Likewise, a received radio signal from an orbiting satellite will have a frequency shift, though steps are typically taken to correct this effect instantaneously. To attain a better understanding of the Doppler effect, the components of frequency are examined.

The velocity of a wave V_{wave} from a stationary source can be measured in terms of wavelength λ and transmitted frequency f_t with the equation

$$V_{wave} = \lambda f_t \tag{2.38}$$

If the source has a relative velocity with respect to an observer, the observed frequency f_r is shifted from f_t, as per the equation:

$$f_r = f_t \left(1 + \frac{V_{source}}{V_{wave}}\right) \tag{2.39}$$

if the source is moving at velocity V_{source} towards an observer, or

$$f_r = f_t \left(1 - \frac{V_{source}}{V_{wave}}\right) \tag{2.40}$$

In the case of signal measured from a passing satellite, V_{wave} is hereby referred to as the speed of light c.

With a simulated radio signal and an orbit propagator, equations (2.11) and (2.12) will be used later in this paper to determine the range-rate of the satellite.

2.4 ORBITAL PERTURBATIONS

In a basic two-body problem where the center of mass is dominated by a massive spherical body, an unaltered orbit could be achieved by the secondary body, though, in reality, none of these assumptions hold true. Adding complexity to the two-body problem, there are four forces that alter the orbital elements of a satellite: third-body perturbations, perturbations due to non-spherical planet, atmospheric drag, and solar radiation pressure [18].

The first orbit influencing force, third-body perturbations, is caused by the sun and moon. These bodies induce periodic changes to each orbital element of the satellite. Additionally, secular variations are experienced by the longitude of the ascending node and the mean anomaly due to the gravitational presence of these bodies.

The second force in this list is due to the ellipsoidal shape of the earth. While the planet is typically modelled as having a spherical shape, a better estimate shows that more mass is found along the equator, leaving a flattening effect at the poles. To accurately predict an orbit, zonal coefficients J_n are used to form a geopotential function.

The oblateness of the earth dominates the geopotential expansion. In this expansion, the J_2 term represents perturbations caused by this flattening. This force results in secular changes in the longitude of the ascending node and the argument of perigee [18].

The third orbit perturbing force is atmospheric drag. As a body moves through a fluid, momentum is lost from the body and imparted to particles in the fluid. This exchange of momentum causes a decrease in the velocity of the body. In the case of a satellite, a decrease in velocity means orbital decay. Fluctuations in atmospheric density are caused by varying solar activity. During periods of high solar activity, altitudes in the range of 500 – 800 km can have an atmospheric density around two orders of magnitude greater than seen during low solar activity [18].

The final force in this list is caused by solar radiation pressure. In the lower atmosphere,

atmospheric drag is the most influencing force on orbital elements, but at altitudes greater than 800 km, solar radiation pressure becomes the greater force [18].

3. ORBIT SIMULATION AND DATA ACQUISITION

Paramount to the success of orbital determination is accurate data acquisition. While the data used for the present paper rely on a simulated orbit, realistic scenarios present errors which must be accounted for. The goal of this chapter is to provide the method used to calculate an accurate orbit using noisy data. Reasons for this method choice will be discussed, as well as corrections for the orbital perturbations outlined in the previous chapter.

The first part of the chapter will give a brief history of statistical orbital determination (OD). Next, a discussion on how initial orbital data is obtained using MatLab software is given. The corrections used for orbital perturbations will be examined. To accurately predict the future location of an orbiting body, all perturbations mentioned in chapter 2 must be considered. After this, the sources of data error and the procedures used to reduce these errors will be explained.

3.1 A BREIF HISTORY OF ORBITAL DETERMINATION

While astronomers have contemplated motion through space for millennia, it wasn't until Johannes Kepler (c.1610), a German mathematician, astronomer, and astrologer discovered that not all orbits are circular that true statistical OD began [14]. Kepler was a student of a wealthy astronomer, Tycho Brahe, whose beliefs placed Earth at the center of the universe. When asked by Brahe to determine the orbit of Mars, Kepler eventually discovered the elliptical shape of the orbit. Later, mathematicians would proceed with Kepler's work to give OD a much more defined subject.

The formulation of least-squares (LS) algorithms, discussed further in Section 3.3, was first imagined by Karl Friedrich Gauss, a German mathematician, in 1795. As with many discoveries, controversy ensued when French mathematician Adrien-Marie Legendre also discovered the method of LS and became the first to publish his findings in 1806 [19]. The idea would eventually be

attributed to Gauss, who published his own works for OD methods in 1809. While Gauss and Legendre were trying to figure out who discovered LS, another hallmark achievement was accomplished.

In 1801, the Ceres comet was rediscovered after astronomers used observations to predict its location. This was the first time OD was used to locate an orbiting body [20]. Over next two decades, many mathematicians worked to refine the work done by Gauss and Legendre, though the Gaussian method is still widely used today.

During the Cold War, the United States first used its ability to observe satellites through radio frequency when a Naval scientist, Richard Anderle, used the Doppler method to derive range-rate of the Sputnik I. After Sputnik II was launched, satellite tracking methods were refined further leading to modern orbit determination, to which Anderle is accredited. Study of the Sputnik satellites allowed improved estimates for J_2 perturbations and the Earth's gravitational field [20].

3.2 DATA ACQUISITION

The data analyzed in this paper will be simulated using both the two-line element set (TLE) of a given satellite and MatLab software. Using an orbit propagator, the TLE set will be used to form an initial estimate for the orbit of a satellite. For this simulation, slant range-rate between the satellite and an input ground station will transformed to frequency-rate (Doppler shift) by rearranging the equation [2]:

$$f_r = f_t\left(1 - \frac{\dot{\rho}}{c}\right) \tag{3.1}$$

where:

f_r = the frequency received at the ground station

f_t = the frequency transmitted from the satellite

$\dot{\rho}$ = the line-of-sight (LOS) range-rate between the ground station and satellite

c = the speed of light.

Finally, $f_r(t)$ will be recorded from multiple passes to derive a new TLE set. Visibility windows will be calculated on a satellite to satellite basis, though passes with a low maximum elevation will be avoided. Additionally, because satellite tracking software (Gpredict) will be used for initial TLE sets and later for comparison, a simulated ground station will be present in both Gpredict and MatLab, using the longitude and latitude of San Jose State University for coordinates: 37.3352° N, 121.8811° W.

Because a real-life measurement of the Doppler shift would contain noise, the simulated orbit will have a Gaussian noise added.

3.2.1 Sources of Error

Due to the complexity of satellite communications, error sources are prevalent. According to [20], there are three main sources of data error: Instrument errors, measurement errors, and mathematical modeling errors.

Instrument errors can be caused by the operator or hardware. For instance, if the operator should decide to record the pass of a satellite with a low maximum elevation, meaning a short pass just over the horizon, atmospheric distortion can induce large errors. Hydrometeors such as clouds and rain can also cause errors through attenuation of the signal [21]. Errors introduced from hardware can emanate from poorly maintained sensors or improper wiring [20].

Measurement errors are produced from biases, non-random time-varying errors (drift), and noise

[20]. Vallado and McClain [20] state biases are “a constant offset from the true value.” In astrodynamics, it is common enough to assume this bias is zero. Drift is known as a slow variation to data over time. The largest contributor to drift is clock instabilities in the satellite, which can be cause by temperature differentials [20]. Due to the short windows of visibility, drift will be negligible in the present study. Noise is a statistical indication, or standard deviation of varying data around the measured average.

Noise errors can stem from several sources. The on-board oscillator can degrade accuracy of the Doppler shift measurements without short-term stability [22]. According to Bart Root (personal communication, 2018), a lecturer at the Delft University of Technology in Delft, Netherlands, this makes tracking smaller satellites (e.g., CubeSats) through one-way Doppler measurements difficult due to the cheap oscillators used. Additional sources of noise may stem from surface radio frequency (RF) emissions or other air/space vehicles.

Mathematical modeling errors happen during data processing. This can mean incorrectly entered data, typos in coding, and general misunderstanding of data field (Bart Root, personal communication, 2018). The best way to avoid modeling errors is to take care in both data recording and coding.

3.2.2 Two-Line Elements (TLEs)

As mentioned in chapter 2, six elements are needed to accurately predict an orbit. While the present study focuses on the use of range-rate information supplied through Doppler data, initial knowledge of the location of the satellite is necessary, according to Gauss, who states in his book, *Theoria Motus* (as translated in [19]), “… this problem [of accurate OD] can only be properly undertaken when an approximate knowledge of the orbit has been already attained.” Gpredict, a real-time satellite tracking application, will be used to supply TLE data to MatLab to form an initial estimate

of an orbit for a given satellite.

It is worth mentioning, to accurately simulate an orbit, for the purposes of this study, the orbital perturbations must be included in the simulation model. This requires a code that will factor in perturbing forces when numerically iterating an orbit.

3.3 SOLUTION TO THE ORBIT DETERMINATION PROBLEM

Algorithms to compute best fit in linear regression models are standard in most modern calculator software (e.g., Microsoft Excel, MatLab, etc.); however, when attempting to solve nonlinear systems, user work is required. The objective of the current paper is to find some trajectory equation, $y = f(x)$, to model the orbit of a satellite. Gauss suggests this problem can be solved by finding the *most probable values*. Gauss writes in *Theoria Motus* (translated by [19]):

> *... the most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum."*

With this definition, Gauss was able to find a solution to non-linear systems.

The study conducted in the present paper requires an algorithm which will minimize the sum of squares of the residuals in a data set. This section is dedicated to defining the non-linear LS solution to statistical OD. The majority of the information found in this section comes from Gavin [23] and Nash [24].

3.3.1 Finding a Least-Squares Algorithm

Since Gauss and Legendre discovered the method for minimizing the sum of squares of equation residuals [23],

$$S(\boldsymbol{x}) = \sum_{i=1}^{N} [f_i(\boldsymbol{x})]^2 \tag{3.2}$$

where N is the number of data points and **x** is a vector of parameters x_j, j = 1,2,3,…,n. The vector of residuals **f** is found by assembling the N functions $f_i(\boldsymbol{x})$, where i = 1,2,3,…,N, resulting in [24]

$$S(\boldsymbol{x}) = \boldsymbol{f}^T \boldsymbol{f}. \tag{3.3}$$

Numerous mathematicians have made alterations to Gauss's method for both better understanding of the problem and to decrease computational cost.

Also known as the damped least-squares (DLS) method, the Levenberg-Marquardt algorithm (LMA) is one such modified algorithm that solves curve fitting problems. The LMA is a combination of the steepest descent method (also known as gradient descent method) and Gauss-Newton method.

3.3.1.1 The Steepest Descent Method

Suited for general minimization problems, in the steepest descent method (SDM), parameter values are updated in the "downhill" direction (i.e., towards the minimum). This method is best suited for problems with trivial objective functions [23]. Starting with the gradient 2**v**(**x**) of S(**x**), the SDM steps down along the gradient [24]. Using *t*, the step length along the step path, it is shown that

$$S(\mathbf{x} - \mathrm{t}\mathbf{v}) < S(\boldsymbol{x}) \tag{3.4}$$

where $S(\boldsymbol{x})$ was defined in Eqn. (3.2).

The SDM uses $(\mathbf{x} - \mathrm{t}\mathbf{v})$ in place of **x** and iterates forward from a new position. This process is carried on until a t no longer exists for Eqn. (3.4), at which point the operation has converged.

3.3.1.2 The Gauss-Newton Method

In the Gauss-Newton method (GNM), a sum-of-squares objective function is minimized. This method assumes the desired function is approximately quadratic near the optimal solution [23]. The GNM allows faster convergence than the gradient descent method when solving moderately sized problems.

The GNM takes advantage of the fact that the gradient **v**(**x**) must be zero at the minimum. That's to say, the functions $\boldsymbol{v}_j(\boldsymbol{x})$, j = 1,2,3,…,m, create a nonlinear set of *m* functions with *m* unknowns $\boldsymbol{x}$ such that [24]

$$\boldsymbol{v}(\boldsymbol{x}) = 0. \tag{3.5}$$

The solution to Eq. (3.5) lies on the local minimum or maximum of the function $S(\boldsymbol{x})$. Further analysis of Eqs. (3.2-3.3) suggests gradient components [24]

$$2v_j(\boldsymbol{x}) = 2\sum_{i=1}^{N} f_i(\boldsymbol{x})\delta f_i(\boldsymbol{x})/\delta x_j \tag{3.6}$$

which leads to

$$v_j(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}) J_{ij}(\boldsymbol{x}) \tag{3.7}$$

restated in matrix form as

$$\boldsymbol{J}^T \boldsymbol{q} = -\boldsymbol{v} = -\boldsymbol{J}^T \boldsymbol{f} \tag{3.8}$$

with Jacobian matrix $\boldsymbol{J}$ defined as

$$\mathrm{J_{ij}} = {}^{\delta \mathrm{f_i}}\!/_{\delta \mathrm{y_j}} \tag{3.9}$$

otherwise stated as

$$\frac{\text{perturbed state} - \text{actual state}}{\text{amount this state was perturbed by}}.$$

Simplification of Eq. (3.5) is required; thus, an approximation must be made. To find this approximation the Taylor expansion of $v_j(\boldsymbol{x})$ about **x** is examined [24]

$$v_j(\boldsymbol{x} + \boldsymbol{q}) = v_j(\boldsymbol{x}) + \frac{\sum_{k=1}^{n} \boldsymbol{q}_k \, \delta \boldsymbol{v}_j(\boldsymbol{x})}{\delta \boldsymbol{x_k}} + (terms\ in\ \boldsymbol{q}^2). \tag{3.10}$$

Assuming $v_j(\boldsymbol{x} + \boldsymbol{q})$ is the solution, and thus equal to zero, and the terms $q_k q_j$ in q^2 are negligible, then

$$\frac{\sum_{k=1}^{n} q_k \, \delta v_j(\boldsymbol{x})}{\delta x_k} = \; -v_j(\boldsymbol{x}) \tag{3.11}$$

for each element in *j*. Thus, incorporating Eqns. (3.7) and (3.9),

$$\frac{\delta v_j(\boldsymbol{x})}{\delta x_k} = \; \sum_{i=1}^{N} \left[J_{ik}(\boldsymbol{x}) J_{ij}(\boldsymbol{x}) + \; f_i(\boldsymbol{x}) \frac{\delta^2 f_i(\boldsymbol{x})}{\delta x_j \delta x_k} \right] \tag{3.12}$$

The GNM iterates forward, using (**x + q**) in place of **x** and repeats the process until the value of **q** falls below a prescribed tolerance or

$$S(\boldsymbol{x} \; + \; \boldsymbol{q}) \geq \; S(\boldsymbol{x}\,). \tag{3.13}$$

3.3.2 Marquardt's Method

Using both the gradient descent and Gauss-Newton methods, the LMA changes based on the value of an algorithmic parameter λ, as seen in the equation (adapted from [24]):

$$(\boldsymbol{J^T J} + \lambda \boldsymbol{D^2})\boldsymbol{q} = \boldsymbol{-J^T f} \tag{3.14}$$

where:

$\mathbf{J^T}$ = transpose of $\boldsymbol{J}$

D = a diagonal matrix with positive diagonal elements

f = column vector of residuals

$\boldsymbol{q}$ = vector of increments of **x.**

The vector of residuals **f** is defined as the difference between observed values y_{o_i} and calculated values y_{c_i} as seen in [24]

$$f_i(\boldsymbol{x}) = y_{o_i}(i, \boldsymbol{x}_{o_i}) - y_{c_i} \tag{3.15}$$

where $\boldsymbol{x}_{o_i}$ **is** the observed state vector. Residuals are defined in this vector as distances of data points from the mean curve. Because the matrix $\mathbf{J^T J} + \lambda \mathbf{D^2}$ is always positive definite, Cholesky decomposition can be used to increase the efficiency of the LMA by breaking the matrix into a product of two matrices: a lower triangular matrix and its conjugate transpose. For a more in-depth examination of Cholesky decomposition, see literature by Higham [25].

In the LMA, when λ is very small compared to the norm of $\boldsymbol{J^T J}$, $\boldsymbol{q}$ tends towards the Gauss-Newton solution, whereas when λ becomes much larger in comparison to this norm, the steepest descents solution is calculated. Starting the iteration with λ = 0.1 is suggested by Marquardt. Throughout the iteration, λ should be decreased by a factor of 10 if the preceding solution **q** was found to be

$$S(\boldsymbol{x} - t\boldsymbol{v}) < S(\boldsymbol{x}) \tag{3.16}$$

as was seen with the steepest descent method. Should

$$S(\boldsymbol{x} + \boldsymbol{q}) \geq S(\boldsymbol{x}) \tag{3.17}$$

λ should be increased by the same factor followed by repeating Eqn. (3.14).

Marquardt's modified LS algorithm was tested for its curve fitting capability. The decaying exponential function

$$y(i, \boldsymbol{x}) = c_1 + c_2 * e^{c_3 x_i} \tag{3.18}$$

was used to test the algorithm, and the results of this test can be seen in Figure 3.1.

The hitherto information in this paper form the basis for finding a solution to the Doppler shift problem. Using Marquardt's method of damped least-squares curve-fitting, the sum-of-squares of the residuals caused by noise, which will be added as random noise in the simulation, will be reduced, with the hopes that they fall to zero.

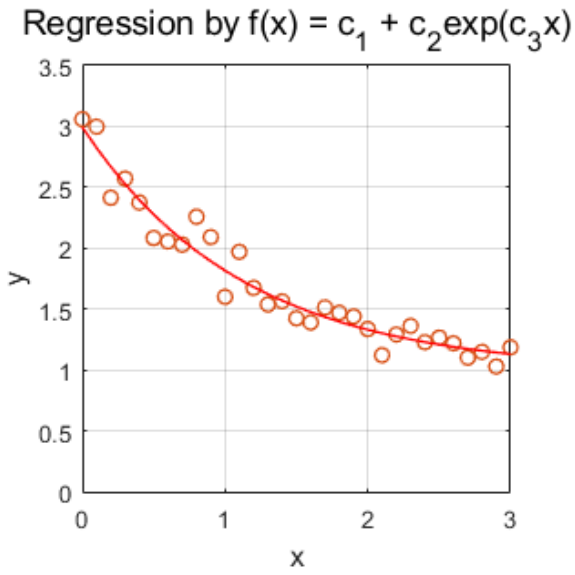

Figure 3.1 Solution of a decaying exponential function. Circles on the graph represent noisy data. The Solid red line is the LMA solution.

In the next chapter, the Levenberg-Marquardt algorithm, as it is applied to the present study, will be discussed in detail. The method used to provide simulated data will be established and the software used to fit an estimated orbit will be outlined.

4. MARQUARDT DAMPED LEAST SQUARES FILTER DESIGN

The Levenberg-Marquardt Algorithm (LMA), also known as the Marquardt algorithm, is acclaimed for its proficiency in orbit estimation. The LMA is robust and allows for a higher degree of error in measured data if your initial estimate of the state vector is reasonable [26]. The Levenberg-Marquardt Filter (LMF) used in the present study borrows from the LMF outlined by Nash [24] with modifications presented by Transtrum and Sethna [27]. The algorithm provides a "best estimate" for state vector **x** when provided with noisy data and an initial "estimated" state vector. As it was applied to the present investigation, data was collected from a simulated orbit, then the orbit was perturbed to provide an initial estimate to the system.

4.1 ACQUIRING DATA

Using NORAD Two-Line Element (TLE) data retrieved from Celestrak, the radial and velocity components of a given satellite are calculated using the Simplified General Perturbations 4 (SGP4) propagator, which can be found online in many computing languages. The MatLab version of the SGP4 propagator used in the present paper, written by Mahooti [28], can be found on the MathWorks website. Initially, azimuth and elevation data taken from Gpredict satellite tracking software are used to generate the "truth" state vector.

Gpredict (GP) is a free, downloadable satellite tracking application and was the original source of data needed to produce the state vector of a satellite in orbit at a given time. In this method, right ascension and declination angles were transferred from GP to MatLab for processing. Using Gauss's method of preliminary orbit determination, a state vector can be produced using three sets of right ascension and declination angles along with their corresponding time. As the first orbit being tested for this experiment belonged to the ISS, GP data was enough to generate a state vector; however,

problems arose when examining satellites in a larger orbit. Curtis [14] explains the Gauss method in detail and mentions that the time between the measured angles should be small. While it is possible to manipulate the output from GP to an extent, the software did not provide data with short enough time intervals. This became apparent as state vectors generated for larger orbits (e.g., orbits of NAVSTAR and MOLNIYA satellites) were too flawed to provide adequate testbeds. Learning from this, it was determined that using TLE sets was the best alternative; however, because a user may want to produce an estimated orbit from observed angles, future work to the program used in this paper will provide the means to accomplish this goal.

Once a state vector is produced form TLE data, it is propagated for several steps, allowing the collection of new state vector $\boldsymbol{x}_i$ for *i = 1,...,n,* where n is the number of observational sets of data recorded. Depending on which data is chosen to be simulated, calculations of azimuth A_i, elevation a_i, and/or range rate $\dot{\rho}_\iota$ are stored in a matrix along with their corresponding time t_i. Azimuth is calculated as

$$A = \tan^{-1}\left(\frac{\rho_x}{\rho_y}\right) \tag{4.1}$$

where ρ_x and ρ_y are x and z components of the range vector. If A is negative, 360^0 is added to its value. Elevation is calculated as

$$a = \tan^{-1}\left(\frac{\rho_z}{\sqrt{\rho_x^2 + \rho_y^2}}\right) \tag{4.2}$$

Range rate is calculated using

$$\dot{\rho} = \frac{\boldsymbol{\rho} \cdot \dot{\boldsymbol{\rho}}}{\rho} \tag{4.3}$$

where

- $\boldsymbol{\rho}$ is the range vector $\boldsymbol{\rho} = \boldsymbol{r} - \boldsymbol{R}$
- $\dot{\boldsymbol{\rho}}$ is the ECI frame time derivative of $\boldsymbol{\rho}$ $\dot{\boldsymbol{\rho}} = \boldsymbol{v} - (\boldsymbol{\omega}_E \, x \, \boldsymbol{R})$
- ρ is the magnitude of $\boldsymbol{\rho}$ $\boldsymbol{\rho} = \sqrt{\rho_x^2 + \rho_y^2 + \rho_z^2}$

$\boldsymbol{R}$ and **r** are the ground site and satellite positions with respect to the center of the earth and $\boldsymbol{\omega}_E$ is the angular velocity of the earth about its $\boldsymbol{ec_z}$ axis.

Process noise vector $\boldsymbol{w}_i$ is then added to the data to resemble realistic measurements. While there are various options for state vector elements to fit using least-squares methods, two of these element sets are more commonly used [27].

Common element choices for state vector **x**, as it applies to orbit determination, are the cartesian and equinoctial sets of elements [29]. The cartesian state vector contains position and velocity components $\{x, y, z, \dot{x}, \dot{y}, \dot{z}\}$, typically studied in the Earth-Centered, Inertial (ECI) frame or, as is the case in this study, the topocentric horizon frame, as shown in Figure 2.1. The equinoctial elements are calculated using the classical Keplerian elements. The equinoctial elements are not used in the present paper. There are many sources for information on the equinoctial elements, such as Battin [30].

4.2 LEVENBERG-MARQUARDT LEAST SQUARES FILTER

In this section, the Levenberg-Marquardt Filter (LMF) as it applies to this study is detailed. First, a

brief discussion on the formation of the LMF is given, elaborating on the description given in Chapter Three. Then, modifications proposed by [27] are explained.

In the study, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Levenberg [32] proposed damping of parameter increments to improve first-order Taylor series approximations when a flaw was noticed "standard" methods. In past procedures [32] least squares algorithms using linear approximations found updated values for estimated parameters, but the algorithm would fail if the new values were not sufficiently close to the initial estimate. This is because the algorithm may neglect higher order term, which leads to a larger sum of squares of the residuals. Thus, Levenberg [32] determined that finding function residuals under damped conditions was a beneficial alternative. This was done by including a damping parameter λ to the least-squares system. The purpose of λ is to change the eigenvalues of the matrix $\boldsymbol{J}^T\boldsymbol{J} + \lambda\, \boldsymbol{D}^T\boldsymbol{D}$, where $\boldsymbol{D}^T\boldsymbol{D}$ is a scaling matrix, to be equal to λ or greater [27]. After Levenberg's [32] development of this novel approach to least-squares methods, Marquardt [33] added modifications.

Marquardt [33] produced additions to Levenberg's [32] method for least-squares in his paper, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." Noting that the two methods used most often for non-linear least-squares estimation, that is, iteratively correcting parameters of the Gauss-Newton Method (GNM) and various methods of using the Steepest-Descent Method (SDM), often fail. The Taylor series method fails due to divergence of successive iterations. SDM failures are caused by slow convergence just a few iterations in [33]. Thus, Marquardt [33] proposed a "maximum neighborhood" method. This method switches between the GNM and SDM based on the maximum neighborhood where sufficient representations of the non-linear model can be found from the truncated Taylor series. As was discussed in Chapter 3, the Marquardt method switches between the GNM and SDM by comparing values of the sum of squares of residuals of successive

steps to those of previous steps. Should the sum of squares resulting from the corrected set of parameters be greater than the previous sum of squares, the algorithm favors the SDM. Conversely, if the new sum of squares is less than the previous iteration, the method proceeds with the GNM. The algorithm, once supplied with a vector of initial estimates **p** and recorded data, iteratively repeats the following steps [27]:

1. Calculate new data and Jacobian values based on the updated parameters.
2. Calculate new Marquardt parameter (damping term) λ and scaling matrix $\boldsymbol{J}^T\boldsymbol{J}$.
3. Calculate parameter step $\delta\boldsymbol{p}$ using Eqn. (3.14).
4. Test the updated parameter set ($\boldsymbol{p} + \delta\boldsymbol{p}$) by calculating the residuals of new data to the previous sum of squares.
5. If the new sum of squares of residuals is less than the previous sum of squares, $\delta\boldsymbol{p}$ is accepted.
6. Cease iterations if convergence criteria are met or a predetermined maximum iteration count has been reached.

There are various methods used to determine the damping parameter and scaling matrix.

4.2.1 Damping Parameter and Scaling Matrix

There are two classes of methods used to determine the damping parameter λ: direct and indirect. Direct methods imply that if $\boldsymbol{\delta p}$ results in a smaller sum of squares of residuals, λ is decreased by some factor. Conversely, if $\delta\boldsymbol{p}$ is rejected, λ is increased by some factor. The factor to increase or decrease λ in this method are decided by the user. It is determined that choosing a λ decreasing factor lower than the λ increasing factor leads to better results [27]. Transtrum and Sethna [27] suggest using a factor of 5 when decreasing λ and a factor of 1.5 to raise λ for larger problems. For

smaller problems, decreasing and increasing factors of 3 and 2, respectively, work best [27]. Nash [24] suggests making decreasing and increasing factors 0.4 and 10, respectively.

To use the indirect method for determining λ, a step size Δ is first determined. The damping parameter that ensures $|\delta \boldsymbol{p}| \leq \Delta$ is then found. As this method for determining λ will not be used in this study, greater detail on the matter will not be provided in the present paper. For further information on this indirect method for finding λ, see literature such as Moré [34]. Transtrum and Sethna [27] determined that some problems perform better using the direct method for determining λ, while others favor the indirect method.

There are several options when choosing the scaling matrix $\boldsymbol{D}^T\boldsymbol{D}$. While Levenberg first determined the scaling matrix be the identity matrix **I** [27], both Levenberg [32] and Marquardt [33] settled on using the diagonal entries of $\boldsymbol{J}^T\boldsymbol{J}$ [24]. Moré [34] determined that the optimal scaling matrix would be a diagonal matrix which updates its entries with the largest diagonal entries of $\boldsymbol{J}^T\boldsymbol{J}$ encountered through the duration of the run.

4.2.2 Gain Factor

To ensure a faster convergence using the LMF a gain factor β is used to control which corrections are accepted. This gain factor is formulated as follows (adapted from [35]):

$$\beta = \frac{F(\boldsymbol{x}) - F(\boldsymbol{x}+\boldsymbol{q})}{L(\mathbf{0}) - L(\boldsymbol{q})} \tag{4.1}$$

where F(**x**) is the data function evaluated with parameter vector **x** and F(**x** + **q**) is the data function evaluated with parameter correction vector **q**. The denominator is evaluated as

$$L(\mathbf{0}) - L(\boldsymbol{q}) = -\boldsymbol{q}\boldsymbol{J}^{\boldsymbol{T}}\boldsymbol{f} - \frac{1}{2}\boldsymbol{q}^{\boldsymbol{T}}\boldsymbol{J}^{\boldsymbol{T}}\boldsymbol{J}\,\boldsymbol{q} \tag{4.2}$$

$$= -\frac{1}{2}\boldsymbol{q}^{\boldsymbol{T}}[2\boldsymbol{F}' + \boldsymbol{J}^{\boldsymbol{T}}\boldsymbol{J} + \lambda\boldsymbol{I} - \lambda\boldsymbol{I})\boldsymbol{q}] \tag{4.3}$$

$$= \frac{1}{2}\,\boldsymbol{q}^{\boldsymbol{T}}(\lambda\boldsymbol{q} - \boldsymbol{F}') \tag{4.4}$$

where

$$\boldsymbol{F}' = \mathbf{J}^{\mathrm{T}}\boldsymbol{f} \tag{4.5}$$

When β falls below a predetermined value ϵ_4 the parameter correction is rejected and λ in increased. Otherwise, the parameter correction is accepted and λ is decreased.

4.2.3 Broyden Rank-1 Jacobian

Each time a correction is accepted to the parameter vector, the Jacobian matrix is updated so new corrections can be determined. This Jacobian matrix is typically evaluated as

$$\mathrm{J_{ij}} = {}^{\delta \mathrm{f_i}}\!/_{\delta \mathrm{y_j}} \tag{4.6}$$

or

$$\boldsymbol{J} = \frac{\boldsymbol{F}(\boldsymbol{x} + \delta\boldsymbol{x}) - \boldsymbol{F}(\boldsymbol{x})}{\delta\boldsymbol{x}} \tag{4.7}$$

As calculating **J** each time can become computationally expensive, Transtrum and Sethna [27] suggest using an alternative update method set forth by Broyden [36]. Broyden [36] determined that a quasi-Newton root finding method that updates **J** with first derivatives on the first iteration, then alternates between reevaluating **J** with a rank-1 update. This Broyden rank-1 update is written as (adapted from [36])

$$\boldsymbol{J}_k = \boldsymbol{J}_{k-1} + \frac{(F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k-1}) - \boldsymbol{J}\boldsymbol{q})\boldsymbol{q}^T}{\boldsymbol{q}^T\boldsymbol{q}} \tag{4.8}$$

where subscript k indicates the current step and k-1 represents the previous step.

4.2.4 Convergence and Stopping Criterion

If the software running the least-squares estimator is not told when it is a good place to stop, it may continue iterating indefinitely. This implies that either the parameters have converged to a solution and further iterations cease to produce worthwhile results, or the function is not solvable under the given conditions and further iterations produce worthless results. Convergence and stopping criteria are added to the least-squares program to ensure further calculations are not carried out once the criteria are met. It is suggested to use the following convergence criteria (adapted from [35]):

1. $\left|\left|\boldsymbol{F}'(\boldsymbol{x})\right|\right| \leq \epsilon_1$
2. $\frac{||\boldsymbol{q}||}{\left|\left|\boldsymbol{x}+\boldsymbol{\alpha}\right|\right|} \leq \epsilon_2$, where α is greater than zero
3. $k \geq k_{max}$

The first criterion stops the program should the highest absolute value in the gradient vector be less than a user specified value ϵ_1. This will be called the gradient convergence criterion. The second criterion stops the program if the highest absolute value of the correction vector divided by its counterpart in the absolute value of the parameter vector plus α, a small number greater than zero, is less than the user specified ϵ_2. The third criterion stops the program should the iteration count meet or exceed some predetermined value.

4.3 BUILDING THE SOFTWARE

For this study, MatLab was used to write the orbit determination software. To use the LMF, a program was first designed to calculate the "truth" state vector from the TLE set of a given satellite. Next, a program to propagate the state vector for the pass duration of the satellite is used, and simulated data is collected. Then, a vector of perturbing elements is added to the "truth" orbit to simulate an initial estimate of the state vector. Finally, the "estimated" state vector and simulated data are passed to the LMF to find the "best estimate" of the orbit fitting the supplied data. The setup of the LMF and subroutines are set up similarly to many other programs using least squares algorithms.

Three main routines are required when testing a least-squares filter: the least-squares filter, a data acquisition function, and a testing program. The least-squares filter, in this case, the LMF, is built to handle a variety of data fitting applications. Next, a test program is created to declare a vector of initial estimated parameters, system constants, and filter options. Additionally, the test program reads a file consisting of data and times the data was taken. Finally, the data acquisition function uses the vector of parameters and the vector of times corresponding to the times of real data measurement to simulate data.

4.3.1 Test Program

In the algorithm test program, “Orbit100.m”, users can alter testing options before state vectors are produced. Satellites that are currently available for testing are NAVSTAR-77, MOLNIYA 3-50, and the ISS. The test program is broken up into seven sections.

Satellite selection and orbit propagator options can be set in the first section of this program. The TLE set for the selected satellite downloads automatically when the program is started. While only the mentioned satellites are available for the user’s convenience, additional satellites can be tested with a text file containing the TLE of the desired satellite. Also, in the first section, the choice of which observation set to use can be made.

For case one, only range rate observations are calculated. The focus of this paper is to study the ability of the LMF to estimate an orbit purely from the Doppler data received from the downlink signal of a satellite. Thus, the range rate “observations” used in this study are meant to simulate data derived from Doppler data by solving Eqn. (1.23) for $\dot{\rho}$:

$$\dot{\rho}_i = \Delta f_i \cdot \left(-\frac{c}{f_t}\right) \tag{4.9}$$

where $\Delta f_i = f_{r,i} - f_t$ (frequency received at the ground site at time t_i minus signal frequency transmitted by the satellite) is the Doppler shift at time t_i and c is the speed of light. As the observations of Doppler shift are simulated in this paper, another method of calculating range rate is desired. This method was discussed in Section 4.1.

For case two, azimuth and elevation calculations can be taken along with the range rate. Alternatively, testing can be done using only azimuth and elevation in case three. While the focus of

this paper is on using Doppler data (range rate) for orbit estimation, these additional cases will serve well for comparison. Methods used to calculate azimuth and elevation were covered in Section 4.1.

Section two of the main program allows changing of options used in the LMF. The options included are:

- bdx - Small perturbation value used for Jacobian calculation ($\delta\mathbf{x}$ in Eqn. 4.7)
- lambda - The Marquardt scaling parameter λ
- incr - A value to increase lambda
- decr - A value to decrease lambda
- maxIter - Determines maximum iteration count
- eps1 - Gradient convergence criteria ϵ_1
- eps2 - Parameter convergence criteria ϵ_2
- eps3 - Root mean square convergence criteria ϵ_3
- eps4 - state correction acceptance criteria ϵ_4

In section three of "Orbit100.m", the start time for the satellite pass is entered. Satellite tracking software, such as GP, or internet databases can be used to find satellite flyby times. Time is entered in Universal Time (UT). In section four of "Orbit100.m", the user can change the position of the ground site. Currently, the simulated ground site shares the location of San Jose State University. Constants and coefficients are read into the program from exterior files in sections 5 and 6 of "Orbit100.m."

In sections 5 and 6 of "Orbit100.m," files containing constants and coefficients used in the orbit generator are loaded. These files contain Earth Orientation Parameters (EOP), the GRACE gravity model (GGM03S), and NASA JPL Development Ephemerides (DE430). Finally, in section 7 of Orbit100.m, data is generated, and the "estimated" orbit is produced.

4.3.2 Orbit and Data Generator

The orbit propagator used in this study was part of a package put together by Meysam Mahooti. The unaltered version of "High Precision Orbit Propagator" (HPOP) can be found on the MathWorks File Exchange. Mahooti's HPOP was chosen for its ability to model the variety of forces that act on Earth-orbiting satellites. These forces are:

- Gravity field of the earth
- Gravity of the solar system planets
- Drag effect
- Solar radiation pressure
- Solid Earth tides
- Ocean tides

The ordinary differential equation solver used in HPOP is the Radau IIa, which is derived by Hairer and Wanner [31]. Radau IIa is derived from implicit Runge-Kutta methods that offer step size control and continuous output.

The programs "get_obs.m" and "get_data.m" are called to propagate the state vector **x** to times determined by step size and the number of observation sets. The "truth" state vector is propagated in "get_obs.m", where azimuth, right ascension, and/or range rate data are calculated. White noise is added to this data to simulate data that may be picked up by ground site hardware. Similarly, "get_data.m", used throughout the LMF, propagates the "estimated" orbit and records data at the times used for observations. As the LMF searches for state vectors with a better fit, "get_data.m" is used to calculate data in the generated orbits.

4.3.3 Least-Squares Filter

The following algorithm borrows from algorithms found in [24, 26, 27, 33, 35].

Step 1) Enter parameters, observations, options, and constants
Enter Y_0, vector of initial parameter estimates
Enter nxm matrix of observations **Obs** where column 1 is time
Enter options, discussed in Section 4.3.1
Let $\delta x = 25 * 10^{-7}$, perturbation value for the Jacobian
Let $\lambda_0 = 1 * 10^{-4}$, starting value of the Marquardt parameter
Let incr = 10, factor to increase λ
Let decr = 0.4, factor to increase λ
Let maxIter = 30, maximum iteration value
Let $\epsilon_1 = 1 * 10^{-4}$, gradient convergence criteria
Let $\epsilon_2 = 1 * 10^{-10}$, parameter convergence criteria
Let $\epsilon_3 = 1 * 10^{-6}$, root mean square convergence criteria
Let $\epsilon_1 = 1 * 10^{-12}$, parameter correction acceptance criteria
Let iterat = 1, to count iterations
Let n = length(**Obs**)
Let lx = length($\boldsymbol{Y}_0$)
Let $\boldsymbol{Obs}_{\boldsymbol{vec}}$ be a vector of the data found in matrix **Obs**
Let $\boldsymbol{data}_{vec}$ be a vector of data calculated from initial parameter estimates
Step 2) Calculate SSx = $S(\boldsymbol{Y}_0) = \boldsymbol{f}^T(\boldsymbol{Y}_0)\boldsymbol{f}(\boldsymbol{Y}_0)$, sum of squares of from initial estimate
If SSx cannot compute, stop.
Calculate **g** and test for gradient convergence
Let SSx[k] = SSx [k+1]
Step 3) Calculate $\boldsymbol{J}^T\boldsymbol{J}$ and $\boldsymbol{J}^T\boldsymbol{f}$
Let iterat = iterat + 1
For i = 1 to l
For k = 1 to lx
Let dx = bdx
Let **xd** = $\boldsymbol{Y}_0$
Let xd[k] = xd[k] +dx
Calculate $\boldsymbol{J}_{dat}$, data from perturbed parameters
Let **J**j[1:w-1,k] = $(\boldsymbol{J}_{dat} - \boldsymbol{data}_{vec})/dx$
End loop on k
Collect **J**j calculations into matrix **J**
End loop on i
Calculate **A** = $\boldsymbol{J}^T\boldsymbol{J}$
Calculate $\boldsymbol{g} = \boldsymbol{J}^T\boldsymbol{f}$, gradient
Let $\boldsymbol{D} = diag\big(diag(\boldsymbol{A})\big)$
Step 4) Solution of Eqn. (3.14)
Calculate $\boldsymbol{q} = (\boldsymbol{A} + \boldsymbol{\lambda} * \boldsymbol{D})^{-1}\,\boldsymbol{g}$
Step 5) Test parameter correction
Let $\boldsymbol{Y}_{try} = \boldsymbol{Y}_0 + \boldsymbol{q}$
Calculate data $\boldsymbol{y}_{try}$ using $\boldsymbol{Y}_{try}$

Let **f** = $\boldsymbol{Obs_{vec}} - \boldsymbol{y}_{try}$
Calculate $SSx_{try} = \boldsymbol{f}'\boldsymbol{f}$
Calculate $\beta = (SSx - SSx_{try})/(\boldsymbol{q}^T(\lambda\boldsymbol{q} + \boldsymbol{g})$, acceptance criteria
If $\beta > \epsilon_4$
Let dSSx = $SSx - SSx_{try}$
Let SSx[k] = SSx[k+1]
Let $\boldsymbol{Y}_0[k] = \boldsymbol{Y}_0[k+1]$
Let $\boldsymbol{data}_{vec}[k] = \boldsymbol{data}_{vec}[k+1]$
Let $\boldsymbol{Y}_0 = \boldsymbol{Y}_{try}$
Let $\lambda = \lambda * decr$
Else
SSx[k+1] = SSx[k]
$\lambda = \lambda * incr$

Step 6) Test for convergence
If max $|\boldsymbol{g}|| \leq \epsilon_1$ & iterat > 2
Stop
If max$\left|\left|\boldsymbol{g}\right|\right|$ / ($\left|\left|\boldsymbol{Y_0}\right|\right|$ + $\boldsymbol{1 * 10^{-6}}$) & iterat > 2
Stop
If iterat = maxIter
Stop

Step 7) Return to step 3, and try again to reduce the sum of squares

4.4 REMARKS

The LMF is said to be one of the best methods for fitting a state vector to noisy data. In this chapter, the LMF used in the present study is explained to give the reader an understanding of the necessary steps for using this algorithm. Methods used to simulate data are discussed and a detailed explanation of the software used is given. In the next chapter, the LMF will be tested using the satellites discussed in Section 4.3.1. By simulating the Doppler shift using a calculated range rate (with added noise), it is determined that if the LMF can converge to a solution for each of these satellites, the Marquardt algorithm for least-squares is a premier choice for fitting using Doppler data.

5. RESULTS AND ANALYSIS

In this chapter, the Levenberg-Marquardt Damped Least-Squares algorithm, outlined in previous chapters, is tested for its ability to provide the "best estimate" for the state vector of an orbit at a given epoch. To generate the "truth" orbit of a satellite, the Two-Line Element set of the desired satellite is downloaded into a text file, then it is processed using an SGP4 propagator and the cartesian "truth" state vector is produced. With the state vector acquired, an ephemeris is generated, and simulated observations of range-rate, azimuth, and elevation can be calculated. In the various cases run in this chapter, different combinations of these data are studied. In testing of real data, range-rate would be calculated using the Doppler shift of the carrier signal from the satellite. For the simulated case, it is assumed that the Doppler data has already been processed, giving range rate at each respective point in the orbit. A simulated ground site is used as an observation point. The simulated ground site shares the latitude and longitude of San Jose State University: $37.3352^o\ N, 121.8811^o\ W$. Next, the state vector of the "estimated" orbit is generated.

To generate simulated observations, an initial "estimated" state vector is created by perturbing the initial "truth" state vector. This "estimated" state vector is propagated, and the desired parameters are calculated using methods described in the previous chapter. To this data, white, zero-mean, Gaussian noise is added. The standard deviation for the noise added to each parameter can be seen in Table 4.1. The standard deviation for azimuth and elevation are "realistic" for satellite tracking radar sensors [29]. By both perturbing the predicted orbit and adding noise to the simulated data the response of the filter to an erroneous initial state vector can be evaluated.

Table 5.1 Gaussian Measurement Noise Standard Deviation

Parameter Type	Standard Deviation
Azimuth, Elevation	18 arc-seconds
Range-Rate	5 meters per second

The following outputs are collected from the filter upon execution:

- Summary table
    - Predicted state vector
    - A priori (estimated) state vector
    - Correction vector (quantifies correction made by filter to each state)
    - Best estimate for state vector at epoch

To determine the accuracy of the filter for each case, the correction vector is the most vital piece of information. To quantify the precision of the filter one needs only compare this Correction Vector (**CV**), to the Perturbing Vector (**PV**) that was added to the "truth" initial state vector. For example, should the filter produce a perfect fit of the data, the **CV** would be equal to the negative of the **PV**, or

$$\boldsymbol{CV} = -\boldsymbol{PV}. \tag{5.1}$$

Realistically, there will still be error in the "estimated" orbit.

The satellites tested in this study are the ISS, NAVSTAR-77, and MOLNIYA 3-50. These satellites were chosen for the variety seen between their respective orbits. For the ISS, a low altitude (about 416 km), circular orbit with small eccentricity is observed. Due the smaller size of the ISS orbit, the space station has an orbital period of about 1.5 hours. The NAVSTAR orbit is similar to that of the ISS, in that it is circular with small eccentricity; however, this orbit is much larger at around 20,189

km. The larger orbit gives the NAVSTAR satellite an orbital period of about 12 hours. The largest variation is introduced in the Molinaya orbit. The MOLNIYA 3-50 boasts a very high eccentricity of 0.72129. Like the NAVSTAR orbit, the MOLNIYA has an orbital period of about 12 hours. Tables 5.1 and 5.2 show the orbital elements of the three satellites used in this study. These state vectors are presented in both Cartesian and Keplerian elements. The Cartesian elements are meant to give the reader an idea of what the "truth" vector should look like, and the Keplerian elements are shown to give an idea of the size and shape of the respective orbits.

For the LMF, each orbit is propagated from the cartesian state vector at an epoch corresponding to the time of an overhead pass of the satellite. The epoch of the ISS was May 2, 2019 at 12:08:04 UTC. The initial state vector used for the NAVSTAR-77 had the epoch May 5, 2019 at 11:55:32 UTC. The epoch of the Molinaya orbit was May 2, 2019 at 12:08:04 UTC. Although it was attempted to get these passes close to each other, the significance of the ISS and MOLNIYA epochs are purely coincidence.

Table 5.1 Cartesian state vectors of test case satellites.

	ISS	NAVSTAR-77	MOLNIYA 3-50
x (km)	5700.1	-24840.6	10138.65
y (km)	2899.8	-8865.8	-19796.5
z (km)	2269.0	3155.2	24154.7
$\dot{x}$ (km/s)	-0.5624	1.1046	1.202
$\dot{y}$ (km/s)	5.3769	-1.9818	0.9401
$\dot{z}$ (km/s)	-5.4370	3.1397	-2.634

Table 5.2 Keplerian state vectors of test case satellites.

	ISS	NAVSTAR-77	MOLNIYA 3-50
h $\left(\frac{km^2}{s}\right)$	52029.77	102898.1	71305.23
e	0.001259	.000378	0.7208
Ω (deg)	223.3	194.8	152.2
i (deg)	51.65	55.0	62.1
ω (deg)	106.0	276.2	271.7
θ (deg)	48.72	92.17	212.0
a (km)	6791.5	26563.0	26549.2

5.1 TEST CASE RESULTS

For this investigation, various test cases were included. First, it was desired to test how accurately the LMF corrected satellite orbits of varying sizes and shapes. Many sources focus on testing filters on satellites following a circular orbit, so a highly elliptic orbit will be included in testing to discover if this factor changes the accuracy of the filter. Second, while the goal of this experiment is to test the ability of the filter to fit corrupted data from the downlink frequency shift (which is used to derive range-rate), test cases involving additional parameters are included for comparison.

The satellites tested in this paper were chosen for the size and shape of their respective orbits. Starting with the orbit with the lowest altitude in this experiment, the International Space Station (ISS) is tested. Next, increasing the altitude to MEO, the orbit of NAVSTAR-77, part of the Global Positioning System (GPS) family of satellites, is examined.

Finally, to test the filter on a highly eccentric orbit, the MOLNIYA 3-50 is included. As the focus of this paper is orbit estimation using the Doppler shift from the downlink of a satellite, satellites located in GEO were not considered for this experiment as there would be no apparent change in the downlink frequency of the respective satellite.

The “observation” parameters were chosen due to their observability. According to Folcik [29], typical observations used for satellite orbit estimation are made up of angular optical observations and radar observations. The angular optical observations consist of right ascension and declination measurements made against the background of stars. While radar observations also include two angle measurements, azimuth and elevation, they also include range and range-rate measurements. The present analysis assumes that for each case, direct determination of range-rate is not feasible. The number of observations used for each satellite varies based on the size of the orbit.

While the smaller orbit of the ISS requires fewer data points (25 were used in the present analysis) to allow the LMF to converge, Hunter [17] advises using an increased number of observations for the larger orbits of NAVSTAR-77 and MOLNIYA 3-50 to better capture their curvature. An increased step size allows the full pass of the satellite to be captured without creating abundant data points to calculate. For the both the NAVASTAR and MOLNIYA satellites, 100 observations were simulated. The output from the LMF is indicative of how well the filter performed. While optimizing a program could entail limiting the amount of function calls, it is expected that running the LMF with increased step size and observation counts will take longer to converge than the more circular ISS case. For the ISS, a step size of 25 seconds was chosen. With the 25 sets of observations, this step size allowed the filter access to the full pass duration of 10.5 minutes. To cover the full pass durations, the NAVSTAR and MOLNIYA orbits were given step sizes of 252 seconds and 360 seconds, respectively.

The output of the LMF is a set of four vectors. These vectors are

- Truth - the initial “truth” state vector retrieved from TLE of the satellite
- Estimated - “estimated” initial state vector. In the case of this paper, this state vector was “estimated” by perturbing the “truth” vector. Each satellite initial state vector is perturbed by

a unique value vector. For smaller orbits, i.e., the orbit of the ISS, a closer estimate is assumed. Conversely, the orbits of the NAVSTAR and MOLNIYA satellites will have a higher assumed error in the initial state estimate.

- Correction - a vector of overall corrections made by the LMF to the "estimated" state vector.
- Final - a vector showing the corrected "estimated" state vector.

Table 5.3 shows the vectors used to perturb each of the satellites under investigation.

It is worth noting that when using the SGP4 propagator on the TLE for the ISS, the resultant state vector is the negative of the true state vector. To remedy this, signs of the state vector were changed each time the ISS was tested. To understand how well the LMF performs, the run time of each case will also be recorded.

Table 5.3 State Vector Perturbation Vectors.

	ISS	NAVSTAR	MOLNIYA
x (km)	0.5	-5.0	6.0
y (km)	-2.0	-3.0	-2.0
z (km)	1.0	4.0	5.0
$\dot{x}$ (km/s)	$-1.3* 10^{-3}$	$-1.6* 10^{-3}$	$3.0* 10^{-3}$
$\dot{y}$ (km/s)	$1.0* 10^{-3}$	$1.0* 10^{-3}$	$1.0* 10^{-3}$
$\dot{z}$ (km/s)	$-0.5* 10^{-3}$	$-2.5* 10^{-3}$	$-2.5* 10^{-3}$

5.1.1 Range rate, Azimuth, and Elevation Cases

While the focus of the present paper is to determine the adequacy of the LMF to fit noisy $\dot{\rho}$ data, it is important to have cases with various parameters to compare results. In this subsection, the results from using the most data types – namely, range rate, azimuth, and elevation – are displayed. In all

cases, range rate data represents transformed Doppler data. If real data were present, $\dot{\rho}$ would be calculated from the Doppler shift in the downlink signal of the satellite using Eqn. (1.23). Tables (5.4-5.6) show the results of using range rate, azimuth, and elevation. The final row of each table gives the time t for the LMF to converge for the respective case.

Table 5.4 ISS Results from $\dot{\rho}$, Azimuth, Elevation.

	Truth	Predicted	Correction	Final
x (km)	-6197.6	-6197.1	-0.5	-6197.6
y (km)	-1366.2	-1368.2	2.0	-1366.2
z (km)	2416.3	2417.3	-1.0	2416.3
$\dot{x}$ (km/s)	3.0828	3 .0815	$1.33* 10^{-3}$	3.0828
$\dot{y}$ (km/s)	-4.5287	-4.5277	$-1.0* 10^{-3}$	-4.5287
$\dot{z}$ (km/s)	5.3561	5.3556	$0.5* 10^{-3}$	5.361
t = 446.626 s				

Table 5.5 NAVSTAR-77 Results from $\dot{\rho}$, Azimuth, Elevation.

	Truth	Predicted	Correction	Final
x (km)	-14280.1	-14285.0	5.0	-14280.1
y (km)	-15437.3	-15440.3	3.0	-14537.3
z (km)	16212.8	16216.8	-4.0	16212.8
$\dot{x}$ (km/s)	3.1686	3.167	$1.6* 10^{-3}$	3.1687
$\dot{y}$ (km/s)	-0.7177	-0.7067	$-1.0* 10^{-3}$	-0.7077
$\dot{z}$ (km/s)	2.1153	2.1128	$2.5* 10^{-3}$	2.1153
t = 2639.753 s				

Table 5.6 MOLNIYA 3-50 Results from $\dot{\rho}$, Azimuth, Elevation.

	Truth	Predicted	Correction	Final
x (km)	-14280.1	-14274.1	-6.0	-6197.6
y (km)	-15437.3	-15439.3	2.0	-15439.4
z (km)	16212.8	16217.8	-5.0	16212.8
$\dot{x}$ (km/s)	3.1686	3.1716	$-3.01* 10^{-3}$	3.1686
$\dot{y}$ (km/s)	-0.7077	-0.7067	$-1.01* 10^{-3}$	-0.7078
$\dot{z}$ (km/s)	2.1153	2.1128	$2.49* 10^{-3}$	2.1153
t = 3539.89 s				

As seen in Tables (5.4-5.6), the LMF performed remarkably well. Each of the estimated state vectors were correctly fitted to the noisy data. The data collected from the orbit of the propagated "estimated" state vector, as well as noisy data collected from the "truth" orbit can be found in Appendix A.

5.1.2 Angles Only Case

For the second set of cases, angles of azimuth and elevation (with added noise) will serve as the "observed" data. This case is included in the present analysis to determine if including range rate information has a significant impact on the performance of the LMF. Tables (5.7-5.9) show the results of using angles only with the LMF.

Table 5.7 ISS Results from Azimuth & Elevation.

	Truth	Predicted	Correction	Final
x (km)	-6197.3	-6197.8	-0.5	-6197.3
y (km)	-1366.8	-1368.8	2.0	-1366.8
z (km)	2416.8	2417.8	-1.0	2416.8
$\dot{x}$ (km/s)	3.0836	3 .0823	$1.29* 10^{-3}$	3.0836
$\dot{y}$ (km/s)	-4.5285	-4.5275	$-0.99* 10^{-3}$	-4.5285
$\dot{z}$ (km/s)	5.3558	5.3553	$0.49* 10^{-3}$	5.3557
t = 90.013				

Table 5.8 NAVSTAR-77 Results from Azimuth & Elevation.

	Truth	Predicted	Correction	Final
x (km)	14277.2	-14272.2	5.0	14277.2
y (km)	15437.6	15440.6	3.0	15437.6
z (km)	-16214.8	-16210.8	-4.0	-16214.8
$\dot{x}$ (km/s)	-3.1690	-3.706	$1.59* 10^{-3}$	-3.1690
$\dot{y}$ (km/s)	0.7175	0.7085	$-0.9901* 10^{-3}$	0.7075
$\dot{z}$ (km/s)	-2.1149	-2.1174	$2.49* 10^{-3}$	-2.1149
t = 5835.172 s				

Table 5.9 MOLNIYA 3-50 Results from Azimuth & Elevation.

	Truth	Predicted	Correction	Final
x (km)	-19347.0	-19341.0	-6.0	-19347.0
y (km)	1824.3	1822.3	2.0	1824.3
z (km)	14893.2	14898.2	-5.0	14893.2
$\dot{x}$ (km/s)	-1.1151	-1.1121	$-2.99 * 10^{-3}$	-1.1150
$\dot{y}$ (km/s)	-1.6175	-1.6165	$-0.99 * 10^{-3}$	-1.6175
$\dot{z}$ (km/s)	3.6997	3.6972	$2.49 * 10^{-3}$	3.6997
t = 2805.202 s				

5.1.1 Range Rate Only Case

For the final set of cases, range rate $\dot{\rho}$ data will be the only observations used for the LMF. The results of this case are seen in Tables (5.10-5.12).

Table 5.4 ISS Results from $\dot{\rho}$ only.

	Truth	Predicted	Correction	Final
x (km)	-6197.3	-6196.8	-0.5	-6197.3
y (km)	-1366.8	-1368.8	2.0	-1366.8
z (km)	2416.3	2417.3	-1.0	2416.8
$\dot{x}$ (km/s)	3.0836	3 .0815	$1.29 * 10^{-3}$	3.0828
$\dot{y}$ (km/s)	-4.5285	-4.5277	$-1.01 * 10^{-3}$	-4.5287
$\dot{z}$ (km/s)	5.3558	5.3553	$0.49 * 10^{-3}$	5.3557
t = 365.792 s				

For the NAVSTAR-77, using the original perturbation mentioned Table 5.3 consistently caused the orbit LMF to diverge. When the perturbation vector was decreased to

$$\mathbf{PV} = [-4 \quad -2.5 \quad 3 \quad -1.6* 10^{-3} \quad 1* 10^{-3} \quad -2* 10^{-3}],$$

the LMF was able to converge.

Table 5.5 NAVSTAR-77 Results from $\dot{\rho}$ only (using updated PV).

	Truth	Predicted	Correction	Final
x (km)	14277.2	-14273.2	4.0	-14277.2
y (km)	15437.6	-154351	2.5	-14437.3
z (km)	-16214.8	16216.8	-3.0	16214.8
$\dot{x}$ (km/s)	-3.1686	3.167	$1.61* 10^{-3}$	3.1690
$\dot{y}$ (km/s)	0.7177	-0.7067	$-0.99* 10^{-3}$	-0.7075
$\dot{z}$ (km/s)	-2.1153	2.1128	$2.01* 10^{-3}$	2.1149
t = 4416.133 s				

Table 5.9 MOLNIYA 3-50 results from $\dot{\rho}$ only.

	Truth	Predicted	Correction	Final
x (km)	-19341.5	-19341.5	-6.0	-19341.5
y (km)	1826.6	-15439.3	2.0	-1826.6
z (km)	14880.8	16217.8	-5.0	14880.8
$\dot{x}$ (km/s)	-1.1162	3.1716	$-2.99* 10^{-3}$	-1.1162
$\dot{y}$ (km/s)	-1.6178	-0.7067	$-0.99* 10^{-3}$	-1.6178
$\dot{z}$ (km/s)	3.7013	2.1128	$2.51* 10^{-3}$	3.7013
t = 2514.526 s				

5.2 ANALYSIS

The results have shown that the LMF is remarkable at fitting an orbit to noisy data, including cases where the only observation is range rate. Most test cases converged using the perturbing vectors outlined in Table 5.3 with the outlier being angles only case for the NAVSTAR satellite.

Using the original PV caused the LMF to apply zero corrections to the estimated orbit for

NAVSTAR-77. Through the iterations, it was observed that each new change was rejected, and λ continued to grow until it hit a max at 10^{-7}. Potential changes were repeatedly rejected even as the potential corrections dwindled down to zero. It was believed that the initial guess may have been too far off for this case (as the literature suggests the filter relies on an estimate close to the solution), so the **PV** was decreased in an attempt to remedy the failure. Indeed, the decrease in **PV** allowed the filter to find the solution. Previous builds of the LMF ran into problems as well.

The results found in this section are the outcome of a fourth generation build of the LMF. Previous iterations proved successful (see Appendix C for results) for the ISS, which converged, and NAVSTAR, which gained successful, albeit lacking, corrections, but it failed to properly correct the MOLNIYA cases. Additionally, even though the ISS and NAVSTAR cases were able to be corrected, there was still room for improvement to the solution, but additional observations only served to exacerbate the error in the final estimate. This failure was prominent in the MOLNIYA case as it was believed that adding data from an additional pass would fix the problem, but the added pass only increased computational cost with zero benefit to the solution. The latest build (used for the results in this chapter) allowed convergence of each of the satellite cases by making several modifications to the program structure.

In previous LMF software designs created in this study, the least-squares algorithm and orbit generator were grouped into one program. While this design can work, as it did for the ISS and NAVSTAR cases, it became difficult to track variable usage. For example, one section of the program split a time of the format HH:MM:SS, where HH, MM, and SS are hours, minutes, and seconds, respectively. The hours, minutes, and seconds were saved to a time vector with variables {hr min sec}. When the output from the program was far from what was expected, it took many hours before the mistake was found. In this case, saving a number under the variable “min” caused

equations that were meant to find the minimum (using the min.m function) to output erroneous solutions. The use of "sec" as a variable also caused errors when calling external functions as MatLab uses "sec" to find the secant of an angle. Though MatLab typically spots these types of errors and notifies the user, in this case the only hint of the error was in the output of the program. This was one of many instances that urged the reformatting of the software. Thus, the LMF and orbit/data generator were built into separate functions, leaving the original file to serve as a testing function. Once this procedure was complete, each case was able to converge.

As was expected, the ISS was able to converge much faster than the other two cases. This relatively quick convergence is the product of fewer necessary observations and smaller step sizes. Using this ideology, it was surprising to find the MOLNIYA cases converging about forty minutes faster (averaged) than the NAVSTAR in the range rate only and angles only cases. This may be due to the eccentricity of the Molinaya orbit. Future work will investigate this matter further.

To understand the full capabilities of the LMF used in the paper, future work should involve the testing of real satellite data. While adding noise to simulated data gives a feel for what could be seen at a ground site, real observations would challenge the algorithm and discovering the trajectory of a real satellite with a least-squares algorithm would be all the more rewarding.

6. CONCLUSION AND FUTURE WORK

The goal of this investigation included applying the Levenberg-Marquardt least-squares Filter (LMF) on noisy Doppler shift data to determine the orbit of a given satellite. The focus was to determine if noisy Doppler data could be used to produce an accurate – or better put, a "best estimate" – for the state vector of the satellite at epoch. Previous studies using range rate information for the orbit determination problem, e.g. work done by Nick Komaroff (personal conversation, December 22, 1993), relied on self-manufacture hardware to capture the Doppler shift from the downlink signal of a satellite.

Komaroff (personal conversation, 1993), designed a circuit around a crystal discriminator, which was integrated into ground station hardware. This hardware allowed an output of the received signal frequency over time, which was converted to range rate data using methods described in in this report. While the range rate data in this report is simulated, the present study is still designed to follow the goal of Komaroff (personal conversation, 1993). While real data was not present in the present investigation, it is worth noting that current software can give this frequency over time without the need for the additional hardware.

For this project, the LMF was chosen for its reputation for accurately predicting a state vector from noisy observations. An orbit propagator was used to generate state vectors over the duration of the pass time of a satellite, and data was calculated from each state vector. To test the proficiency of the filter, observations were simulated from LEO, MEO, and Molinaya orbit test cases. For each case, the filter corrected the initial "estimated" state to the supplied noisy data. Three combinations of data were used in this process.

To compare the results of the focus case, range rate, additional cases with various data types were

included in the analysis. The first additional case had data comprised of range rate, azimuth, and elevation. Because this data set had the most data type, poor resolution in the range rate only case could be quantified. The second added case used only azimuth and elevation angles. Should this case have outperformed the previous case and the range rate only case, an understanding of how the range rate impacted the results could be acquired.

Throughout the building of the LMF software, many iterations of the filter were attempted and failed. Initial builds allowed convergence on the orbit of the ISS and at least some correction for NAVSTAR, but it diverged consistently with the Molinaya orbit. When the addition of data from a subsequent pass only made state vector estimates worse, the program was overhauled and netted positive results. In addition to reformatting the software, several additions were made to improve performance of the filter.

While the small step sizes used for the ISS allowed a quick convergence, the orbits of NAVSTAR and MOLNIYA require larger steps to cover their arch. This large step size causes increased computational cost when calling the Orbit Generating Function (OGF). For each iteration, the OGF is called three times, once to calculate data for the current estimated orbit, once for the Jacobian, and again to test the estimated state vector with added corrections. The Broyden rank-1 Jacobian, suggested by Transtrum and Sentha [27] decreases the amount of OGF calls by estimating the Jacobian over several iterations using previous Jacobian elements. Testing this addition proved that similar data was provided using the rank-1 approximation, so it was implemented into the software. The MatLab code for the newest build can be found in Appendix D.

It is determined that the Levenberg-Marquardt filter is quite capable of providing a state estimate for a satellite using topocentric range rate. If range rate is calculated from the Doppler data of a given satellite, this filter will give its user enough information to find the trajectory of the satellite.

6.1 FUTURE WORK

The successful outcome of this filter begs that it be used with real data. Future work should test the performance of the filter when given processed Doppler data. Initial testing should include data from satellites with known trajectories. If the filter proves successful, more difficult targets could be studied. Should it be desired, several modifications could still be made to the software.

While the filter has seen successful testing with simulated data, the Orbit Propagator has not been tested for its accuracy. Indeed, the OP simulated orbital perturbations, i.e., J2, atmosphere, solar pressure, etc., but results have not been compared to higher end software. Additionally, data extracted from the orbit propagator used in "get_obs.m" did not match data provided by GP. Future work should conduct these tests to ensure any accrued error found when using real data does not stem from this OP.

Future work could also include modifications to allow for two-way Doppler processing. An added function to perform this processing could allow the filter to estimate a state vector for a non-communicating satellite.

References

1. Deng, L., Sun, X., and Han, C.,Analysis and Comparison on UKF and BLS for Orbit Determination,Proc., *AAS/AIAA 2015 Astrodynamics Specialists Conf.*, American Astronautical Society (AAS), Vail, 9-13 Aug. 2015.
2. Bordi, J.J,. Analysis of the Precise Range and Range-Rate Equipment (PRARE) and Application to Precise Orbit Determination, May 1999.
3. Amiri, S., and Mehdipour, M., Accurate Doppler Frequency Shift Estimation for any Satellite Orbit, *RAST 2007 3rd International Conference on Recent Advances in Space Technologies*, Recent Advances in Space Technologies (RAST), Istanbul, Turkey, 14-16 June 2007.
4. Stolarski, M., and Woźniak, G., Estimation of PW-Sat Satellite Orbit Based on Doppler Effect, *Proc. SPIE 8454, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2012, 84540H*, 7 November 2012.
   doi: 10.1117/12.2000192
5. Tabakovic, Z., Doppler Effect in Non-GSO Satellite Propagation. *IEEE/AP 2000 Millennium Conference on Antennas and Propagation*, Davos, Switzerland, 9-14 May 2000.
6. Agostino, M. D., Manzino, A., and Marucco, G., Doppler Measurement Integration for Kinematic Real-Time GPS Positioning, *Applied Geomatics,2*(4), Dec. 2010, pp. 155-162.
   doi:10.1007/s12518-010-0031-z
7. Ialongo, G., Method of Doppler Data Processing For Orbit Determination, TR-0066(5110-01)-5, Oct. 1969.
   doi:10.21236/ad0704587
8. Shuch, H., Demonstrating Celestial Mechanics through Measured Doppler Shift [online], Apr. 1992, http://www.setileague.org/articles/ham/kepler.pdf.

9. Kirschner, S., Samii, M., Broaddus, S., and Doll, C., Preliminary Orbit Determination System for Tracking and Data Relay Satellite System-Tracked Target Spacecraft using the Homotopy Continuation Method, *In its Flight Mechanics/Estimation Theory Symposium*, May 10-11 1988, pp. 217-237.
10. Izsak, I. G., Orbit Determination from Simultaneous Doppler Shift Measurements, *SAO Special Report #38*, Dec. 1959.
11. Estefan, J., Precise Orbit Determination of High-Earth Elliptical Orbiters using Differenced Doppler and Ranging Measurements, *IEEE PLANS 92 Position Location and Navigation Symposium Record*, Vol. 7, No. 5, May 1992, pp. 12-18. doi:10.1109/plans.1992.185827
12. Guier, W. H., and Weiffenbach, G. C., The Doppler Determination of Orbits, AD-409 103, Jul. 1963.
13. Zhang, J., Zhang, K., Grenfell, R., and Deakin, R., Short Note: On the Relativistic Doppler Effect for Precise Velocity Determination using GPS. *Journal of Geodesy,* Vol. 80, No. 2, May 2006, pp. 104-110. doi:10.1007/s00190-006-0038-8
14. Curtis, H. D., Orbital Mechanics for Engineering Students, 3rd ed., Elsevier Aerospace Engineering Series, 2014.
15. Coyne, G. V., Hoskin, M. A., and Pedersen, O., Gregorian Reform of the Calendar, *Proc. Vatican Conference to Commemorate its 400th Anniversary, 1582-1982*, Specola Vaticana, Vatican City, 1982, pp. xxv-323.
16. Scharringhausen, B., How Was the Starting Point for the Julian Date System Chosen? (Advanced), Ask an Astronomer, URL: <http://curious.astro.cornell.edu/about-us/125-observational-astronomy/timekeeping/calendars/763-how-was-the-starting-point-for-the-julian-

date-system-chosen-advanced>, June 2018.

17. Hunter, J., *Orbital Mechanics Course Reader,* 2018.
18. Michele, R., Position in an Elliptical Orbit, Aerospace Engineering, URL: http://www.aerospacengineering.net/?p=537, Oct. 2013.
19. Sorenson, Harold W., Least-Squares Estimation: from Gauss to Kalman. *IEEE Spectrum,* Vol. 7, No. 7, 1970, pp. 63-68.
20. Vallado, D. A., and McClain, W. D., *Fundamentals of astrodynamics and applications,* 3rd ed., Microcosm Press, Hawthorne, 2007.
21. Ippolito L.J.. "Radio Noise in Satellite Communications", Radiowave Propagation in Satellite Communications, Springer, Jan. 1986, pp. 122-138. doi: 10.1007/978-94-011-7027-7
22. Kuga, H., and Orlando, V., Assessing Orbit Determination Through One Way Doppler Signals, *ISSFD 2003 International Symposium on Spacecraft Flight Dynamics*, Moscow, Jan. 2003.
23. Gavin, H. P., The Levenberg-Marquardt Method for Nonlinear Least Squares Curve- Fitting Problems, 2017.
24. Nash, J. C., Compact Numerical Methods for Computers: Linear Algebra and Function Minimization, 3rd ed., CRC PRESS, New York, 1990.
25. Higham, N. J., Cholesky Factorization, *Wiley Interdisciplinary Reviews: Computational Statistics,*Vol. 1, No. 2, 2009, pp. 251-254. doi:10.1002/wics.18
26. Press, W. H., and Vetterling, W. T., Numerical Recipes, 3rd ed., Cambridge Univ. Press. Cambridge, 1999.
27. Transtrum, M. K., & Sentha, J., Improvements to the Levenberg-Marquardt Algorithm for Nonlinear Least-Squares Minimization, 2012.

28. Mahooti, M., *SGP4*, MathWorks, URL: https://www.mathworks.com/matlabcentral/fileexchange/62013-sgp, March 2017.
29. Folcik, Z. J., Orbit Determination Using Modern Filters/smoothers and Continuous Thrust Modeling (Unpublished Master's Thesis), Massachusetts Institute of Technology, URL: <https://dspace.mit.edu/bitstream/handle/1721.1/44936/312478369-MIT.pdf?sequence=2>, 2008.
30. Battin, R., An Introduction to the Mathematics and Methods of Astrodynamics, revised ed., AIAA Education Series, 1999.
31. Hairer, E., and Wanner, G., Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems, Springer Berlin, Berlin, 2010.
32. Levenberg, K., A Method for the Solution of Certain Non-Linear Problems in Least Squares, *The Quarterly of Applied Mathematics*, 1944.
33. Marquardt, D.W., An Algorithm for Least-Squares Estimation of Nonlinear Parameters, *Journal of the Society for Industrial and Applied Mathematics*, 1963.
34. Moré, J., The Levenberg-Marquardt Algorithm: Implementation and Theory, *Lecture Notes in Mathematics 630*, 1977.
35. Nielson, H.B., Damping Parameter in Marquardt's Method, IMM-REP-1999-05, 1999.
36. Broyden, C. G., A Class of Methods for Solving Nonlinear Simultaneous Equations, *Mathematics of Computation*, Vol. 19, No. 92, 1965, p. 577. doi:10.2307/2003941

APPENDIX A. SIMULATED DATA

ISS MEASUREMENTS FROM TRUTH ORBIT

Least-squares orbit determination
Measurements from “truth” orbit

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/12	05:36:44.000	181.450	5.710	-5.404
2019/05/12	05:37: 4.000	178.368	7.068	-5.158
2019/05/12	05:37:24.000	174.863	8.469	-4.859
2019/05/12	05:37:44.000	170.873	9.898	-4.498
2019/05/12	05:38: 4.000	166.333	11.329	-4.065
2019/05/12	05:38:24.000	161.190	12.722	-3.548
2019/05/12	05:38:44.000	155.412	14.017	-2.941
2019/05/12	05:39: 4.000	149.009	15.139	-2.243
2019/05/12	05:39:24.000	142.058	15.997	-1.464
2019/05/12	05:39:44.000	134.706	16.512	-0.625
2019/05/12	05:40: 4.000	127.172	16.625	0.241
2019/05/12	05:40:24.000	119.706	16.326	1.097
2019/05/12	05:40:44.000	112.549	15.649	1.907
2019/05/12	05:41: 4.000	105.883	14.668	2.643
2019/05/12	05:41:24.000	99.817	13.470	3.291
2019/05/12	05:41:44.000	94.389	12.137	3.848
2019/05/12	05:42: 4.000	89.585	10.737	4.319
2019/05/12	05:42:24.000	85.359	9.320	4.711
2019/05/12	05:42:44.000	81.649	7.918	5.037
2019/05/12	05:43: 4.000	78.393	6.551	5.306
2019/05/12	05:43:24.000	75.528	5.228	5.529
2019/05/12	05:43:44.000	73.001	3.955	5.713
2019/05/12	05:44: 4.000	70.761	2.731	5.865
2019/05/12	05:44:24.000	68.770	1.554	5.991
2019/05/12	05:44:44.000	66.990	0.422	6.096

ISS MEASUREMENTS WITH ADDED NOISE
Least-squares orbit determination
Measurements with added noise

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/12	05:36:44.000	181.458	5.710	-5.395
2019/05/12	05:37: 4.000	178.369	7.073	-5.165
2019/05/12	05:37:24.000	174.860	8.475	-4.862
2019/05/12	05:37:44.000	170.876	9.896	-4.499
2019/05/12	05:38: 4.000	166.326	11.323	-4.060
2019/05/12	05:38:24.000	161.190	12.718	-3.551
2019/05/12	05:38:44.000	155.416	14.017	-2.938
2019/05/12	05:39: 4.000	149.011	15.142	-2.241
2019/05/12	05:39:24.000	142.051	15.999	-1.458
2019/05/12	05:39:44.000	134.710	16.500	-0.626
2019/05/12	05:40: 4.000	127.170	16.627	0.239
2019/05/12	05:40:24.000	119.706	16.334	1.101
2019/05/12	05:40:44.000	112.551	15.645	1.906
2019/05/12	05:41: 4.000	105.884	14.666	2.641
2019/05/12	05:41:24.000	99.819	13.478	3.279
2019/05/12	05:41:44.000	94.384	12.142	3.851
2019/05/12	05:42: 4.000	89.584	10.739	4.314
2019/05/12	05:42:24.000	85.353	9.322	4.719
2019/05/12	05:42:44.000	81.653	7.916	5.041
2019/05/12	05:43: 4.000	78.392	6.547	5.304
2019/05/12	05:43:24.000	75.525	5.227	5.531
2019/05/12	05:43:44.000	72.995	3.957	5.718
2019/05/12	05:44: 4.000	70.762	2.734	5.865
2019/05/12	05:44:24.000	68.780	1.559	5.982
2019/05/12	05:44:44.000	66.991	0.413	6.101

NAVSTAR-77 MEASUREMENTS FROM TRUTH ORBIT

Least-squares orbit determination
Measurements from “truth” orbit (no added noise)

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/12	18:16:22.000	169.059	-11.410	0.716
2019/05/12	18:20:34.000	168.895	-9.674	0.715
2019/05/12	18:24:46.000	168.702	-7.963	0.714
2019/05/12	18:28:58.000	168.480	-6.276	0.712
2019/05/12	18:33:10.000	168.227	-4.615	0.708
2019/05/12	18:37:22.000	167.944	-2.979	0.703
2019/05/12	18:41:34.000	167.628	-1.369	0.698
2019/05/12	18:45:46.000	167.280	0.214	0.691
2019/05/12	18:49:58.000	166.898	1.770	0.683
2019/05/12	18:54:10.000	166.482	3.298	0.674
2019/05/12	18:58:22.000	166.030	4.798	0.665
2019/05/12	19:02:34.000	165.543	6.269	0.654
2019/05/12	19:06:46.000	165.018	7.711	0.643
2019/05/12	19:10:58.000	164.455	9.122	0.630
2019/05/12	19:15:10.000	163.853	10.502	0.617
2019/05/12	19:19:22.000	163.212	11.850	0.603
2019/05/12	19:23:34.000	162.530	13.166	0.588
2019/05/12	19:27:46.000	161.806	14.448	0.573
2019/05/12	19:31:58.000	161.041	15.696	0.557
2019/05/12	19:36:10.000	160.232	16.908	0.540
2019/05/12	19:40:22.000	159.380	18.083	0.522
2019/05/12	19:44:34.000	158.484	19.222	0.504
2019/05/12	19:48:46.000	157.544	20.322	0.485
2019/05/12	19:52:58.000	156.559	21.382	0.466
2019/05/12	19:57:10.000	155.528	22.401	0.446
2019/05/12	20:01:22.000	154.453	23.379	0.425
2019/05/12	20:05:34.000	153.332	24.314	0.405
2019/05/12	20:09:46.000	152.166	25.204	0.383
2019/05/12	20:13:58.000	150.956	26.050	0.362
2019/05/12	20:18:10.000	149.702	26.849	0.340
2019/05/12	20:22:22.000	148.405	27.600	0.318
2019/05/12	20:26:34.000	147.065	28.303	0.295
2019/05/12	20:30:46.000	145.685	28.956	0.273
2019/05/12	20:34:58.000	144.265	29.558	0.250
2019/05/12	20:39:10.000	142.808	30.109	0.227
2019/05/12	20:43:22.000	141.315	30.607	0.203
2019/05/12	20:47:34.000	139.789	31.052	0.180
2019/05/12	20:51:46.000	138.231	31.444	0.157
2019/05/12	20:55:58.000	136.645	31.781	0.133
2019/05/12	21:00:10.000	135.033	32.064	0.110
2019/05/12	21:04:22.000	133.399	32.292	0.087

2019/05/12	21:08:34.000	131.744	32.465	0.064
2019/05/12	21:12:46.000	130.072	32.584	0.041
2019/05/12	21:16:58.000	128.387	32.648	0.018
2019/05/12	21:21:10.000	126.690	32.658	-0.004
2019/05/12	21:25:22.000	124.986	32.614	-0.027
2019/05/12	21:29:34.000	123.277	32.518	-0.049
2019/05/12	21:33:46.000	121.566	32.369	-0.070
2019/05/12	21:37:58.000	119.856	32.171	-0.091
2019/05/12	21:42:10.000	118.149	31.922	-0.112
2019/05/12	21:46:22.000	116.448	31.625	-0.132
2019/05/12	21:50:34.000	114.753	31.282	-0.152
2019/05/12	21:54:46.000	113.069	30.893	-0.172
2019/05/12	21:58:58.000	111.395	30.460	-0.190
2019/05/12	22:03:10.000	109.733	29.986	-0.208
2019/05/12	22:07:22.000	108.084	29.472	-0.226
2019/05/12	22:11:34.000	106.450	28.920	-0.243
2019/05/12	22:15:46.000	104.830	28.332	-0.259
2019/05/12	22:19:58.000	103.225	27.710	-0.274
2019/05/12	22:24:10.000	101.635	27.056	-0.288
2019/05/12	22:28:22.000	100.059	26.372	-0.302
2019/05/12	22:32:34.000	98.498	25.660	-0.315
2019/05/12	22:36:46.000	96.951	24.923	-0.327
2019/05/12	22:40:58.000	95.418	24.163	-0.338
2019/05/12	22:45:10.000	93.897	23.382	-0.347
2019/05/12	22:49:22.000	92.387	22.582	-0.356
2019/05/12	22:53:34.000	90.888	21.765	-0.364
2019/05/12	22:57:46.000	89.398	20.934	-0.371
2019/05/12	23:01:58.000	87.916	20.091	-0.377
2019/05/12	23:06:10.000	86.441	19.238	-0.382
2019/05/12	23:10:22.000	84.972	18.377	-0.385
2019/05/12	23:14:34.000	83.506	17.512	-0.388
2019/05/12	23:18:46.000	82.044	16.644	-0.389
2019/05/12	23:22:58.000	80.583	15.775	-0.389
2019/05/12	23:27:10.000	79.122	14.908	-0.388
2019/05/12	23:31:22.000	77.660	14.046	-0.386
2019/05/12	23:35:34.000	76.196	13.191	-0.382
2019/05/12	23:39:46.000	74.728	12.345	-0.377
2019/05/12	23:43:58.000	73.255	11.510	-0.371
2019/05/12	23:48:10.000	71.777	10.690	-0.364
2019/05/12	23:52:22.000	70.292	9.887	-0.356
2019/05/12	23:56:34.000	68.799	9.103	-0.346
2019/05/13	00:00:46.000	67.297	8.341	-0.335
2019/05/13	00:04:58.000	65.787	7.603	-0.324
2019/05/13	00:09:10.000	64.267	6.893	-0.310
2019/05/13	00:13:22.000	62.738	6.211	-0.296
2019/05/13	00:17:34.000	61.199	5.562	-0.281
2019/05/13	00:21:46.000	59.650	4.947	-0.265

2019/05/13	00:25:58.000	58.091	4.369	-0.247
2019/05/13	00:30:10.000	56.523	3.830	-0.229
2019/05/13	00:34:22.000	54.947	3.333	-0.209
2019/05/13	00:38:34.000	53.363	2.879	-0.189
2019/05/13	00:42:46.000	51.772	2.472	-0.168
2019/05/13	00:46:58.000	50.176	2.112	-0.146
2019/05/13	00:51:10.000	48.575	1.801	-0.124
2019/05/13	00:55:22.000	46.972	1.542	-0.101
2019/05/13	00:59:34.000	45.368	1.336	-0.077
2019/05/13	01:03:46.000	43.764	1.184	-0.053
2019/05/13	01:07:58.000	42.163	1.087	-0.028
2019/05/13	01:12:10.000	40.568	1.046	-0.003

NAVSTAR MEASUREMENTS WITH ADDED NOISE

Least-squares orbit determination
Measurements with added noise

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/12	18:16:22.000	169.060	-11.407	0.716
2019/05/12	18:20:34.000	168.896	-9.676	0.715
2019/05/12	18:24:46.000	168.700	-7.957	0.714
2019/05/12	18:28:58.000	168.482	-6.268	0.712
2019/05/12	18:33:10.000	168.231	-4.608	0.708
2019/05/12	18:37:22.000	167.937	-2.980	0.703
2019/05/12	18:41:34.000	167.633	-1.360	0.698
2019/05/12	18:45:46.000	167.271	0.214	0.691
2019/05/12	18:49:58.000	166.908	1.771	0.683
2019/05/12	18:54:10.000	166.479	3.291	0.674
2019/05/12	18:58:22.000	166.030	4.800	0.665
2019/05/12	19:02:34.000	165.544	6.269	0.654
2019/05/12	19:06:46.000	165.018	7.714	0.643
2019/05/12	19:10:58.000	164.460	9.114	0.630
2019/05/12	19:15:10.000	163.856	10.499	0.617
2019/05/12	19:19:22.000	163.215	11.847	0.603
2019/05/12	19:23:34.000	162.536	13.171	0.588
2019/05/12	19:27:46.000	161.802	14.445	0.573
2019/05/12	19:31:58.000	161.049	15.700	0.557
2019/05/12	19:36:10.000	160.223	16.910	0.540
2019/05/12	19:40:22.000	159.382	18.079	0.522
2019/05/12	19:44:34.000	158.478	19.223	0.504
2019/05/12	19:48:46.000	157.538	20.325	0.485
2019/05/12	19:52:58.000	156.558	21.373	0.466
2019/05/12	19:57:10.000	155.532	22.397	0.446
2019/05/12	20:01:22.000	154.450	23.389	0.425
2019/05/12	20:05:34.000	153.331	24.313	0.405
2019/05/12	20:09:46.000	152.171	25.211	0.383
2019/05/12	20:13:58.000	150.953	26.046	0.362
2019/05/12	20:18:10.000	149.698	26.851	0.340
2019/05/12	20:22:22.000	148.403	27.602	0.318
2019/05/12	20:26:34.000	147.071	28.308	0.295
2019/05/12	20:30:46.000	145.681	28.951	0.273
2019/05/12	20:34:58.000	144.269	29.560	0.250
2019/05/12	20:39:10.000	142.812	30.110	0.227
2019/05/12	20:43:22.000	141.307	30.603	0.203
2019/05/12	20:47:34.000	139.786	31.051	0.180
2019/05/12	20:51:46.000	138.232	31.444	0.157
2019/05/12	20:55:58.000	136.642	31.783	0.133
2019/05/12	21:00:10.000	135.034	32.063	0.110
2019/05/12	21:04:22.000	133.397	32.285	0.087
2019/05/12	21:08:34.000	131.749	32.472	0.064

2019/05/12	21:12:46.000	130.075	32.579	0.041
2019/05/12	21:16:58.000	128.389	32.642	0.018
2019/05/12	21:21:10.000	126.692	32.660	-0.004
2019/05/12	21:25:22.000	124.991	32.620	-0.027
2019/05/12	21:29:34.000	123.281	32.512	-0.048
2019/05/12	21:33:46.000	121.571	32.364	-0.070
2019/05/12	21:37:58.000	119.851	32.185	-0.091
2019/05/12	21:42:10.000	118.148	31.920	-0.112
2019/05/12	21:46:22.000	116.450	31.621	-0.132
2019/05/12	21:50:34.000	114.756	31.281	-0.152
2019/05/12	21:54:46.000	113.068	30.890	-0.172
2019/05/12	21:58:58.000	111.395	30.455	-0.190
2019/05/12	22:03:10.000	109.738	29.983	-0.208
2019/05/12	22:07:22.000	108.082	29.467	-0.226
2019/05/12	22:11:34.000	106.448	28.928	-0.243
2019/05/12	22:15:46.000	104.833	28.321	-0.259
2019/05/12	22:19:58.000	103.221	27.708	-0.274
2019/05/12	22:24:10.000	101.630	27.054	-0.288
2019/05/12	22:28:22.000	100.056	26.377	-0.302
2019/05/12	22:32:34.000	98.496	25.654	-0.315
2019/05/12	22:36:46.000	96.953	24.917	-0.327
2019/05/12	22:40:58.000	95.418	24.170	-0.338
2019/05/12	22:45:10.000	93.897	23.386	-0.347
2019/05/12	22:49:22.000	92.388	22.580	-0.356
2019/05/12	22:53:34.000	90.894	21.756	-0.364
2019/05/12	22:57:46.000	89.412	20.935	-0.371
2019/05/12	23:01:58.000	87.919	20.089	-0.377
2019/05/12	23:06:10.000	86.435	19.228	-0.382
2019/05/12	23:10:22.000	84.979	18.370	-0.385
2019/05/12	23:14:34.000	83.502	17.511	-0.388
2019/05/12	23:18:46.000	82.038	16.645	-0.389
2019/05/12	23:22:58.000	80.583	15.772	-0.389
2019/05/12	23:27:10.000	79.125	14.903	-0.388
2019/05/12	23:31:22.000	77.657	14.050	-0.386
2019/05/12	23:35:34.000	76.197	13.193	-0.382
2019/05/12	23:39:46.000	74.735	12.347	-0.377
2019/05/12	23:43:58.000	73.255	11.512	-0.371
2019/05/12	23:48:10.000	71.780	10.681	-0.364
2019/05/12	23:52:22.000	70.291	9.881	-0.356
2019/05/12	23:56:34.000	68.802	9.108	-0.346
2019/05/13	00:00:46.000	67.303	8.336	-0.335
2019/05/13	00:04:58.000	65.785	7.609	-0.324
2019/05/13	00:09:10.000	64.268	6.891	-0.310
2019/05/13	00:13:22.000	62.737	6.212	-0.296
2019/05/13	00:17:34.000	61.203	5.560	-0.281
2019/05/13	00:21:46.000	59.646	4.947	-0.264
2019/05/13	00:25:58.000	58.092	4.369	-0.247

2019/05/13	00:30:10.000	56.527	3.843	-0.229
2019/05/13	00:34:22.000	54.953	3.327	-0.209
2019/05/13	00:38:34.000	53.361	2.870	-0.189
2019/05/13	00:42:46.000	51.768	2.478	-0.168
2019/05/13	00:46:58.000	50.184	2.123	-0.146
2019/05/13	00:51:10.000	48.571	1.806	-0.124
2019/05/13	00:55:22.000	46.969	1.549	-0.101
2019/05/13	00:59:34.000	45.373	1.334	-0.077
2019/05/13	01:03:46.000	43.752	1.183	-0.053
2019/05/13	01:07:58.000	42.157	1.081	-0.028
2019/05/13	01:12:10.000	40.565	1.039	-0.003

MOLNIYA 3-50 MEASUREMENTS FROM TRUTH ORBIT

Least-squares orbit determination
Measurements

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/13	03:08:26.000	76.293	70.304	2.918
2019/05/13	03:14:26.000	69.776	70.121	2.825
2019/05/13	03:20:26.000	64.098	69.785	2.731
2019/05/13	03:26:26.000	59.166	69.351	2.638
2019/05/13	03:32:26.000	54.879	68.858	2.547
2019/05/13	03:38:26.000	51.142	68.331	2.458
2019/05/13	03:44:26.000	47.870	67.790	2.371
2019/05/13	03:50:26.000	44.993	67.245	2.286
2019/05/13	03:56:26.000	42.450	66.706	2.204
2019/05/13	04:02:26.000	40.192	66.177	2.123
2019/05/13	04:08:26.000	38.178	65.662	2.045
2019/05/13	04:14:26.000	36.374	65.164	1.968
2019/05/13	04:20:26.000	34.752	64.683	1.893
2019/05/13	04:26:26.000	33.289	64.220	1.820
2019/05/13	04:32:26.000	31.965	63.776	1.749
2019/05/13	04:38:26.000	30.765	63.351	1.679
2019/05/13	04:44:26.000	29.675	62.945	1.611
2019/05/13	04:50:26.000	28.682	62.557	1.544
2019/05/13	04:56:26.000	27.777	62.187	1.479
2019/05/13	05:02:26.000	26.951	61.835	1.414
2019/05/13	05:08:26.000	26.198	61.501	1.351
2019/05/13	05:14:26.000	25.509	61.183	1.289
2019/05/13	05:20:26.000	24.880	60.883	1.228
2019/05/13	05:26:26.000	24.306	60.599	1.167
2019/05/13	05:32:26.000	23.783	60.331	1.108
2019/05/13	05:38:26.000	23.306	60.079	1.049
2019/05/13	05:44:26.000	22.872	59.843	0.991
2019/05/13	05:50:26.000	22.478	59.622	0.934
2019/05/13	05:56:26.000	22.122	59.417	0.878
2019/05/13	06:02:26.000	21.801	59.226	0.822
2019/05/13	06:08:26.000	21.513	59.050	0.766
2019/05/13	06:14:26.000	21.256	58.889	0.711
2019/05/13	06:20:26.000	21.028	58.743	0.657
2019/05/13	06:26:26.000	20.827	58.611	0.603
2019/05/13	06:32:26.000	20.652	58.493	0.549
2019/05/13	06:38:26.000	20.503	58.389	0.496
2019/05/13	06:44:26.000	20.376	58.300	0.443
2019/05/13	06:50:26.000	20.273	58.225	0.390
2019/05/13	06:56:26.000	20.191	58.164	0.338
2019/05/13	07:02:26.000	20.129	58.117	0.285
2019/05/13	07:08:26.000	20.088	58.085	0.233
2019/05/13	07:14:26.000	20.066	58.066	0.181

2019/05/13	07:20:26.000	20.062	58.062	0.129
2019/05/13	07:26:26.000	20.077	58.072	0.077
2019/05/13	07:32:26.000	20.110	58.096	0.026
2019/05/13	07:38:26.000	20.159	58.134	-0.026
2019/05/13	07:44:26.000	20.226	58.187	-0.078
2019/05/13	07:50:26.000	20.310	58.255	-0.130
2019/05/13	07:56:26.000	20.410	58.337	-0.182
2019/05/13	08:02:26.000	20.528	58.433	-0.234
2019/05/13	08:08:26.000	20.661	58.545	-0.286
2019/05/13	08:14:26.000	20.812	58.672	-0.338
2019/05/13	08:20:26.000	20.980	58.814	-0.391
2019/05/13	08:26:26.000	21.165	58.972	-0.443
2019/05/13	08:32:26.000	21.368	59.145	-0.496
2019/05/13	08:38:26.000	21.589	59.334	-0.550
2019/05/13	08:44:26.000	21.830	59.539	-0.603
2019/05/13	08:50:26.000	22.090	59.760	-0.657
2019/05/13	08:56:26.000	22.372	59.998	-0.712
2019/05/13	09:02:26.000	22.676	60.253	-0.767
2019/05/13	09:08:26.000	23.003	60.525	-0.822
2019/05/13	09:14:26.000	23.357	60.815	-0.878
2019/05/13	09:20:26.000	23.737	61.122	-0.934
2019/05/13	09:26:26.000	24.147	61.448	-0.991
2019/05/13	09:32:26.000	24.590	61.793	-1.049
2019/05/13	09:38:26.000	25.069	62.156	-1.108
2019/05/13	09:44:26.000	25.588	62.540	-1.167
2019/05/13	09:50:26.000	26.151	62.943	-1.227
2019/05/13	09:56:26.000	26.763	63.366	-1.288
2019/05/13	10:02:26.000	27.431	63.811	-1.349
2019/05/13	10:08:26.000	28.161	64.276	-1.412
2019/05/13	10:14:26.000	28.963	64.764	-1.476
2019/05/13	10:20:26.000	29.848	65.273	-1.541
2019/05/13	10:26:26.000	30.826	65.804	-1.607
2019/05/13	10:32:26.000	31.915	66.357	-1.674
2019/05/13	10:38:26.000	33.130	66.931	-1.743
2019/05/13	10:44:26.000	34.495	67.527	-1.813
2019/05/13	10:50:26.000	36.035	68.142	-1.884
2019/05/13	10:56:26.000	37.783	68.774	-1.957
2019/05/13	11:02:26.000	39.780	69.421	-2.031
2019/05/13	11:08:26.000	42.072	70.076	-2.108
2019/05/13	11:14:26.000	44.718	70.733	-2.185
2019/05/13	11:20:26.000	47.788	71.380	-2.264
2019/05/13	11:26:26.000	51.364	72.002	-2.345
2019/05/13	11:32:26.000	55.537	72.576	-2.427
2019/05/13	11:38:26.000	60.399	73.070	-2.510
2019/05/13	11:44:26.000	66.030	73.439	-2.594
2019/05/13	11:50:26.000	72.467	73.625	-2.678
2019/05/13	11:56:26.000	79.669	73.553	-2.762

2019/05/13	12:02:26.000	87.480	73.134	-2.843
2019/05/13	12:08:26.000	95.617	72.266	-2.919
2019/05/13	12:14:26.000	103.715	70.846	-2.988
2019/05/13	12:20:26.000	111.410	68.764	-3.043
2019/05/13	12:26:26.000	118.423	65.903	-3.076
2019/05/13	12:32:26.000	124.598	62.122	-3.076
2019/05/13	12:38:26.000	129.894	57.236	-3.020
2019/05/13	12:44:26.000	134.343	50.999	-2.879
2019/05/13	12:50:26.000	138.018	43.094	-2.604
2019/05/13	12:56:26.000	140.997	33.164	-2.130
2019/05/13	13:02:26.000	143.350	20.932	-1.389

MOLNIYA 3-50 MEASUREMENTS WITH ADDED NOISE

Least-squares orbit determination
Measurements

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/13	03:08:26.000	76.298	70.308	2.919
2019/05/13	03:14:26.000	69.778	70.118	2.833
2019/05/13	03:20:26.000	64.098	69.794	2.732
2019/05/13	03:26:26.000	59.170	69.344	2.635
2019/05/13	03:32:26.000	54.885	68.855	2.550
2019/05/13	03:38:26.000	51.140	68.331	2.452
2019/05/13	03:44:26.000	47.864	67.794	2.371
2019/05/13	03:50:26.000	44.989	67.241	2.291
2019/05/13	03:56:26.000	42.450	66.708	2.206
2019/05/13	04:02:26.000	40.196	66.179	2.116
2019/05/13	04:08:26.000	38.180	65.668	2.048
2019/05/13	04:14:26.000	36.362	65.162	1.966
2019/05/13	04:20:26.000	34.753	64.680	1.893
2019/05/13	04:26:26.000	33.297	64.224	1.822
2019/05/13	04:32:26.000	31.961	63.776	1.750
2019/05/13	04:38:26.000	30.762	63.350	1.681
2019/05/13	04:44:26.000	29.683	62.933	1.606
2019/05/13	04:50:26.000	28.688	62.560	1.543
2019/05/13	04:56:26.000	27.779	62.182	1.473
2019/05/13	05:02:26.000	26.953	61.843	1.418
2019/05/13	05:08:26.000	26.195	61.505	1.350
2019/05/13	05:14:26.000	25.505	61.181	1.286
2019/05/13	05:20:26.000	24.879	60.886	1.222
2019/05/13	05:26:26.000	24.308	60.605	1.168
2019/05/13	05:32:26.000	23.786	60.331	1.118
2019/05/13	05:38:26.000	23.310	60.070	1.049
2019/05/13	05:44:26.000	22.863	59.848	0.993
2019/05/13	05:50:26.000	22.482	59.627	0.943
2019/05/13	05:56:26.000	22.121	59.416	0.881
2019/05/13	06:02:26.000	21.806	59.221	0.819
2019/05/13	06:08:26.000	21.513	59.038	0.772
2019/05/13	06:14:26.000	21.251	58.895	0.707
2019/05/13	06:20:26.000	21.031	58.747	0.656
2019/05/13	06:26:26.000	20.830	58.610	0.606
2019/05/13	06:32:26.000	20.647	58.489	0.556
2019/05/13	06:38:26.000	20.500	58.391	0.488
2019/05/13	06:44:26.000	20.374	58.296	0.442
2019/05/13	06:50:26.000	20.278	58.230	0.396
2019/05/13	06:56:26.000	20.192	58.167	0.332
2019/05/13	07:02:26.000	20.127	58.124	0.290
2019/05/13	07:08:26.000	20.090	58.078	0.235
2019/05/13	07:14:26.000	20.068	58.064	0.183

2019/05/13	07:20:26.000	20.075	58.059	0.121
2019/05/13	07:26:26.000	20.064	58.068	0.074
2019/05/13	07:32:26.000	20.104	58.106	0.031
2019/05/13	07:38:26.000	20.163	58.136	-0.040
2019/05/13	07:44:26.000	20.235	58.187	-0.082
2019/05/13	07:50:26.000	20.300	58.255	-0.119
2019/05/13	07:56:26.000	20.412	58.341	-0.183
2019/05/13	08:02:26.000	20.533	58.426	-0.229
2019/05/13	08:08:26.000	20.664	58.544	-0.285
2019/05/13	08:14:26.000	20.820	58.672	-0.334
2019/05/13	08:20:26.000	20.973	58.820	-0.382
2019/05/13	08:26:26.000	21.162	58.966	-0.448
2019/05/13	08:32:26.000	21.371	59.141	-0.497
2019/05/13	08:38:26.000	21.582	59.339	-0.554
2019/05/13	08:44:26.000	21.834	59.542	-0.604
2019/05/13	08:50:26.000	22.092	59.758	-0.659
2019/05/13	08:56:26.000	22.378	59.998	-0.717
2019/05/13	09:02:26.000	22.684	60.248	-0.765
2019/05/13	09:08:26.000	22.999	60.523	-0.820
2019/05/13	09:14:26.000	23.362	60.813	-0.876
2019/05/13	09:20:26.000	23.736	61.122	-0.940
2019/05/13	09:26:26.000	24.147	61.455	-0.996
2019/05/13	09:32:26.000	24.584	61.800	-1.043
2019/05/13	09:38:26.000	25.073	62.149	-1.100
2019/05/13	09:44:26.000	25.585	62.533	-1.170
2019/05/13	09:50:26.000	26.152	62.946	-1.226
2019/05/13	09:56:26.000	26.761	63.375	-1.291
2019/05/13	10:02:26.000	27.431	63.807	-1.347
2019/05/13	10:08:26.000	28.152	64.280	-1.418
2019/05/13	10:14:26.000	28.963	64.758	-1.482
2019/05/13	10:20:26.000	29.838	65.268	-1.540
2019/05/13	10:26:26.000	30.829	65.811	-1.604
2019/05/13	10:32:26.000	31.904	66.352	-1.675
2019/05/13	10:38:26.000	33.117	66.930	-1.746
2019/05/13	10:44:26.000	34.496	67.530	-1.808
2019/05/13	10:50:26.000	36.034	68.141	-1.877
2019/05/13	10:56:26.000	37.793	68.772	-1.956
2019/05/13	11:02:26.000	39.775	69.421	-2.028
2019/05/13	11:08:26.000	42.069	70.077	-2.107
2019/05/13	11:14:26.000	44.715	70.730	-2.176
2019/05/13	11:20:26.000	47.783	71.378	-2.272
2019/05/13	11:26:26.000	51.364	72.004	-2.348
2019/05/13	11:32:26.000	55.539	72.573	-2.421
2019/05/13	11:38:26.000	60.399	73.074	-2.500
2019/05/13	11:44:26.000	66.032	73.446	-2.593
2019/05/13	11:50:26.000	72.465	73.620	-2.680
2019/05/13	11:56:26.000	79.675	73.557	-2.767

2019/05/13	12:02:26.000	87.479	73.127	-2.841
2019/05/13	12:08:26.000	95.622	72.262	-2.933
2019/05/13	12:14:26.000	103.718	70.862	-2.985
2019/05/13	12:20:26.000	111.411	68.766	-3.038
2019/05/13	12:26:26.000	118.433	65.911	-3.077
2019/05/13	12:32:26.000	124.598	62.123	-3.080
2019/05/13	12:38:26.000	129.899	57.237	-3.013
2019/05/13	12:44:26.000	134.342	51.007	-2.888
2019/05/13	12:50:26.000	138.027	43.086	-2.609
2019/05/13	12:56:26.000	140.997	33.157	-2.128
2019/05/13	13:02:26.000	143.353	20.938	-1.391

MOLNIYA 3-50 MEASUREMENTS FROM ESTIMATED ORBIT

Least-squares orbit determination
Measurements

Date	UTC	Az(deg)	El(deg)	Range rate (km/s)
2019/05/13	03:08:26.000	76.226	70.299	2.915
2019/05/13	03:14:26.000	69.714	70.115	2.822
2019/05/13	03:20:26.000	64.041	69.777	2.728
2019/05/13	03:26:26.000	59.113	69.342	2.635
2019/05/13	03:32:26.000	54.829	68.847	2.544
2019/05/13	03:38:26.000	51.095	68.319	2.455
2019/05/13	03:44:26.000	47.827	67.776	2.367
2019/05/13	03:50:26.000	44.952	67.231	2.282
2019/05/13	03:56:26.000	42.412	66.690	2.200
2019/05/13	04:02:26.000	40.156	66.160	2.119
2019/05/13	04:08:26.000	38.144	65.644	2.040
2019/05/13	04:14:26.000	36.342	65.144	1.964
2019/05/13	04:20:26.000	34.723	64.662	1.889
2019/05/13	04:26:26.000	33.262	64.198	1.816
2019/05/13	04:32:26.000	31.941	63.753	1.744
2019/05/13	04:38:26.000	30.743	63.327	1.674
2019/05/13	04:44:26.000	29.655	62.919	1.606
2019/05/13	04:50:26.000	28.665	62.530	1.539
2019/05/13	04:56:26.000	27.762	62.159	1.473
2019/05/13	05:02:26.000	26.939	61.805	1.409
2019/05/13	05:08:26.000	26.188	61.470	1.346
2019/05/13	05:14:26.000	25.502	61.151	1.283
2019/05/13	05:20:26.000	24.876	60.849	1.222
2019/05/13	05:26:26.000	24.305	60.564	1.162
2019/05/13	05:32:26.000	23.784	60.295	1.102
2019/05/13	05:38:26.000	23.310	60.042	1.043
2019/05/13	05:44:26.000	22.880	59.804	0.985
2019/05/13	05:50:26.000	22.490	59.582	0.928
2019/05/13	05:56:26.000	22.137	59.375	0.871
2019/05/13	06:02:26.000	21.820	59.183	0.815
2019/05/13	06:08:26.000	21.535	59.006	0.760
2019/05/13	06:14:26.000	21.282	58.844	0.705
2019/05/13	06:20:26.000	21.057	58.696	0.650
2019/05/13	06:26:26.000	20.861	58.563	0.596
2019/05/13	06:32:26.000	20.691	58.444	0.542
2019/05/13	06:38:26.000	20.546	58.339	0.488
2019/05/13	06:44:26.000	20.424	58.249	0.435
2019/05/13	06:50:26.000	20.325	58.172	0.382
2019/05/13	06:56:26.000	20.248	58.110	0.330
2019/05/13	07:02:26.000	20.192	58.063	0.277
2019/05/13	07:08:26.000	20.156	58.029	0.225
2019/05/13	07:14:26.000	20.140	58.009	0.173

2019/05/13	07:20:26.000	20.143	58.004	0.121
2019/05/13	07:26:26.000	20.164	58.013	0.068
2019/05/13	07:32:26.000	20.203	58.036	0.016
2019/05/13	07:38:26.000	20.259	58.074	-0.035
2019/05/13	07:44:26.000	20.333	58.126	-0.087
2019/05/13	07:50:26.000	20.425	58.193	-0.140
2019/05/13	07:56:26.000	20.533	58.274	-0.192
2019/05/13	08:02:26.000	20.659	58.371	-0.244
2019/05/13	08:08:26.000	20.801	58.482	-0.296
2019/05/13	08:14:26.000	20.961	58.608	-0.349
2019/05/13	08:20:26.000	21.138	58.750	-0.402
2019/05/13	08:26:26.000	21.334	58.907	-0.455
2019/05/13	08:32:26.000	21.548	59.080	-0.508
2019/05/13	08:38:26.000	21.781	59.268	-0.562
2019/05/13	08:44:26.000	22.033	59.473	-0.616
2019/05/13	08:50:26.000	22.307	59.694	-0.670
2019/05/13	08:56:26.000	22.602	59.932	-0.725
2019/05/13	09:02:26.000	22.921	60.187	-0.780
2019/05/13	09:08:26.000	23.264	60.459	-0.836
2019/05/13	09:14:26.000	23.634	60.749	-0.892
2019/05/13	09:20:26.000	24.032	61.056	-0.949
2019/05/13	09:26:26.000	24.462	61.382	-1.006
2019/05/13	09:32:26.000	24.926	61.727	-1.064
2019/05/13	09:38:26.000	25.427	62.091	-1.123
2019/05/13	09:44:26.000	25.970	62.474	-1.183
2019/05/13	09:50:26.000	26.560	62.877	-1.243
2019/05/13	09:56:26.000	27.201	63.301	-1.305
2019/05/13	10:02:26.000	27.900	63.745	-1.367
2019/05/13	10:08:26.000	28.666	64.210	-1.430
2019/05/13	10:14:26.000	29.507	64.697	-1.494
2019/05/13	10:20:26.000	30.434	65.205	-1.560
2019/05/13	10:26:26.000	31.461	65.735	-1.627
2019/05/13	10:32:26.000	32.602	66.285	-1.694
2019/05/13	10:38:26.000	33.877	66.857	-1.764
2019/05/13	10:44:26.000	35.309	67.449	-1.834
2019/05/13	10:50:26.000	36.925	68.058	-1.907
2019/05/13	10:56:26.000	38.759	68.683	-1.980
2019/05/13	11:02:26.000	40.852	69.320	-2.055
2019/05/13	11:08:26.000	43.254	69.962	-2.132
2019/05/13	11:14:26.000	46.025	70.601	-2.210
2019/05/13	11:20:26.000	49.233	71.225	-2.290
2019/05/13	11:26:26.000	52.962	71.815	-2.372
2019/05/13	11:32:26.000	57.296	72.346	-2.455
2019/05/13	11:38:26.000	62.321	72.783	-2.538
2019/05/13	11:44:26.000	68.100	73.078	-2.623
2019/05/13	11:50:26.000	74.648	73.169	-2.707
2019/05/13	11:56:26.000	81.895	72.977	-2.789

2019/05/13	12:02:26.000	89.657	72.411	-2.869
2019/05/13	12:08:26.000	97.644	71.368	-2.943
2019/05/13	12:14:26.000	105.503	69.741	-3.008
2019/05/13	12:20:26.000	112.906	67.418	-3.056
2019/05/13	12:26:26.000	119.615	64.270	-3.079
2019/05/13	12:32:26.000	125.506	60.140	-3.061
2019/05/13	12:38:26.000	130.554	54.821	-2.979
2019/05/13	12:44:26.000	134.797	48.038	-2.794
2019/05/13	12:50:26.000	138.303	39.454	-2.455
2019/05/13	12:56:26.000	141.144	28.721	-1.890
2019/05/13	13:02:26.000	143.382	15.644	-1.039

APPENDIX B. SIMULATION RESULTS

ISS RANGE RATE ONLY

```
Summary:
  truth      estimated    correction       final
x[km]
    -6197.3     -6196.8           -0.5     -6197.3

y[km]
    -1366.8     -1368.8            2.0     -1366.8

z[km]
     2416.8      2417.8           -1.0      2416.8

vxv[km/s]
     3.0836      3.0823         1.2900      3.0836

vyv[km/s]
    -4.5285     -4.5275        -1.0100     -4.5285

vzv[km/s]
     5.3558      5.3553         0.4900      5.3557

Elapsed time is 365.792232 seconds.
```

ISS ANGLES ONLY

```
Summary:
  truth      estimated    correction       final
x[km]
    -6197.3     -6196.8           -0.5     -6197.3

y[km]
    -1366.8     -1368.8            2.0     -1366.8

z[km]
     2416.8      2417.8           -1.0      2416.8

vxv[km/s]
     3.0836      3.0823         1.2900      3.0836

vyv[km/s]
    -4.5285     -4.5275        -0.9900     -4.5285

vzv[km/s]
     5.3558      5.3553         0.4900      5.3557

Elapsed time is 90.013438 seconds.
```

ISS RANGE RATE AND ANGLES

```
Summary:
  truth      estimated    correction       final
x[km]
    -6197.6     -6197.1           -0.5     -6197.6

y[km]
    -1366.2     -1368.2            2.0     -1366.2

z[km]
     2416.3      2417.3           -1.0      2416.3

vxv[km/s]
     3.0828      3.0815         0.0013      3.0828

vyv[km/s]
    -4.5287     -4.5277        -0.0010     -4.5287

vzv[km/s]
     5.3561      5.3556         0.0005      5.3561

Elapsed time is 446.626837 seconds.
```

NAVSTAR RANGE RATE ONLY

```
Summary:
  truth      estimated    correction       final
x[km]
    14277.2     14273.2            4.0     14277.2

y[km]
    15437.6     15435.1            2.5     15437.6

z[km]
   -16214.8    -16211.8           -3.0    -16214.8

vxv[km/s]
    -3.1690     -3.1706         1.6100     -3.1690

vyv[km/s]
     0.7075      0.7085        -0.9900      0.7075

vzv[km/s]
    -2.1149     -2.1169         2.0100     -2.1149

Elapsed time is 4416.132813 seconds.
```

```
Summary:
  truth      estimated   correction      final
x[km]
    14277.2     14272.2            5.0     14277.2

y[km]
    15437.6     15434.6            3.0     15437.6

z[km]
   -16214.8    -16210.8           -4.0    -16214.8

vxv[km/s]
    -3.1690     -3.1706         1.5900     -3.1690

vyv[km/s]
     0.7075      0.7085        -0.9901      0.7075

vzv[km/s]
    -2.1149     -2.1174         2.4900     -2.1149

Elapsed time is 5835.171982 seconds.
```

NAVSTAR RANGE RATE AND ANGLES

```
Summary:
  truth      estimated    correction       final
x[km]
    14277.2     14272.2            5.0     14277.2

y[km]
    15437.6     15434.6            3.0     15437.6

z[km]
   -16214.8    -16210.8           -4.0    -16214.8

vxv[km/s]
    -3.1690     -3.1706         1.5900     -3.1690

vyv[km/s]
     0.7075      0.7085        -0.9900      0.7075

vzv[km/s]
    -2.1149     -2.1174         2.4900     -2.1149

Elapsed time is 2639.753016 seconds.
```

MOLNIYA RANGE RATE ONLY

```
Summary:
  truth      estimated   correction      final
x[km]
   -19341.5    -19335.5           -6.0    -19341.5

y[km]
     1826.6      1824.6            2.0      1826.6

z[km]
    14880.8     14885.8           -5.0     14880.8

vxv[km/s]
    -1.1162     -1.1132        -2.9900     -1.1162

vyv[km/s]
    -1.6178     -1.6168        -0.9900     -1.6178

vzv[km/s]
     3.7013      3.6988         2.5100      3.7013

Elapsed time is 2514.525525 seconds.
```

MOLNIYA ANGLES ONLY

Summary:

	truth	estimated	correction	final
x[km]	-19347.0	-19341.0	-6.0	-19347.0
y[km]	1824.3	1822.3	2.0	1824.3
z[km]	14893.2	14898.2	-5.0	14893.2
vxv[km/s]	-1.1151	-1.1121	-2.9900	-1.1150
vyv[km/s]	-1.6175	-1.6165	-0.9900	-1.6175
vzv[km/s]	3.6997	3.6972	2.4900	3.6997

MOLNIYA RANGE RATE AND ANGLES

```
Summary:
  truth      estimated    correction       final
x[km]
   -19347.0    -19341.0           -6.0    -19347.0

y[km]
     1824.3      1822.3            2.0      1824.3

z[km]
    14893.2     14898.2           -5.0     14893.2

vxv[km/s]
    -1.1151     -1.1121        -2.9900     -1.1150

vyv[km/s]
    -1.6175     -1.6165        -0.9900     -1.6175

vzv[km/s]
     3.6997      3.6972         2.4901      3.6997



Elapsed time is 3539.886951 seconds.
```

APPENDIX C. RESULTS FROM PREVIOUS BUILD OF LMF

ISS using 10 observations of range rate

```
Summary:
  predicted     estimated   correction      final
x[km]
     -385.5      -375.5          -10.4      -385.9

y[km]
     4511.7      4506.7            3.5      4510.3

z[km]
     5294.2      5295.2            0.1      5295.3

vxv[km/s]
    -9.3367     -9.3380         0.0032     -9.3349

vyv[km/s]
     1.1645      1.1675         0.0018      1.1693

vzv[km/s]
    -2.8215     -2.8220        -0.0030     -2.8250

Root Mean Square:
RMS = 0.001813
Elapsed time is 78.505011 seconds.
```

ISS using 10 observations using range rate, Azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
     -385.5      -375.5          -10.0      -385.5

y[km]
     4511.7      4506.7            5.0      4511.7

z[km]
     5294.2      5295.2           -1.0      5294.2

vxv[km/s]
    -9.3367     -9.3380         0.0013     -9.3368

vyv[km/s]
     1.1645      1.1675        -0.0031      1.1644

vzv[km/s]
    -2.8215     -2.8220         0.0006     -2.8214

Root Mean Square:
RMS = 0.004830
```

ISS using 20 observations with range rate only

```
Summary:
  predicted     estimated   correction      final
x[km]
     -385.5      -375.5          -10.1      -385.5

y[km]
     4511.7      4506.7            4.8      4511.5

z[km]
     5294.2      5295.2           -0.9      5294.4

vxv[km/s]
    -9.3367     -9.3380         0.0015     -9.3365

vyv[km/s]
     1.1645      1.1675        -0.0023      1.1652

vzv[km/s]
    -2.8215     -2.8220         0.0001     -2.8218

Root Mean Square:
RMS = 0.000577


Elapsed time is 235.618692 seconds.
```

ISS using 20 observations of range rate, azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
     -385.5      -375.5         -10.0      -385.5

y[km]
     4511.7      4506.7           4.9      4511.7

z[km]
     5294.2      5295.2          -1.0      5294.3

vxv[km/s]
    -9.3367     -9.3380        0.0012     -9.3368

vyv[km/s]
     1.1645      1.1675       -0.0031      1.1644

vzv[km/s]
    -2.8215     -2.8220        0.0007     -2.8213

Root Mean Square:
RMS = 0.000528
```

ISS using 40 observations using range rate only

```
Summary:

  predicted     estimated   correction      final
x[km]
     -385.5      -375.5           -7.1      -382.6

y[km]
     4511.7      4506.7            7.1      4513.8

z[km]
     5294.2      5295.2           -6.6      5288.6

vxv[km/s]
    -9.3367     -9.3380         0.0052     -9.3328

vyv[km/s]
     1.1645      1.1675         0.0076      1.1751

vzv[km/s]
    -2.8215     -2.8220        -0.0045     -2.8265

Root Mean Square:
RMS = 0.062329
Elapsed time is 607.324302 seconds.
```

ISS using 40 observations of range rate, azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
     -385.5      -375.5          -8.2      -383.6

y[km]
     4511.7      4506.7           8.4      4515.1

z[km]
     5294.2      5295.2          -4.8      5290.4

vxv[km/s]
    -9.3367     -9.3380        -0.0010     -9.3390

vyv[km/s]
     1.1645      1.1675        -0.0075      1.1600

vzv[km/s]
    -2.8215     -2.8220         0.0056     -2.8163

Root Mean Square:
RMS = 0.238215
```

NAVSTAR-77 using 10 observations of range rate only

```
Summary:

  predicted     estimated   correction      final
x[km]
   -12533.9    -12523.9           -5.7    -12529.6

y[km]
    -7203.2     -7208.2            0.2     -7208.0

z[km]
     3655.4      3656.4           -0.2      3656.2

vxv[km/s]
    -0.3214     -0.3227         0.0016     -0.3211

vyv[km/s]
    -1.9193     -1.9163         0.0002     -1.9161

vzv[km/s]
     2.3325      2.3320         0.0033      2.3354

Root Mean Square:
RMS = 0.002741


Elapsed time is 85.395238 seconds.
```

NAVSTAR-77 using 10 observations of range rate, azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
   -12533.9    -12523.9          -12.6    -12536.6

y[km]
    -7203.2     -7208.2            3.9     -7204.3

z[km]
     3655.4      3656.4           -1.7      3654.7

vxv[km/s]
    -0.3214     -0.3227         0.0004     -0.3223

vyv[km/s]
    -1.9193     -1.9163         0.0022     -1.9141

vzv[km/s]
     2.3325      2.3320         0.0034      2.3354

Root Mean Square:
RMS = 0.002240
```

NAVSTAR-77 using 20 observations of range rate only

```
Summary:
  predicted     estimated   correction      final
x[km]
   -12533.9    -12523.9           -7.9    -12531.8

y[km]
    -7203.2     -7208.2            0.5     -7207.7

z[km]
     3655.4      3656.4           -0.4      3656.0

vxv[km/s]
    -0.3214     -0.3227         0.0015     -0.3212

vyv[km/s]
    -1.9193     -1.9163         0.0004     -1.9158

vzv[km/s]
     2.3325      2.3320         0.0029      2.3349

Root Mean Square:
RMS = 0.003012
Elapsed time is 318.568321 seconds.
```

NAVSTAR-77 using 20 observations of range rate, azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
   -12533.9    -12523.9           -9.9    -12533.8

y[km]
    -7203.2     -7208.2            3.5     -7204.7

z[km]
     3655.4      3656.4           -2.3      3654.0

vxv[km/s]
    -0.3214     -0.3227         0.0004     -0.3223

vyv[km/s]
    -1.9193     -1.9163         0.0020     -1.9142

vzv[km/s]
     2.3325      2.3320         0.0043      2.3363

Root Mean Square:
RMS = 0.001599
```

NAVSTAR-77 using 40 observations of range rate only

```
Summary:
  predicted     estimated    correction      final
x[km]
   -12533.9    -12523.9           -2.8    -12526.8

y[km]
    -7203.2     -7208.2            1.1     -7207.1

z[km]
     3655.4      3656.4           -0.8      3655.5

vxv[km/s]
    -0.3214     -0.3227         0.0014     -0.3213

vyv[km/s]
    -1.9193     -1.9163         0.0005     -1.9158

vzv[km/s]
     2.3325      2.3320         0.0051      2.3371

Root Mean Square:
RMS = 0.001744
Elapsed time is 189.575496 seconds.
```

NAVSTAR-77 using 40 observations of range rate, azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
   -12533.9    -12523.9          -11.5    -12535.4

y[km]
    -7203.2     -7208.2            3.4     -7204.8

z[km]
     3655.4      3656.4           -2.5      3653.9

vxv[km/s]
    -0.3214     -0.3227         0.0004     -0.3223

vyv[km/s]
    -1.9193     -1.9163         0.0018     -1.9145

vzv[km/s]
     2.3325      2.3320         0.0036      2.3356

Root Mean Square:
RMS = 0.000758
```

MOLNIYA 3-50 using 10 observations of range rate

```
Summary:

  predicted     estimated    correction      final
x[km]
    -8447.2     -8437.2          -0.2     -8437.4

y[km]
     5226.3      5221.3           26.7      5248.0

z[km]
    -2522.6     -2521.6          -0.2     -2521.8

vxv[km/s]
    -6.1943     -6.1956        -0.0012     -6.1968

vyv[km/s]
     0.0357      0.0387         0.0022      0.0409

vzv[km/s]
     4.7314      4.7309        -0.0031      4.7278

Root Mean Square:
RMS = 0.002410
Elapsed time is 38.993514 seconds.
```

MOLNIYA -3-50 using 10 observation of range rate, azimuth and elevation

Summary:

	predicted	estimated	correction	final
x[km]	-8447.2	-8437.2	-2.2	-8439.3
y[km]	5226.3	5221.3	22.0	5243.3
z[km]	-2522.6	-2521.6	0.6	-2521.0
vxv[km/s]	-6.1943	-6.1956	0.0002	-6.1954
vyv[km/s]	0.0357	0.0387	0.0027	0.0415
vzv[km/s]	4.7314	4.7309	-0.0017	4.7291

Root Mean Square:

RMS = 0.004025

MOLNIYA -3-50 using 20 observation of range rate

```
Summary:
  predicted     estimated   correction      final
x[km]
    -8447.2     -8437.2          -0.2     -8437.4

y[km]
     5226.3      5221.3          30.5      5251.8

z[km]
    -2522.6     -2521.6          -0.1     -2521.7

vxv[km/s]
    -6.1943     -6.1956       -0.0011     -6.1967

vyv[km/s]
     0.0357      0.0387        0.0019      0.0407

vzv[km/s]
     4.7314      4.7309       -0.0029      4.7280

Root Mean Square:
RMS = 0.004610

Elapsed time is 286.099713 seconds.
```

MOLNIYA -3-50 using 20 observation of range rate, azimuth and elevation

```
Summary:
  predicted     estimated   correction      final
x[km]
    -8447.2     -8437.2          -2.1     -8439.2

y[km]
     5226.3      5221.3          27.7      5249.0

z[km]
    -2522.6     -2521.6           1.1     -2520.4

vxv[km/s]
    -6.1943     -6.1956        0.0005     -6.1951

vyv[km/s]
     0.0357      0.0387        0.0023      0.0410

vzv[km/s]
     4.7314      4.7309       -0.0014      4.7295

Root Mean Square:
RMS = 0.007826
```

MOLNIYA -3-50 using 40 observation of range rate

```
Summary:
  predicted     estimated   correction      final
x[km]
    -8447.2     -8437.2          -0.8     -8438.0

y[km]
     5226.3      5221.3          22.6      5243.9

z[km]
    -2522.6     -2521.6           0.3     -2521.3

vxv[km/s]
    -6.1943     -6.1956       -0.0008     -6.1964

vyv[km/s]
     0.0357      0.0387        0.0019      0.0407

vzv[km/s]
     4.7314      4.7309       -0.0029      4.7279

Root Mean Square:

RMS = 0.000900
```

```
Elapsed time is 1523.224941 seconds.
```

MOLNIYA -3-50 using 40 observation of range rate, azimuth and elevation

Summary:

	predicted	estimated	correction	final
x[km]	-8447.2	-8437.2	-2.5	-8439.7
y[km]	5226.3	5221.3	13.5	5234.8
z[km]	-2522.6	-2521.6	1.4	-2520.2
vxv[km/s]	-6.1943	-6.1956	0.0005	-6.1951
vyv[km/s]	0.0357	0.0387	0.0026	0.0413
vzv[km/s]	4.7314	4.7309	-0.0018	4.7291

Root Mean Square:

RMS = 0.015478

APPENDIX D. MATLAB CODE

TEST PROGRAM

Least Squares Orbit Determination

The following program uses a least squares algorithm to predict an orbit with erroneous data. This program simulates measurements using an orbit propagated using data recovered from Gpredict software. Specifically, a file containing the date, time, right-ascension, and declination is used, along with Gauss's method for retrieving a state vector, is used to produce the "measurements."

Goal: Minimize $S(\boldsymbol{x}) = \sum_1^m [f_i(\boldsymbol{x})]^2$

```
clc; clear
format long g
randn('seed',0)
tic
global  PC  AuxParam Cnm Snm eopdata const  ...

constants
get_eopdata
```

1) SATELLITE AND PROPAGATOR OPTIONS:

The user has the choice for which observed parameters they want to use. The cases are as follows:

Choice 1. Range-rate only

Choice 2. Range-rate, azimuth, elevation

Choice 3. Angles only (Azimuth & Elevation)

```
% Satellite options
chooseSat    = 2;                       % [1] ISS [2] NAVSTAR [3] MOLNIYA
n_obs        = 100;                      % number of observations
step         = 252;                      % Time between observations [s]
choice       = 2;                       % Determines which parameter set
to run
sig1         = 0.005*pi/180;            % Noise standard deviation [Az
El]
sig2         = 5;                       % Range rate standard deviation
```

2) MARQUARDT FILTER OPTIONS:

```
% marquardt options
options.bdx    = 25e-5;                 % Perturbation value (used for
```

```
Jacobian)
options.lambda = 0.0001;                % Starting point for Marq.
parameter
options.incr   = 10;                    % Factor for increasing lambda
options.decr   = 0.4;                   % Factor for decreasing lambda
options.maxIter= 29;                    % Maximum amount of iterations
options.eps1   = 1e-4;                  % Gradient convergence Criteria
options.eps2   = 1e-8;                 % Parameter convergence criteria
options.eps3   = 1e-6;                  % RMS criterion
options.eps4   = 1e-20;                 % Acceptence criteria
if choice == 1 || choice == 3
    options.wts = 1/sig1^2;
elseif choice == 2
    options.wts = [];
    for i = 1:n_obs
        options.wts = [options.wts;1/sig1^2;1/sig1^2;1/sig2^2];
    end
end
```

3) LOAD SATELLITE DATA AND CALCULATE INITIAL ESTIMATE STATE VECTOR

There are currently three satellites to choose from. Should the user desire studying an alternative satellite, enter the name of the satellite as done below. Note that each name has 24 characters including spaces. Consult Celestrak for lists of satellites and TLEs.

From here, it is possible to calculate the position of the satellite using Two-Line Element (TLE) data. TLE data from all active satellites is read from Celestrak and stored in the file TLE_DATA.txt. This text file is scanned for the relevant satellite, then the TLE of that satellite is stored in the file new_tle.txt. The state vector is calculated using sgp4.m.

In the TLE, epoch is represented in days (with fraction of day) since Jan. 1 of the current year. To get time since epoch for an event, first days since Jan. 1 of the event is calculated, then epoch is subtracted from that value. The function sgp4.m is used to extract the state vector from the TLE at the event time.

```
% Enter Satellite Event Time (UT):
% passTime = [year month day hour minute second]
if chooseSat == 1
    ft = 145.8*1e6;                         % ISS downlink freq
    satellite = 'ISS (ZARYA)             ';
    get_tle(satellite);
    passTime1 = [2019 5 12 5 36 24];
elseif chooseSat == 2
    ft = 1575.42*1e6;
    satellite = 'NAVSTAR 77 (USA 289)    ';
    get_tle(satellite);
    passTime1 = [2019 5 12 18 45 46];
```

```
else
    satellite = 'MOLNIYA 3-50            ';
    Molniya_tle(satellite);
    passTime1 = [2019 5 13 3 2 26];  % Pass 1
end
sat.year1   = passTime1(1);
sat.month1  = passTime1(2);
sat.day1    = passTime1(3);
sat.hour1   = passTime1(4);
sat.minute1 = passTime1(5);
sat.second1 = passTime1(6);

sat.UT   = sat.hour1 + sat.minute1./60 + sat.second1./(60*60);
sat.MJD0 = Mjday(sat.year1, sat.month1, sat.day1, sat.hour1, sat.minute1,
sat.second1);
ndays = days_past_Jan1(sat.year1, sat.month1, sat.day1, sat.hour1,
sat.minute1, sat.second1);
[r0, v0] = sgp4(ndays);       % [km km/s]
% Epoch state "truth"
Y0_ref = [r0*1e3   v0*1e3] ;  %  [[m]  [m/s]]
```

4) SIMULATE GROUND STATION

Constructing the position of the ground site can be done using Eqn. (5.86) in the Curtis book. For the purposes of this experiment, the latitude, longitude, and altitude of San Jose State University are used for this construction.

```
sat.alt   = .025;              %  Altitude   [km]
sat.phi   = 37.3352*pi/180;       %  Latitude   [rad]
%  East longitude [deg]:
degrees = -121;
minutes = 52;
seconds = 31.19;
%  convert negative (west) longitude to east longitude:
if degrees < 0
    degrees = degrees + 360;
end
%  Express the longitudes as decimal numbers:
sat.EL = degrees + minutes/60 + seconds/3600;
```

5) READ IN EARTH GRAVITY FIELD COEFFICIENTS AND MODEL PARAMETERS

```
% File for planetary and lunar ephemerides
load DE430Coeff.mat
PC = DE430Coeff;
% File for Earth gravity field
load GGM03S.txt
%% read Earth gravity field coefficients
```

```
Cnm = zeros(181,181);
Snm = zeros(181,181);
fid = fopen('GGM03S.txt','r');
for n=0:180
    for m=0:n
        temp = fscanf(fid,'%d %d %f %f %f %f',[6 1]);
        Cnm(n+1,m+1) = temp(3);
        Snm(n+1,m+1) = temp(4);
    end
end
fclose(fid);
% model parameters
AuxParam = struct
('Mjd_UTC',0,'area_solar',0,'area_drag',0,'mass',0,'Cr',0,...

'Cd',0,'n',0,'m',0,'sun',0,'moon',0,'sRad',0,'drag',0,...
                   'planets',0,'SolidEarthTides',0,'OceanTides',0,...
                   'Relativity',0,'n_a',0,'m_a',0,'n_G',0,'m_G',0);
```

6) READ IN EARTH ORIENTATION PARAMETERS:

In this section, the program reads Earth Orientation Parameters (EOPs) from Celestrak. Because Celestrak constantly update the EOPs, downloadilng the code directly through the code will facilitate future uses of the program. In the case that the user is not connected to the internet, a downloaded set of EOP data can be used.

Read Earth Orientation Parameters

```
eopdata;

fid = fopen('eopdata.txt','r');
eopdata = fscanf(fid,'%i %d %d %i %f %f %f %f %f %f %f %f %f',[13 inf]);
fclose(fid);
Label =  ['x','y','z'];              % Used for printing of results
%%
AuxParam.Mjd_UTC    = sat.MJD0;
AuxParam.area_solar = 10*2; % [m^2]
AuxParam.area_drag  = 10*2; % [m^2]
AuxParam.mass       = 2000;  % [kg]
AuxParam.Cr         = 1.0;
AuxParam.Cd         = 2.0;
AuxParam.n          = 10;
AuxParam.m          = 10;
AuxParam.n_a        = 10;
AuxParam.m_a        = 10;
AuxParam.n_G        = 10;
AuxParam.m_G        = 10;
```

```
AuxParam.sun        = 1;
AuxParam.moon       = 1;
AuxParam.sRad       = 0;
AuxParam.drag       = 0;
AuxParam.planets    = 0;
AuxParam.SolidEarthTides = 1;
AuxParam.OceanTides = 0;
AuxParam.Relativity = 0;
```

8) BEGIN LEAST SQUARES ALGORITHM

Beginning the least squares algorithm:

```
% generation of artificial observations from given epoch state
sat.n_obs = n_obs;
sat.choice = choice;
sat.sig1 = sig1;
sat.sig2 = sig2;
for i = 1:n_obs
    t(i) = i*step;
end
t = t(:);
fprintf('Least-squares orbit determination\n\n');
    fprintf('Measurements \n\n');
    if sat.choice == 1
        fprintf('     Date          UTC     Range rate (km/s)\n');
    elseif sat.choice == 2
        fprintf('     Date          UTC        Az(deg)   El(deg)   Range
rate (km/s)\n');
    elseif sat.choice == 3
        fprintf('     Date          UTC        Az(deg)   El(deg)\n');
    end
 for i = 1:length(t)
    Obs(i,:) = get_obs(Y0_ref,t(i),sat,1);    %  Evaluate model using Y0
 end
% [Obs,rr] = get_obs(Y0_ref,MJD0,sat,step);
```

9) SIMULATE PARAMETERS FROM "PREDICTED" ORBIT

Range rate is calculated using

$$\dot{\rho} = \frac{\vec{\rho} \cdot \dot{\vec{\rho}}}{\rho}$$

where

- $\vec{\rho}$ is the range vector in the ECI frame $\vec{\rho} = \vec{r} - \vec{R}$

- $\dot{\vec{\rho}}$ is the time derivative of $\vec{\rho}$ $\dot{\vec{\rho}} = \vec{v} - \left(\vec{\omega_E} \times \vec{R}\right)$
- ρ is the magnitude of $\vec{\rho}$. $\rho = \sqrt{\rho_x^2 + \rho_y^2 + \rho_z^2}$
-

Right-ascension α and declination δ are found using (Curtis, 2014)

$$\delta = \sin^{-1}\left(\frac{\rho_z}{\rho}\right)$$

$$\alpha = \cos^{-1}\left(\frac{\rho_x}{\rho \cos(\delta)}\right)$$

if $\rho_y / \rho < 0$

$$\alpha = 360 - \alpha.$$

Azimuth A and Elevation a are found using Curtis...

Now the discovered quantities are recorded in the observation **Obs** matrix

Plotting:

This code can output 3 plots. The data for the first plot, which plots the predicted and estimated orbits together, should be extracted from the orbit propagator found in the present code. This method is meant to show how the estimated orbit evolves over the course of the least-squares interation. This first plot shows only preliminary orbit. With added orbital perturbations, the second and third plot show the orbit in both the Earth-Centered, Earth-Fixed (ECEF) reference frame and the Earth-Centered Inertial (ECI) frame.

```
% Step = 60;
% N_Step = 420;
%
% [Eph, Eph_ecef] = Workspace_OrbitGen(Step,N_Step, Y0_ref, sat.MJD0);
```

Transformation of simulated range-rate to Doppler shift

To generate the desired "waterfall" plots for Doppler shift, the simulated range-rate measurements need to be transformed into frequency shift measurements. This can be done through modification of the range-rate formula

$$\dot{\rho} = c\left(1 - \frac{f_r}{f_t}\right)$$

where c is the speed of light, f_r is the percieved frequency at the ground site reciever, and f_t is the nominal transmitted frequency. First the Doppler shift equation is solved for f_r

$$\Delta f = f_r - f_t$$

$$f_r = \Delta f + f_t,$$

which is plugged into the $\dot{\rho}$ equation:

$$\dot{\rho} = c\left(1 - \frac{\Delta f + f_t}{f_t}\right)$$

$$= c\left(1 - \frac{\Delta f}{f_t} + \frac{f_t}{f_t}\right)$$

$$= c\left(-\frac{\Delta f}{f_t}\right).$$

This equation can now be solved for Δf

$$\Delta f = -\dot{\rho}\left(\frac{f_t}{c}\right).$$

Orbit determination:

```
Y0_apr = Y0_ref' + [-5e3,-3e3,4e3,-1.6,1,-2.5]';  % Orbit from noisy data
Y0     = Y0_apr;                                  % A priori state vector
rUp =Y0(1:3) + 6e3;
rLow = Y0(1:3)-6e3;
vUp = Y0(4:6)+2;
vLow = Y0(4:6)-2;
upb = [r0+.01 v0+0.00001];
lobs = [r0-.01 v0-0.00001];
options.bnds = [lobs*1e3 upb*1e3];
c = [sat];                                        % Constants used in
marquardt
     fprintf('\nResiduals: \n\n');                % Label for residual
table
     fprintf('     Date          UTC          RA(deg)    Dec(deg)
Range-rate(km/s)\n');
```

The **Collector** function is used to extract the Jacobian **J**, vector of residuals **f**, and the sum of squares of the residuals **SSx**. The variable SSx represents S(x) the sum of squares using the previous initial state, while **SSxq** represents the sum of squares of residuals for the state with the added correction S(x+q). Inputs to **Collector** are the inital state vector and the position vector of the ground site. The **Collector** subfunction can be found towards the end of this code.

Marquardt's method for least squares:

$$(J^T J + \lambda D^2)\vec{q} = -J^T f \quad (1)$$

where

- J : The Jacobian Matrix where $J_{ij} = \frac{\partial f_i\left(\vec{X}\right)}{\partial y_j}$,
  $i = 1, 2, \ldots,$ number of observations, $j = 1, 2, \ldots,$ number of states.
- D : A diagonal matrix containing positive diagonal elements (Nash). For this program, $D^2 = \text{diag}(\text{diag}(A))$
- λ : The Marquardt parameter to determine magnitude and direction of step
- f : Vector of residuals
- $\vec{q}$: Change to state.

Because we desire to fit our measurements to the estimated orbit, **Eq. 1** is solved for $\vec{q}$

$$\vec{q} = (J^T J + \lambda D^2)^{-1}(-J^T f).$$

At each iteration, the sum of squares for the previous step S(**x**) and the current step S(**x**+**q**) are compared. If

$$S(\boldsymbol{x} + \boldsymbol{q}) > S(\boldsymbol{x})$$

the Marquardt parameter λ is increased by a factor of 10 and the previous corrections are reversed. If

$$S(\boldsymbol{x} + \boldsymbol{q}) < S(x)$$

λ is decreased by a factor of 0.4 for the next iteration (Nash, 1990).

```
Y0 = marquardt('get_obs',Y0,Obs,step,options,c)
```

Printout of Summary:

This section will print out the results of the program. The predicted vector was the initial orbit constructed from Gpredict data. The estimated state was the vector used to simulate data. The correction column shows corrections made to the estimated state vector. The last column of this data gives the "best estimate" of the orbiting body's initial state vector. Finally, the Root Mean Square (RMS) error is shown.

```
fprintf('\nSummary: \n');
fprintf('  truth     estimated   correction      final    \n');
for i=1:3
    fprintf('%s',Label(i),'[km]');
    fprintf('%11.1f %11.1f %14.1f %11.1f',Y0_ref(i)/1e3, Y0_apr(i)/1e3
...
        ,(Y0(i)-Y0_apr(i))/1e3, Y0(i)/1e3);
    fprintf('\n');
end
for i=4:6
    fprintf('v%s',Label(i-3),'[km/s]');
    fprintf('%11.4f %11.4f %14.4f %11.4f',Y0_ref(i)/1e3, Y0_apr(i)/1e3,
...
```

```
            (Y0(i)-Y0_apr(i)),Y0(i)/1e3);
    fprintf('\n');
end

toc
```

ORBIT AND DATA GENERATION PROGRAM

```
function [Data, rr] = get_obs(Y0,t,sat,print)

    global AuxParam

    We  = 72.9217e-6;           % angular vel. of Earth in
ECI..[rad/s]
    deg = pi/180;
    Ob  = 1/298.26;                         % Earth's flattening
factor
    Re  = physconst('EarthRadius')/1000;  % Radius of the
earth.....[km]

% Tolerances for orbit propagation
    options = rdpset('RelTol',1e-13,'AbsTol',1e-16);

% Build components of the site position vector:
    sat.fac1  = Re/sqrt(1 - (2*Ob - Ob*Ob)*sin(sat.phi)^2);
    sat.fac2  = ((Re*(1 - Ob)^2)/sqrt(1 - (2*Ob -
Ob*Ob)*sin(sat.phi)^2) + sat.alt)*sin(sat.phi);

         MJD_UTC = sat.MJD0 + t/86400;  % Modified Julian Date

        % Time increment and propagation
        AuxParam.Mjd_UTC = MJD_UTC;
        [~,yout] = radau(@Accel,[0 t],Y0,options); % State vector
        Y = yout(end,:)';

        % Get local sidereal time. Pass 1 data used but t
compensates.
        lst = LST(sat.year1, sat.month1, sat.day1, sat.UT+t/3600,
sat.EL);

        lst = lst*deg; % [rad]

        % Coordinate Transformation Matrix
        Q = geocentric2topocentric(sat.phi, lst);

        % Ground site position vector [m](rotating)[Eq. (5.56)]:
        Rs(1) = (sat.fac1 + sat.alt)*cos(sat.phi)*cos(lst);
        Rs(2) = (sat.fac1 + sat.alt)*cos(sat.phi)*sin(lst);
        Rs(3) = sat.fac2;

        Rs = Rs*1000;

        r = Y(1:3);
```

```
        v = Y(4:6);

        Vs    = cross([0 0 We],Rs);      % Derivative of Rs

        s = Q*(r-Rs');
        s_dt = Q*(v - Vs');

        %Observations
        [Az, El] = AzEl(s);   % [rad]
        range_rate = dot(s,s_dt)/norm(s);

        rr = range_rate;
        % Observation record
        MJD = MJD_UTC;
        if sat.choice == 1
%             Data = range_rate;
            Data = range_rate + sat.sig2;
        elseif sat.choice == 2
%             Data = [Az  El  range_rate];
% [rad]
             Data = [Az+sat.sig1*randn  El+sat.sig1*randn
range_rate+sat.sig2*randn];

        elseif sat.choice == 3
            Data = [Az+sat.sig1*randn  El+sat.sig1*randn] ;   %
[rad]
%             Data = [Az  El];                                  %
[rad]
        end

         % for printing of results
         if print == 1
           [year,mon,day,hr,minute,second] =
invjday(MJD_UTC+2400000.5);
           if sat.choice == 1
               fprintf('  %4d/%2.2d/%2.2d  %2.2d:%2.2d:%6.3f
%10.3f\n', ...
               year,mon,day,hr,minute,second, Data/1000);
           elseif sat.choice == 2
               fprintf('  %4d/%2.2d/%2.2d  %2.2d:%2.2d:%6.3f
%10.3f%10.3f %10.3f\n', ...
               year,mon,day,hr,minute,second,
Data(1)*180/pi,Data(2)*180/pi,Data(3)/1000);
           elseif sat.choice == 3
               fprintf('  %4d/%2.2d/%2.2d  %2.2d:%2.2d:%6.3f
%10.3f%10.3f \n', ...
               year,mon,day,hr,minute,second,
```

```
Data(1)*180/pi,Data(2)*180/pi);
            end
          end

      end % for i = 1:n_obs
```

LEVENBERG-MARQUARDT FILTER

```
function Y0 = marquardt(func,Y0,Obs,step,options,c)
%  func = 'get_obs'

global iterat

lx        = numel(Y0);         % number of states
[nPnt,nP] = size(Obs);         % number of observations or
observations sets
nParam    = nP;                % number of observables per set
Y0_prev   = zeros(lx,1);       % previous parameter set
data_prev = zeros(nPnt*nParam,1);    % previous data set
SSx       = 1e-3/eps;          % initialize sum of squares
SSx_prev  = 1e-3/eps;          % initialize previous sum of squares
J         = zeros(nPnt,lx);    % initialize the Jacobian matrix
Dof       = nPnt - lx + 1;     % statistical degrees of freedom
stop      = 0;                 % initialize termination flag
iterat    = 0;                 % initialize iteration count


bdx     = options.bdx;        % Jacobian perturbation
lambda  = options.lambda ;    % Starting Marquardt parameter
YLow    = options.bnds(1:6)';
YUp     = options.bnds(7:12)';
wts     = options.wts;


for i = 1:nPnt
    t(i) = i*step;
end

 % options are stored in struct



idx   = find(bdx ~= 0);                % indices of the parameters
to be fit
Nfit = length(idx);                    % number of parameters to
fit

if length(bdx) == 1
    bdx = bdx*ones(lx,1);     % perturbation for Jacobian
calculation
end
```

```
 for i = 1:length(t)
    data_init(i,:) = feval(func,Y0,t(i),c,0);     %  Evaluate model
using Y0
 end

% data_init = [];
% for i = 1:length(t)
% y_init = feval(func,Y0,Obs(i,1),t(i),c);     % initialize residual
vector from estimated state
% data_init = [data_init;y_init];
% end

if (var(wts) == 0)
    weight = abs(wts)*ones(nPnt*nParam,1);
    disp('Uniform weights used in analysis')
else
    weight = abs(wts(:));
end

% Initialize Jacobian
[A,g,SSx,dataVec,J] =
get_Ag(func,t,Y0_prev,data_prev,1,J,Y0,Obs,weight,bdx,c);

% Added for Chol Decomp
% D = diag(A);
% d = diag(A);
% UA = triu(A,1);
% A = UA' + UA + diag(d+1*D);
% [U,p] = chol(A);

% dq = U\(U'\g);

% End of Added chol stuff (dq comes out same as q - don't use)


if ( max(abs(g)) < options.eps1)
    fprintf(' Initial Estimate is Close to Convergence')
    stop = 1;
end

SSx_prev = SSx;

history = ones(options.maxIter,lx+3);          % Initialize
convergence history

Obs_vec = [];
```

```
   for i = 1:nPnt
       for k = 1:nParam
           Obs_vec = [Obs_vec;Obs(i,k)];
       end
   end

% Start least squares
while ( ~stop && iterat <= options.maxIter)
   iterat = iterat + 1;
   fprintf('iteratation = %f\n',iterat)

   q = (A + lambda*diag(diag(A))+1) \ g;    % Marquardt correction
to state
%   q = inv(A)*g
%    q=-q
   % check the effect of q

   Ytry = Y0 + q(idx);                      % Update change fitting
parameters
   Ytry = min(max(YLow,Ytry),YUp);          % apply constraints


%    y_try = [];      % data calculation from Ytry
%    for i = 1:nPnt
%        data_try = feval(func,Ytry,Obs(i,1),t(i),c);
%        y_try = [y_try; data_try'];
%    end

   for i = 1:length(t)
    data_try(i,:) = feval(func,Ytry,t(i),c,0);  % Evaluate model
using Y0
%     y_try(3*i-2:3*i) = data_try(i,1:3)';
   end

   y_try = [];
 for i = 1:nPnt
     for j = 1:nP
         y_try = [y_try;data_try(i,j)];   % vectorized data set
from
     end                                  % data_try matrix
 end


%  if nP == 1
%    y_try = data_try(:);
%  else
%      y_try=y_try(:);
```

```
%  end
%  y_try = y_try';

   f_res = Obs_vec - y_try;     % residual error using Ytry

   if ~all(isfinite(f_res))
       stop = 1;
       break
   end


   SSx_try = f_res' * ( f_res.*weight);  % sum of squares error
criteria

    %beta = SSx - SSx_try
      beta = (SSx - SSx_try) / ( q' * (lambda * q + g) );

   if beta > 0%options.eps4         % Ytry is accepted

       dSSx = SSx - SSx_prev;
       SSx_prev = SSx;
       Y0_prev = Y0;
       data_prev = dataVec;
       Y0 = Ytry(:);


       [A,g,SSx,dataVec,J] =
get_Ag(func,t,Y0_prev,data_prev,dSSx,J,Y0,Obs,weight,bdx,c);

       % Decrease lambda ==> Gauss-Newton Method
       lambda = max(lambda*options.decr,1e-7);


   else                            % Ytry not accepted

       SSx = SSx_prev;

       if (~rem(iterat,lx))        % rank-1 Jacobian update
           [A,g,dSSx,dataVec,J] = get_Ag(func,t,Y0_prev,data_prev,-
1,J,Y0,Obs,weight,bdx,c);
       end

       %  increase lambda ==> gradient descent method
       lambda = min(lambda*options.incr,1.e7);

   end
```

```
   if (max(abs(g)) < options.eps1 && iterat > 2 )
       fprintf('eps1');
       stop = 1;
   end
   if (max(abs(q)./(abs(Y0)+1e-6)) < options.eps2 && iterat > 2)
          fprintf('eps2');
       stop = 1;
   end
   if (SSx/Dof < options.eps3 && iterat > 2)
          fprintf('eps3');
       stop = 1;
   end
   if ( iterat == options.maxIter )
       stop = 1;
   end

end  % main loop


 function J = Jacobian(func,t,Y0,Obs,data,bdx,c)
    % Function to calculate the Jacobian matrix
    [m,w] = size(Obs);
       n  = length(Y0);

    % Jacobian
    J = [];                                % Initialize Jacobian
    for i = 1:m

       yDat = [];                          % initialize/reset vector of
data from perturbed run
       for jj = 1:w
           yDat = [yDat;data(i,jj)'];  % (3x1) vector of data at
time t
       end

       for k = 1:n
            dx = bdx(k);                                % basic step
dx
            xd = Y0;                                    % save state
            xd(k) = xd(k)+dx;                           % perturb
state k

            Jdat = feval(func,xd,t(i),c,0)';  % calculate data at t
from perturbed state vector
            Jj(1:w,k) = (Jdat-yDat)/dx;             % collect
Jacobian elements at time t
```

```
       end
       J = [J;Jj];                                        % collect
Jacobian elements for ti to tf
    end

 end


 function J = Broyden_J(Y0_prev,data_prev,J,Y0,data)

 q = Y0 - Y0_prev;

 J = J + (data - data_prev - J*q)*q' / (q'*q);    % Broyden rank-1
update

 end

function [ A,g,SSx,dataVec,J] =
get_Ag(func,t,Y0_prev,data_prev,dSSx,J,Y0,Obs,weight,bdx,c)


 [m,n] = size(Obs);                % get dimensions of observations
matrix
 lx    = length(Y0);               % number of elements in state
vector

 yObs = [];
 for i = 1:m
     for j = 1:n
         yObs = [yObs;Obs(i,j)];   % vectorized data set from obs
matrix
     end
 end



 for i = 1:length(t)
    data(i,:) = feval(func,Y0,t(i),c,0);     %  Evaluate model using
Y0
    dataVec(n*i-n+1:n*i) = data(i,:);
 end

 dataVec = dataVec(:);


 if (~rem(iterat,lx) || dSSx > 0 )
     J = Jacobian(func,t,Y0,Obs,data,bdx,c);        % finite-
```

```
difference
 else
     J = Broyden_J(Y0_prev,data_prev,J,Y0,dataVec);  % rank-1
update
 end

f_r = yObs - dataVec;             % vector of residuals

SSx = f_r'* ( f_r.*weight );   % SSx error criteria

A = J'* ( J .* ( weight * ones(1,lx) ) );

g = J' * ( weight .* f_r );


end %get_Ag

%%

end % marquardt
```