Table of Contents

Admin 1
Housekeeping 1
Constants 1
Part 1 2
1c) Nonlinear Dynamics 3
1c) Linearized DT 6
perturbations plot 6
Lin MEasurements 7
Plotting section for report 10
Provided Ydata 11
Part II - LKF Tunning 13
TMT Plots and Errors 15
Part II - LKF Implementation 19
Part II - EKF Tunning 20
EKF Implementation 26
A Solver 27
C Solver 28
EKF StatOD 28
LKF StatOD 31
StatOD NonLinear Measurements 34
Stat OD NonLinear Noise ODE 34
StatOD NonLinear ODE 35
Vary Color 35

Admin

```
%{

Names: Corey LePine and William Watkins
Professor: McMahon
Class: ASEN 5044 Stat Est for Dyn Sys
Date: December 14, 2021
Final Project: Statistical Orbit Determination

%}
```

Housekeeping

```
clc; clear all; close all;
```

Constants

```
load('orbitdeterm_finalproj_KFdata.mat')
u = 398600; % Earth's standard gravitational paremters [km^3/s^2]
r0 = 6678; % Nominal orbit radius [km]
Re = 6378; % Uniform radius of Earth [km]
```

```
we = 2*pi/86400; % Consant rotation rate of Earth [rad/s]
vel = sqrt(u/r0); % orbital velocity
circ = 2*pi*r0; % circumference of the orbit
T = circ / vel; % Orbital period

n = 4; % number of states
m = 2; % number of inputs, also happens to be number of distrubances
p = 3; % number of measurements
j = 6; % number of measurement stations

dt = 10; % step size [s]
tspan = 0:dt:14000;
initCon = [r0, 0, 0, r0*sqrt(u/r0^3)]; % initial state point - LTV
 sys, so
% have to linearize about a nominal trajectory, see below

DEBUG = 0; % Debug boolean

ColorSet = varycolor(12); % 12 Unique Colors for the Tracking Stations
```

Part 1

```
% 1a) Find the CT Jacobian Matracies
syms x1 x2 x3 x4 mu u1 u2 w1 w2 z1 z2 z3 z4 t % zi is the states of
 the ground stations

f = [x2;
    -mu*x1/sqrt(x1^2 + x3^2)^3 + u1 + w1;
    x4;
    -mu*x3/sqrt(x1^2 + x3^2)^3 + u2 + w2];
state = [x1, x2, x3, x4];
inputs = [u1, u2];
disturb = [w1, w2];

A = jacobian(f, state);
B = jacobian(f, inputs);
Gam = jacobian(f, disturb);

h = [sqrt((x1 - z1)^2 + (x3 - z3)^2);
    ((x1 - z1)*(x2 - z2) + (x3 - z3)*(x4 - z4))/(sqrt((x1 - z1)^2 +
 (x3 - z3)^2));
    atan((x3 - z3)/(x1 - z1))];

C = jacobian(h, state);
D = jacobian(h, inputs);

% 1b) Linearize about Nominal operating points
for ii = 1:length(tspan)
    nomCon(:,ii) = [r0 * cos(sqrt(u/r0^3)*tspan(ii)); -r0 *
 sin(sqrt(u/r0^3)*tspan(ii))*sqrt(u/r0^3);
            r0*sin(sqrt(u/r0^3)*tspan(ii)); r0*cos(sqrt(u/
r0^3)*tspan(ii))*sqrt(u/r0^3)];
            % Have to linearize about nominal trajectory!
```

```
end
mu = u;
```

1c) Nonlinear Dynamics

State NL

```
Rel_Tol = 1e-13;
Abs_Tol = Rel_Tol;
options = odeset('Stats', 'off', 'RelTol', Rel_Tol, 'AbsTol',
 Abs_Tol);

perts = [0, 0.075, 0, -0.021];
Initial_States = perts + initCon;

[Time_out, State_out] = ode45(@(Time, State) StatODNL_ODE(Time,
 State), tspan, Initial_States, options);

State_X = State_out(:, 1);
State_Xdot = State_out(:, 2);
State_Y = State_out(:, 3);
State_Ydot = State_out(:, 4);
% Perturbations
State_X_Pert = State_out(:,1) - nomCon(1,:)';
State_Xdot_Pert = State_out(:,2) - nomCon(2,:)';
State_Y_Pert = State_out(:,3) - nomCon(3,:)';
State_Ydot_Pert = State_out(:,4) - nomCon(4,:)';

figure()
subplot(4, 1, 1)
plot(Time_out, State_out(:, 1), 'k')
% xlabel('Time [s]')
ylabel('X [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(Time_out, State_out(:, 2), 'k')
% xlabel('Time [s]')
ylabel('$\dot{X}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(Time_out, State_out(:, 3), 'k')
% xlabel('Time [s]')
ylabel('Y [km]')
ylim([-1e4, 1e4])
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(Time_out, State_out(:, 4), 'k')
xlabel('Time [s]')
ylabel('$\dot{Y}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)
```

```
sgtitle('States vs Time, Full Nonlinear Dynamics Simulation')

% Tracking Stations Positions
TS_IDS = 1:1:12; % tracking stations ids
theta_TS0 = (TS_IDS - 1)*pi/6; % tracking stations intial positions

for ii = 1:12 % tracking station X and Y position
    TS_X(:, ii) = Re*cos(we*Time_out + theta_TS0(ii));
    TS_Xdot(:, ii) = -Re*we*sin(we*Time_out + theta_TS0(ii));
    TS_Y(:, ii) = Re*sin(we*Time_out + theta_TS0(ii));
    TS_Ydot(:, ii) = Re*we*cos(we*Time_out + theta_TS0(ii));
    theta_TS(:, ii) = atan2(TS_Y(:, ii), TS_X(:, ii));
end

thetaCompare = theta_TS;

% Measurement NL
for ii = 1:12
    rho(:, ii) = sqrt((State_X - TS_X(:, ii)).^2 + (State_Y - TS_Y(:,
 ii)).^2);
    rho_dot(:, ii) = ((State_X - TS_X(:, ii)).*(State_Xdot -
 TS_Xdot(:, ii)) + (State_Y - TS_Y(:, ii)).*(State_Ydot - TS_Ydot(:,
 ii)))./rho(:, ii);
    phi(:, ii) = atan2((State_Y - TS_Y(:, ii)), (State_X - TS_X(:,
 ii)));
    visibleStation(:,ii) = ones(length(tspan),1) * ii;
end

phiCompare = phi;

% Wrap the upper and lower bounds between -pi and pi
% When the upper bound is above pi, need to wrap both bound down to -
pi
% Vice versa when lower bound is
thetaBound1Pos = theta_TS;
thetaBound1PosInd = find(thetaBound1Pos+pi/2 > pi);
thetaBound1Pos(thetaBound1PosInd) = thetaBound1Pos(thetaBound1PosInd)
 - 2*pi;
thetaBound1Neg = theta_TS;
thetaBound1Neg(thetaBound1PosInd) = thetaBound1Neg(thetaBound1PosInd)
 - 2*pi;

thetaBound2Neg = theta_TS;
thetaBound2NegInd = find(thetaBound2Neg-pi/2 < pi);
thetaBound2Neg(thetaBound2NegInd) = thetaBound2Neg(thetaBound2NegInd)
 + 2*pi;
thetaBound2Pos = theta_TS;
thetaBound2Pos(thetaBound2NegInd) = thetaBound2Pos(thetaBound2NegInd)
 + 2*pi;

% Visible Tracking Stations
figure('Position', [200, 200, 1200, 1000])
hold on
```

```
tl = tiledlayout(4,1);
title(tl, "Nonlinear Measurements");
xlabel(tl, "Time [s]");
nexttile
hold on
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    if(DEBUG == 1)
        plot(Time_out, pi/2 + thetaBound1Pos(:,ii));
        hold on
        plot(Time_out, -pi/2 + thetaBound1Neg(:,ii));
        plot(Time_out, phi(:,ii));
        plot(Time_out, theta_TS(:,ii));
        scatter(Time_out(vis_index), phi(vis_index, ii))
        yline(pi);
        yline(-pi);
    end
    scatter(Time_out(vis_index), rho(vis_index,ii), [],
 ColorSet(ii, :));
    ylabel('\rho^i [km]');
    set(gca, 'FontSize', 14)
end

nexttile
hold on
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    scatter(Time_out(vis_index), rho_dot(vis_index,ii), [],
 ColorSet(ii, :));
    ylabel('$\dot{rho}^i$ [km/s]', 'Interpreter','latex');
    set(gca, 'FontSize', 14)
end
nexttile
hold on
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    scatter(Time_out(vis_index), phi(vis_index,ii), [],
 ColorSet(ii, :));
    ylabel('\phi^i [rads]');
    set(gca, 'FontSize', 14)
end
nexttile
hold on
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    scatter(Time_out(vis_index), visibleStation(vis_index,ii), [],
 ColorSet(ii, :));
    ylabel('Visible Station ID');
    set(gca, 'FontSize', 14)
end
```

1c) Linearized DT

```
pertX(:, 1) = perts';
for ii = 2:1401
    x1 = nomCon(1,ii-1);
    x2 = nomCon(2,ii-1);
    x3 = nomCon(3,ii-1);
    x4 = nomCon(4,ii-1);
    Atil = Atil_Solver([x1, x2, x3, x4]);

    Ftil = eye(n) + dt*Atil;
    pertX(:, ii) = Ftil*pertX(:, ii-1);
end

LinX(1, :) = nomCon(1,:) + pertX(1, :);
LinX(2, :) = nomCon(2,:) + pertX(2, :);
LinX(3, :) = nomCon(3,:) + pertX(3, :);
LinX(4, :) = nomCon(4,:) + pertX(4, :);
```

perturbations plot

```
figure()
subplot(4, 1, 1)
plot(Time_out, pertX(1, :), 'k')
% xlabel('Time [s]')
ylabel('\delta X [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(Time_out, pertX(2, :), 'k')
% xlabel('Time [s]')
ylabel('$\dot{\delta X}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
```

```
plot(Time_out, pertX(3, :), 'k')
% xlabel('Time [s]')
ylabel('\delta Y [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(Time_out, pertX(4, :), 'k')
xlabel('Time [s]')
ylabel('$\dot{\delta Y}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

sgtitle('Linearized Approx Perturbations vs Time')

% state plot
figure()
subplot(4, 1, 1)
plot(Time_out, LinX(1, :), 'k')
% xlabel('Time [s]')
ylabel('X [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(Time_out, LinX(2, :), 'k')
% xlabel('Time [s]')
ylabel('$\dot{X}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(Time_out, LinX(3, :), 'k')
% xlabel('Time [s]')
ylabel('Y [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(Time_out, LinX(4, :), 'k')
xlabel('Time [s]')
ylabel('$\dot{Y}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

sgtitle('States vs Time, Linearized Approximate Dynamics Soluiton')
```

Lin MEasurements

```
for j = 1:12
    for i = 1:1401
        x(1) = nomCon(1,i);
        x(2) = nomCon(2,i);
        x(3) = nomCon(3,i);
        x(4) = nomCon(4,i);
        z(1) = TS_X(i,j);
        z(2) = TS_Xdot(i,j);
        z(3) = TS_Y(i,j);
        z(4) = TS_Ydot(i,j);
        Cnom = Ctil_Solver(x,z);

        H = Cnom;
        pertY(:, i) = H*pertX(:, i);
    end

    rhoLinPert(:,j) = pertY(1,:)';
    rho_dotLinPert(:,j) = pertY(2,:)';
    phiLinPert(:,j) = pertY(3,:)';

    rhoNom(:, j) = sqrt((nomCon(1,:)' - TS_X(:, j)).^2 + (nomCon(3,:)'
 - TS_Y(:, j)).^2);
    rho_dotNom(:, j) = ((nomCon(1,:)' - TS_X(:, j)).*(nomCon(2,:)'
 - TS_Xdot(:, j)) + (nomCon(3,:)' - TS_Y(:, j)).*(nomCon(4,:)' -
 TS_Ydot(:, j)))./rhoNom(:, j);
    phiNom(:, j) = atan2((nomCon(3,:)' - TS_Y(:, j)), (nomCon(1,:)' -
 TS_X(:, j)));

    rhoLinNom(:,j) = rhoNom(:,j) + rhoLinPert(:,j);
    rhoDotLinNom(:,j) = rho_dotNom(:,j) + rho_dotLinPert(:,j);
    phiLinNom(:,j) = phiNom(:,j) + phiLinPert(:,j);

    findAbovePiRho = find(rhoLinNom(:,j) > pi);
    findAbovePiPhi = find(phiLinNom(:,j) > pi);
    findBelowPiRho = find(rhoLinNom(:,j) < -pi);
    findBelowPiPhi = find(phiLinNom(:,j) < -pi);

%     rhoLinNom(findAbovePiRho,j) = rhoLinNom(findAbovePiRho,j) -
 2*pi;
    phiLinNom(findAbovePiPhi,j) = phiLinNom(findAbovePiPhi,j) - 2*pi;
%     rhoLinNom(findBelowPiRho,j) = rhoLinNom(findBelowPiRho,j) +
 2*pi;
    phiLinNom(findBelowPiPhi,j) = phiLinNom(findBelowPiPhi,j) + 2*pi;
end


figure('Position', [200, 200, 1200, 1000])
hold on
tl = tiledlayout(4,1);
title(tl, "Linearized Measurements");
xlabel(tl, "Time [s]");
nexttile
hold on
ylim([0 2500]);
DEBUG = 0;
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    if(DEBUG == 1)
%         plot(Time_out, pi/2 + thetaBound1Pos(:,ii));
% %         hold on
%         plot(Time_out, -pi/2 + thetaBound1Neg(:,ii));
        plot(Time_out, phiLin(:,ii));
        plot(Time_out, theta_TS(:,ii));
        scatter(Time_out(vis_index), phiLin(vis_index, ii))
        yline(pi);
        yline(-pi);
    end
     scatter(Time_out(vis_index), rhoLinNom(vis_index,ii));
    ylabel('\rho^i [km]');
end

nexttile
hold on
for ii = 1:12
%     vis_index = find((phiLin(:, ii) <= (pi/2 + thetaCompare(:, ii))
 & phiLin(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
%             (phiLin(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiLin(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
%             (phiLin(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiLin(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    scatter(Time_out(vis_index), rhoDotLinNom(vis_index,ii));
    ylabel('$\dot{\rho}^i$ [km/s]', 'Interpreter','latex');
end
nexttile
hold on
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    scatter(Time_out(vis_index), phiLinNom(vis_index,ii));
    ylabel('\phi^i [rads]');
end
nexttile
hold on
for ii = 1:12
    vis_index = find((phiCompare(:, ii) <= (pi/2 + thetaCompare(:,
 ii)) & phiCompare(:, ii) >= (-pi/2 + thetaCompare(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound1Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound1Neg(:, ii))) | ...
            (phiCompare(:, ii) <= (pi/2 + thetaBound2Pos(:, ii)) &
 phiCompare(:, ii) >= (-pi/2 + thetaBound2Neg(:, ii))));
    scatter(Time_out(vis_index), visibleStation(vis_index,ii));
    ylabel('Visible Station ID');
end
```

Plotting section for report

We will plot the states on top of one another, as well as the measurements

```
figure('Position', [200, 200, 1600, 1000])
t1 = tiledlayout(4,1);
title(t1, "Simulated System State Perturbations");
xlabel(t1, "Time [s]");

nexttile;
hold on;
plot(Time_out, State_X_Pert, 'k')
plot(Time_out, pertX(1,:), 'r');
ylabel('$\delta X$ [km]', 'Interpreter','latex')
set(gca, 'FontSize', 14)
legend('Nonlinear Perturbations', 'Linearized
 Perturbations','location','bestoutside');

nexttile;
hold on;
plot(Time_out, State_Xdot_Pert, 'k')
plot(Time_out, pertX(2,:), 'r');
ylabel('$\delta \dot{X}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

nexttile;
hold on;
plot(Time_out, State_Y_Pert, 'k')
plot(Time_out, pertX(3,:), 'r');
ylabel('$\delta Y$ [km]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

ax1 = nexttile;
hold on;
plot(Time_out, State_Ydot_Pert, 'k')
plot(Time_out, pertX(4,:), 'r');
ylabel('$\delta \dot{Y}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

% System States next
figure('Position', [200, 200, 1600, 1000])
t1 = tiledlayout(4,1);
title(t1, "Simulated System Dynamics");
xlabel(t1, "Time [s]");

nexttile;
hold on;
plot(Time_out, State_X, 'k')
plot(Time_out, LinX(1,:), 'r');
ylabel('$X$ [km]', 'Interpreter','latex')
set(gca, 'FontSize', 14)
legend('Nonlinear Dynamics', 'Linearized
 Dynamics','location','bestoutside');
```

```
nexttile;
hold on;
plot(Time_out, State_Xdot, 'k')
plot(Time_out, LinX(2,:), 'r');
ylabel('$\dot{X}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

nexttile;
hold on;
plot(Time_out, State_Y, 'k')
plot(Time_out, LinX(3,:), 'r');
ylabel('$Y$ [km]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

ax1 = nexttile;
hold on;
plot(Time_out, State_Ydot, 'k')
plot(Time_out, LinX(4,:), 'r');
ylabel('$\dot{Y}$ [km/s]', 'Interpreter','latex')
set(gca, 'FontSize', 14)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
```

Provided Ydata

```
Gam = [0, 0; 1, 0; 0 0; 0 1];
Omega = dt*Gam;

ydata_TS_ID = NaN*ones(2, 1401);
ydata_data = NaN*ones(2*p, 1401);
c = NaN*ones(1, 1401);

for ii = 1:1401
    if ~isempty(ydata{ii})
        [~, c(ii)] = size(ydata{ii});
        if c(ii) == 1
            ydata_TS_ID(1, ii) = ydata{ii}(4, 1);
            ydata_data(1:3, ii) = ydata{ii}(1:3, 1);
        elseif c(ii) == 2
            ydata_TS_ID(1, ii) = ydata{ii}(4, 1);
            ydata_data(1:3, ii) = ydata{ii}(1:3, 1);
            ydata_TS_ID(2, ii) = ydata{ii}(4, 2);
            ydata_data(4:6, ii) = ydata{ii}(1:3, 2);
        end
    else
        ydata_TS_ID(:, ii) = NaN*ones(2, 1);
        ydata_data(:, ii) = NaN*ones(6, 1);
    end

end
```

```
ydata_TS_ID(1, 1) = 1;

TS_state(:, :, 1) = TS_X;
TS_state(:, :, 2) = TS_Xdot;
TS_state(:, :, 3) = TS_Y;
TS_state(:, :, 4) = TS_Ydot;

figure()
subplot(4, 1, 1)
hold on
for ii = 1:12
    [~, ID_index_c] = find(ydata_TS_ID(1, :) == ii);
    scatter(tvec(ID_index_c), ydata_data(1, ID_index_c), [],
 ColorSet(ii, :))
    [~, ID_index_c] = find(ydata_TS_ID(2, :) == ii);
    scatter(tvec(ID_index_c), ydata_data(4, ID_index_c), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('\rho^i [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
hold on
for ii = 1:12
    [~, ID_index_c] = find(ydata_TS_ID(1, :) == ii);
    scatter(tvec(ID_index_c), ydata_data(2, ID_index_c), [],
 ColorSet(ii, :))
    [~, ID_index_c] = find(ydata_TS_ID(2, :) == ii);
    scatter(tvec(ID_index_c), ydata_data(5, ID_index_c), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('\rhodot^i [km/s]')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
hold on
for ii = 1:12
    [~, ID_index_c] = find(ydata_TS_ID(1, :) == ii);
    scatter(tvec(ID_index_c), ydata_data(3, ID_index_c), [],
 ColorSet(ii, :))
    [~, ID_index_c] = find(ydata_TS_ID(2, :) == ii);
    scatter(tvec(ID_index_c), ydata_data(6, ID_index_c), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('\phi^i [rad]')
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
```

```
hold on
for ii = 1:12
    [~, ID_index_c] = find(ydata_TS_ID(1, :) == ii);
    scatter(tvec(ID_index_c), ydata_TS_ID(1, ID_index_c), [],
 ColorSet(ii, :))
    [~, ID_index_c] = find(ydata_TS_ID(2, :) == ii);
    scatter(tvec(ID_index_c), ydata_TS_ID(2, ID_index_c), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('Visible Station ID')
set(gca, 'FontSize', 14)

sgtitle('Provided ydata')
```

Part II - LKF Tunning

```
N = 50;

dx_true = [0, 0.075, 0, -0.021];
Px_true = 1e4*diag([0.1, 0.01, 0.1, 0.01]);
qp = 1e-6;

for jj = 1:N
    % TMT
    perts = mvnrnd(dx_true, qp*Px_true);

    MC_Initial_State = perts + initCon;

    TMT_X(1) = MC_Initial_State(1);
    TMT_Xdot(1) = MC_Initial_State(2);
    TMT_Y(1) = MC_Initial_State(3);
    TMT_Ydot(1) = MC_Initial_State(4);

    v = mvnrnd([0, 0, 0], Rtrue)';
    TMT_y_NL_out(:, 1) = StatOD_NLMeasurement([TMT_X(1), TMT_Xdot(1),
 TMT_Y(1), TMT_Ydot(1)], [TS_X(1, 1), TS_Xdot(1, 1), TS_Y(1, 1),
 TS_Ydot(1, 1)]);
    TMT_y_NL_noise_out(:, 1) = TMT_y_NL_out(:, 1) + v;
    TMT_ydata(1) = {[TMT_y_NL_noise_out(:, 1); ydata_TS_ID(1)]};
    TS_ID = NaN*ones(1401, 1);
    TS_ID(1) = ydata_TS_ID(1);


    for ii = 1:1400
        % Noise
        w = mvnrnd([0, 0], Qtrue);
        v = mvnrnd([0, 0, 0], Rtrue)';

        % State
        ODE45_InitialState = [TMT_X(ii), TMT_Xdot(ii), TMT_Y(ii),
 TMT_Ydot(ii), w(1), w(2)];
```

```
        [~, TMT_test] = ode45(@(Time, State) StatODNL_noise_ODE(Time,
State), [tvec(ii) tvec(ii+1)], ODE45_InitialState, options);

        TMT_X(ii+1) = TMT_test(end, 1);
        TMT_Xdot(ii+1) = TMT_test(end, 2);
        TMT_Y(ii+1) = TMT_test(end, 3);
        TMT_Ydot(ii+1) = TMT_test(end, 4);

        % Measurment
        for kk = 1:12 % compute measurements for each ground station
            yi = StatOD_NLMeasurement([TMT_X(ii+1), TMT_Xdot(ii+1),
TMT_Y(ii+1), TMT_Ydot(ii+1)], [TS_X(ii+1, kk), TS_Xdot(ii+1, kk),
TS_Y(ii+1, kk), TS_Ydot(ii+1, kk)]);
            TMT_y_ALL(ii, kk, 1) = yi(1) + v(1); % rho
            TMT_y_ALL(ii, kk, 2) = yi(2) + v(2); % rhodot
            TMT_y_ALL(ii, kk, 3) = yi(3) + v(3); % phi
        end
    end
    phiCompare = TMT_y_ALL(:, :, 3);

    for kk = 1:12 % compute the current visible ground station
        vis_index = find((phiCompare(:, kk) <= (pi/2 +
thetaCompare(2:end, kk)) & phiCompare(:, kk) >= (-pi/2 +
thetaCompare(2:end, kk))) | ...
            (phiCompare(:, kk) <= (pi/2 + thetaBound1Pos(2:end, kk)) &
phiCompare(:, kk) >= (-pi/2 + thetaBound1Neg(2:end, kk))) | ...
            (phiCompare(:, kk) <= (pi/2 + thetaBound2Pos(2:end, kk)) &
phiCompare(:, kk) >= (-pi/2 + thetaBound2Neg(2:end, kk))));

        TMT_y_NL_noise_out(1, vis_index+1) = TMT_y_ALL(vis_index, kk,
1);
        TMT_y_NL_noise_out(2, vis_index+1) = TMT_y_ALL(vis_index, kk,
2);
        TMT_y_NL_noise_out(3, vis_index+1) = TMT_y_ALL(vis_index, kk,
3);

        TS_ID(vis_index+1) = repmat(kk, length(vis_index), 1);
    end

    for ii = 2:1401
        TMT_ydata(ii) = {[TMT_y_NL_noise_out(:, ii); TS_ID(ii)]};
    end

    TMT_State = [TMT_X; TMT_Xdot; TMT_Y; TMT_Ydot];

    % NEES and NIS
    Q_LKF = 1500*Qtrue;
    R_LKF = Rtrue;

    dx0 = dx_true;
    P0 = Px_true;

    [P, dx, x_stds, eytil, S] = LKF_StatOD(dx0, P0, TMT_ydata, dt,
Q_LKF, R_LKF, Gam, TS_state, nomCon');
```

```
    dxtrue = TMT_State - nomCon;
    ex = dxtrue - dx.pos;
    for ii = 1:1401
        Ex(jj, ii) = ex(:, ii)'*(P.pos(:, :, ii))^-1*ex(:, ii);
        Ey(jj, ii) = eytil(1:3, ii)'*(S(1:3, 1:3, ii))^-1*eytil(1:3,
 ii);
    end

end
```

TMT Plots and Errors

```
x = nomCon + dx.pos;

figure()
subplot(4, 1, 1)
plot(tvec, TMT_State(1, :), 'k')
hold on
plot(tvec, nomCon(1, :), 'b')
plot(tvec, x(1, :), 'r')
hold off
xlabel('Time [s]')
ylabel('X [km]')
set(gca, 'FontSize', 14)
legend('TMT', 'Nominal', 'LKF Estimate')

subplot(4, 1, 2)
plot(tvec, TMT_State(2, :), 'k')
hold on
plot(tvec, nomCon(2, :), 'b')
plot(tvec, x(2, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Xdot [km/s]')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(tvec, TMT_State(3, :), 'k')
hold on
plot(tvec, nomCon(3, :), 'b')
plot(tvec, x(3, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Y [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(tvec, TMT_State(4, :), 'k')
hold on
plot(tvec, nomCon(4, :), 'b')
plot(tvec, x(4, :), 'r')
hold off
```

```
xlabel('Time [s]')
ylabel('Ydot [km/s]')
set(gca, 'FontSize', 14)

sgtitle('TMT Simulated States')

figure() % TMT Measurement Plots
subplot(4, 1, 1)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TMT_y_NL_noise_out(1, ID_index), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('TMT \rho^i [km]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TMT_y_NL_noise_out(2, ID_index), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('TMT \rhodot^i [km/s]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TMT_y_NL_noise_out(3, ID_index), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('TMT \phi^i [rad]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TS_ID(ID_index), [], ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
```

```
ylabel('Visible Station ID')
grid on
set(gca, 'FontSize', 14)

sgtitle('TMT Simulated Measurements vs Time')

figure() % innovations
subplot(3, 1, 1)
scatter(tvec, eytil(1, :))
xlabel('Time [s]')
ylabel('\rho Innovation')
set(gca, 'FontSize', 14)

subplot(3, 1, 2)
scatter(tvec, eytil(2, :))
xlabel('Time [s]')
ylabel('\rhodot Innovation')
set(gca, 'FontSize', 14)

subplot(3, 1, 3)
scatter(tvec, eytil(3, :))
xlabel('Time [s]')
ylabel('\phi Innovation')
set(gca, 'FontSize', 14)

sgtitle('Innovations vs Time')


figure()
subplot(4, 1, 1)
plot(tvec, ex(1, :), 'k')
hold on
plot(tvec, 2*x_stds(1, :), 'r')
plot(tvec, -2*x_stds(1, :), 'r')
hold off
xlabel('Time [s]')
ylabel('X Error [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(tvec, ex(2, :), 'k')
hold on
plot(tvec, 2*x_stds(2, :), 'r')
plot(tvec, -2*x_stds(2, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Xdot Error [km/s]')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(tvec, ex(3, :), 'k')
hold on
plot(tvec, 2*x_stds(3, :), 'r')
plot(tvec, -2*x_stds(3, :), 'r')
```

```
hold off
xlabel('Time [s]')
ylabel('Y Error [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(tvec, ex(4, :), 'k')
hold on
plot(tvec, 2*x_stds(4, :), 'r')
plot(tvec, -2*x_stds(4, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Ydot Error [km/s]')
set(gca, 'FontSize', 14)

sgtitle('States Estimation Error vs Time - LKF')

% Consitancy Plots
Ex_mean = mean(Ex);
Ey_mean = mean(Ey);

alpha = 0.05;
r1 = chi2inv(alpha/2, N*n)/N;
r2 = chi2inv(1-alpha/2, N*n)/N;

figure() % NEES
scatter(tvec, Ex_mean)
hold on
plot(tvec, repmat(r1, 1401, 1), 'r--')
plot(tvec, repmat(r2, 1401, 1), 'r--')
hold off
ylim([2 6])
xlabel('Time [s]')
ylabel('Mean \epsilon_x')
title('LKF NEES Plot')
legend('NEES @ t_k', 'r_1 Bound', 'r_2 Bound')
grid on
set(gca, 'FontSize', 14)

alpha = 0.05;
r1 = chi2inv(alpha/2, N*p)/N;
r2 = chi2inv(1-alpha/2, N*p)/N;

figure() % NIS
scatter(tvec, Ey_mean)
hold on
plot(tvec, repmat(r1, 1401, 1), 'r--')
plot(tvec, repmat(r2, 1401, 1), 'r--')
hold off
ylim([1 5])
xlabel('Time [s]')
ylabel('Mean \epsilon_y')
title('LKF NIS Plot')
legend('NIS @ t_k', 'r_1 Bound', 'r_2 Bound')
```

```
grid on
set(gca, 'FontSize', 14)
```

Part II - LKF Implementation

```
dx0 = [0, 0.075, 0, -0.021];
P0 = Px_true;
[P, dx, x_stds, eytil, S] = LKF_StatOD(dx0, P0, ydata, dt, Q_LKF,
 R_LKF, Gam, TS_state, nomCon');

x = nomCon + dx.pos;

figure()
subplot(4, 1, 1)
plot(tvec, x(1, :), 'k')
hold on
plot(tvec, x(1, :) + 2*x_stds(1, :), 'r--')
plot(tvec, x(1, :) - 2*x_stds(1, :), 'r--')
hold off
xlabel('Time [s]')
ylabel('Estimated X [km]')
legend('Estimated State', '2\sigma')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(tvec, x(2, :), 'k')
hold on
plot(tvec, x(2, :) + 2*x_stds(2, :), 'r--')
plot(tvec, x(2, :) - 2*x_stds(2, :), 'r--')
hold off
xlabel('Time [s]')
ylabel('Estimated Xdot [km/s]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(tvec, x(3, :), 'k')
hold on
plot(tvec, x(3, :) + 2*x_stds(1, :), 'r--')
plot(tvec, x(3, :) - 2*x_stds(1, :), 'r--')
hold off
xlabel('Time [s]')
ylabel('Estimated Y [km]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(tvec, x(4, :), 'k')
hold on
plot(tvec, x(4, :) + 2*x_stds(2, :), 'r--')
plot(tvec, x(4, :) - 2*x_stds(2, :), 'r--')
hold off
```

```
xlabel('Time [s]')
ylabel('Estimated Ydot [km/s]')
grid on
set(gca, 'FontSize', 14)

sgtitle('Implemented LKF Estimated States vs Time')
```

Part II - EKF Tunning

```
N = 50;

xtrue = initCon;
Ptrue = diag([0.05, 0.00025, 0.05, 0.00025]);
qp = 1e-6;

for jj = 1:N
    % TMT
    perts = mvnrnd(xtrue, Ptrue);

    MC_Initial_State = perts;

    TMT_X(1) = MC_Initial_State(1);
    TMT_Xdot(1) = MC_Initial_State(2);
    TMT_Y(1) = MC_Initial_State(3);
    TMT_Ydot(1) = MC_Initial_State(4);

    v = mvnrnd([0, 0, 0], Rtrue)';
    TMT_y_NL_out(:, 1) = StatOD_NLMeasurement([TMT_X(1), TMT_Xdot(1),
 TMT_Y(1), TMT_Ydot(1)], [TS_X(1, 1), TS_Xdot(1, 1), TS_Y(1, 1),
 TS_Ydot(1, 1)]);
    TMT_y_NL_noise_out(:, 1) = TMT_y_NL_out(:, 1) + v;
    TMT_ydata(1) = {[TMT_y_NL_noise_out(:, 1); ydata_TS_ID(1)]};
    TS_ID = NaN*ones(1401, 1);
    TS_ID(1) = ydata_TS_ID(1);


    for ii = 1:1400
        % Noise
        w = mvnrnd([0, 0], Qtrue);
        v = mvnrnd([0, 0, 0], Rtrue)';

        % State
        ODE45_InitialState = [TMT_X(ii), TMT_Xdot(ii), TMT_Y(ii),
 TMT_Ydot(ii), w(1), w(2)];
        [~, TMT_test] = ode45(@(Time, State) StatODNL_noise_ODE(Time,
 State), [tvec(ii) tvec(ii+1)], ODE45_InitialState, options);

        TMT_X(ii+1) = TMT_test(end, 1);
        TMT_Xdot(ii+1) = TMT_test(end, 2);
        TMT_Y(ii+1) = TMT_test(end, 3);
        TMT_Ydot(ii+1) = TMT_test(end, 4);

        % Measurment
        for kk = 1:12 % compute measurements for each ground station
            yi = StatOD_NLMeasurement([TMT_X(ii+1), TMT_Xdot(ii+1),
TMT_Y(ii+1), TMT_Ydot(ii+1)], [TS_X(ii+1, kk), TS_Xdot(ii+1, kk),
TS_Y(ii+1, kk), TS_Ydot(ii+1, kk)]);
            TMT_y_ALL(ii, kk, 1) = yi(1) + v(1); % rho
            TMT_y_ALL(ii, kk, 2) = yi(2) + v(2); % rhodot
            TMT_y_ALL(ii, kk, 3) = yi(3) + v(3); % phi
        end
    end
    phiCompare = TMT_y_ALL(:, :, 3);

    for kk = 1:12 % compute the current visible ground station
        vis_index = find((phiCompare(:, kk) <= (pi/2 +
thetaCompare(2:end, kk)) & phiCompare(:, kk) >= (-pi/2 +
thetaCompare(2:end, kk))) | ...
            (phiCompare(:, kk) <= (pi/2 + thetaBound1Pos(2:end, kk)) &
phiCompare(:, kk) >= (-pi/2 + thetaBound1Neg(2:end, kk))) | ...
            (phiCompare(:, kk) <= (pi/2 + thetaBound2Pos(2:end, kk)) &
phiCompare(:, kk) >= (-pi/2 + thetaBound2Neg(2:end, kk))));

        TMT_y_NL_noise_out(1, vis_index+1) = TMT_y_ALL(vis_index, kk,
1);
        TMT_y_NL_noise_out(2, vis_index+1) = TMT_y_ALL(vis_index, kk,
2);
        TMT_y_NL_noise_out(3, vis_index+1) = TMT_y_ALL(vis_index, kk,
3);

        TS_ID(vis_index+1) = repmat(kk, length(vis_index), 1);


    end

    for ii = 2:1401
        TMT_ydata(ii) = {[TMT_y_NL_noise_out(:, ii); TS_ID(ii)]};
    end

    TMT_State = [TMT_X; TMT_Xdot; TMT_Y; TMT_Ydot];

    % NEES and NIS
    Q_EKF = 0.95*Qtrue;
    R_EKF = Rtrue;

    x0 = initCon;
    P0 = 10*eye(n);

    [P, x, x_stds, eytil, S] = EKF_StatOD(x0, P0, TMT_ydata, dt, tvec,
Q_EKF, R_EKF, Gam, TS_state);

    ex = TMT_State - x.pos;
    for ii = 1:1401
        Ex(jj, ii) = ex(:, ii)'*(P.pos(:, :, ii))^-1*ex(:, ii);
        Ey(jj, ii) = eytil(1:3, ii)'*(S(1:3, 1:3, ii))^-1*eytil(1:3,
ii);
    end
```

```
end

figure()
subplot(4, 1, 1)
plot(tvec, TMT_State(1, :), 'k')
hold on
plot(tvec, nomCon(1, :), 'g')
plot(tvec, x.pos(1, :), 'r')
hold off
xlabel('Time [s]')
ylabel('X [km]')
set(gca, 'FontSize', 14)
legend('TMT', 'Nominal', 'LKF Estimate')

subplot(4, 1, 2)
plot(tvec, TMT_State(2, :), 'k')
hold on
plot(tvec, nomCon(2, :), 'g')
plot(tvec, x.pos(2, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Xdot [km/s]')
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(tvec, TMT_State(3, :), 'k')
hold on
plot(tvec, nomCon(3, :), 'g')
plot(tvec, x.pos(3, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Y [km]')
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(tvec, TMT_State(4, :), 'k')
hold on
plot(tvec, nomCon(4, :), 'g')
plot(tvec, x.pos(4, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Ydot [km/s]')
set(gca, 'FontSize', 14)

sgtitle('TMT Simulated States')

figure() % state estimation errors
subplot(4, 1, 1)
plot(tvec, ex(1, :), 'k')
hold on
plot(tvec, 2*x_stds(1, :), 'r--')
plot(tvec, -2*x_stds(1, :), 'r--')
hold off
```

```
ylim([-0.5 0.5])
xlabel('Time [s]')
ylabel('X Error [km]')
legend('State Estimation Error', '2\sigma Bounds')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(tvec, ex(2, :), 'k')
hold on
plot(tvec, 2*x_stds(2, :), 'r--')
plot(tvec, -2*x_stds(2, :), 'r--')
hold off
ylim([-0.005 0.005])
xlabel('Time [s]')
ylabel('Xdot Error [km/s]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(tvec, ex(3, :), 'k')
hold on
plot(tvec, 2*x_stds(3, :), 'r--')
plot(tvec, -2*x_stds(3, :), 'r--')
hold off
ylim([-0.6 0.6])
xlabel('Time [s]')
ylabel('Y Error [km]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(tvec, ex(4, :), 'k')
hold on
plot(tvec, 2*x_stds(4, :), 'r--')
plot(tvec, -2*x_stds(4, :), 'r--')
hold off
ylim([-0.005 0.005])
xlabel('Time [s]')
ylabel('Ydot Error [km/s]')
grid on
set(gca, 'FontSize', 14)

sgtitle('States Estimation Error vs Time - EKF')

figure() % innovations
subplot(3, 1, 1)
scatter(tvec, eytil(1, :))
xlabel('Time [s]')
ylabel('\rho^i Error [km]');
grid on
set(gca, 'FontSize', 14)
ylim([-0.5 0.5])
```

```
subplot(3, 1, 2)
scatter(tvec, eytil(2, :))
xlabel('Time [s]')
ylabel('\rhodot^i Error [km/s]');
grid on
set(gca, 'FontSize', 14)

subplot(3, 1, 3)
scatter(tvec, eytil(3, :))
xlabel('Time [s]')
ylabel('\phi^i Error [rad]');
grid on
set(gca, 'FontSize', 14)

sgtitle('Inovations vs Time - EKF')

% NEES and NIS Plots

Ex_mean = mean(Ex);
Ey_mean = mean(Ey);

alpha = 0.05;
r1 = chi2inv(alpha/2, N*n)/N;
r2 = chi2inv(1-alpha/2, N*n)/N;

figure() % NEES
scatter(tvec, Ex_mean)
hold on
plot(tvec, repmat(r1, 1401, 1), 'r--')
plot(tvec, repmat(r2, 1401, 1), 'r--')
hold off
ylim([2 6])
xlabel('Time [s]')
ylabel('Mean \epsilon_x')
title('EKF NEES Plot')
legend('NEES @ t_k', 'r_1 Bound', 'r_2 Bound')
grid on
set(gca, 'FontSize', 14)

r1 = chi2inv(alpha/2, N*p)/N;
r2 = chi2inv(1-alpha/2, N*p)/N;

figure() % NIS
scatter(tvec, Ey_mean);
hold on
plot(tvec, repmat(r1, 1401, 1), 'r--')
plot(tvec, repmat(r2, 1401, 1), 'r--')
hold off
ylim([1 5])
xlabel('Time [s]')
ylabel('Mean \epsilon_y')
title('EKF NIS Plot')
legend('NIS @ t_k', 'r_1 Bound', 'r_2 Bound')
grid on
```

```
set(gca, 'FontSize', 14)

figure() % TMT state plots
subplot(4, 1, 1)
plot(tvec, TMT_X, 'k')
hold on
plot(tvec, nomCon(1, :), 'r')
hold off
xlabel('Time [s]')
ylabel('TMT X [km]')
legend('TMT State', 'Nominal Trajectory')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
plot(tvec, TMT_Xdot, 'k')
hold on
plot(tvec, nomCon(2, :), 'r')
hold off
xlabel('Time [s]')
ylabel('TMT Xdot [km/s]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
plot(tvec, TMT_Y, 'k')
hold on
plot(tvec, nomCon(3, :), 'r')
hold off
xlabel('Time [s]')
ylabel('TMT Y [km]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
plot(tvec, TMT_Ydot, 'k')
hold on
plot(tvec, nomCon(4, :), 'r')
hold off
xlabel('Time [s]')
ylabel('TMT Ydot [km/s]')
grid on
set(gca, 'FontSize', 14)

sgtitle('TMT Simulated States vs Time')

figure() % TMT Measurement Plots
subplot(4, 1, 1)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TMT_y_NL_noise_out(1, ID_index), [],
 ColorSet(ii, :))
end
```

```
hold off
xlabel('Time [s]')
ylabel('TMT \rho^i [km]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 2)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TMT_y_NL_noise_out(2, ID_index), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('TMT \rhodot^i [km/s]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TMT_y_NL_noise_out(3, ID_index), [],
 ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('TMT \phi^i [rad]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
hold on
for ii = 1:12
    ID_index = find(TS_ID == ii);
    scatter(tvec(ID_index), TS_ID(ID_index), [], ColorSet(ii, :))
end
hold off
xlabel('Time [s]')
ylabel('Visible Station ID')
grid on
set(gca, 'FontSize', 14)

sgtitle('TMT Simulated Measurements vs Time')
```

EKF Implementation

```
x0 = Initial_States;
P0 = 10*eye(n);

[~, x, x_stds, ~, ~] = EKF_StatOD(x0, P0, ydata, dt, tvec, Q_EKF,
 R_EKF, Gam, TS_state);
```

```
figure('Position', [200, 200, 1800, 1000])
subplot(4, 1, 1)
hold on
plot(tvec, TMT_X, 'k')
plot(tvec, nomCon(1, :), 'g')
plot(tvec, x.pos(1, :), 'r')
hold off
xlabel('Time [s]')
ylabel('X [km]')
grid on
set(gca, 'FontSize', 14)
legend('TMT', 'Nominal','EKF Estimate','location','bestoutside')

subplot(4, 1, 2)
hold on
plot(tvec, TMT_Xdot, 'k')
plot(tvec, nomCon(2, :), 'g')
plot(tvec, x.pos(2, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Xdot [km/s]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 3)
hold on
plot(tvec, TMT_Y, 'k')
plot(tvec, nomCon(3, :), 'g')
plot(tvec, x.pos(3, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Y [km]')
grid on
set(gca, 'FontSize', 14)

subplot(4, 1, 4)
hold on
plot(tvec, TMT_Ydot, 'k')
plot(tvec, nomCon(4, :), 'g')
plot(tvec, x.pos(4, :), 'r')
hold off
xlabel('Time [s]')
ylabel('Ydot [km/s]')
grid on
set(gca, 'FontSize', 14)

sgtitle('TMT Simulated States')
```

A Solver

```
function [Atil] = Atil_Solver(State)
```

```
mu = 398600; % Earth's standard gravitational paremters [km^3/s^2]

x1 = State(1);
% x2 = State(2);
x3 = State(3);
% x4 = State(4);

Atil = [0, 1, 0, 0;
    (3*mu*x1^2)/(x1^2 + x3^2)^(5/2) - mu/(x1^2 + x3^2)^(3/2), 0,
 (3*mu*x1*x3)/(x1^2 + x3^2)^(5/2), 0;
    0, 0, 0, 1;
    (3*mu*x1*x3)/(x1^2 + x3^2)^(5/2), 0, (3*mu*x3^2)/(x1^2 +
 x3^2)^(5/2) - mu/(x1^2 + x3^2)^(3/2), 0];

end
```

C Solver

```
function [Ctil] = Ctil_Solver(State, TS_State)

x1 = State(1);
x2 = State(2);
x3 = State(3);
x4 = State(4);

z1 = TS_State(1);
z2 = TS_State(2);
z3 = TS_State(3);
z4 = TS_State(4);

Ctil = [(2*x1 - 2*z1)/(2*((x1 - z1)^2 + (x3 - z3)^2)^(1/2)), 0, (2*x3
 - 2*z3)/(2*((x1 - z1)^2 + (x3 - z3)^2)^(1/2)), 0;
    (x2 - z2)/((x1 - z1)^2 + (x3 - z3)^2)^(1/2) - ((2*x1 - 2*z1)*((x1
 - z1)*(x2 - z2) + (x3 - z3)*(x4 - z4)))/(2*((x1 - z1)^2 + (x3 -
 z3)^2)^(3/2)), (x1 - z1)/((x1 - z1)^2 + (x3 - z3)^2)^(1/2), (x4 -
 z4)/((x1 - z1)^2 + (x3 - z3)^2)^(1/2) - ((2*x3 - 2*z3)*((x1 - z1)*(x2
 - z2) + (x3 - z3)*(x4 - z4)))/(2*((x1 - z1)^2 + (x3 - z3)^2)^(3/2)),
 (x3 - z3)/((x1 - z1)^2 + (x3 - z3)^2)^(1/2);
    -(x3 - z3)/((x1 - z1)^2*((x3 - z3)^2/(x1 - z1)^2 + 1)), 0, 1/((x1
 - z1)*((x3 - z3)^2/(x1 - z1)^2 + 1)), 0];

end
```

EKF StatOD

```
function [P, x, x_stds, eytil, S] = EKF_StatOD(x0, P0, ydata, dt,
 tvec, Q, R, Gamma, TS_state)


% DEAL WITH MULT Measurement values!!!!!!!

% Matrix sizes and Steps
n = length(x0); % number of states
```

```
p = length(ydata{1}) - 1; % number of measurments, subtract one since
 it has GND station ID
steps = length(ydata); % number of steps for problem; step 1 is the
 zero time vec

%mu = 398600; % Earth's standard gravitational paremters [km^3/s^2]
Omega = dt*Gamma; % Since CT Gamma matrix is LTI we can compute Omega
 outside EKF loop

% ODE Tolerances
Rel_Tol = 1e-13;
Abs_Tol = Rel_Tol;
options = odeset('Stats', 'off', 'RelTol', Rel_Tol, 'AbsTol',
 Abs_Tol);

% initilaize variables for speed
x.neg(1:n, 1) = NaN*ones(n, 1);
x.pos(:, 1) = x0;

P.neg(1:n, 1:n, 1) = NaN*ones(n);
P.pos(:, :, 1) = P0;

x_stds(1:n, 1) = sqrt(diag(P0));

eytil = NaN*ones(2*p, 1);

S = NaN*ones(2*p, 2*p, steps);

y = NaN*ones(2*p, steps);
TS_ID = NaN*ones(2, steps);
c = NaN*ones(1, steps);

% Parse ydata into own data vector and GND station IDS
for ii = 1:steps
    if ~isempty(ydata{ii})
        [~, c(ii)] = size(ydata{ii});
        if c(ii) == 1
            y(1:3, ii) = ydata{ii}(1:3);
            TS_ID(1, ii) = ydata{ii}(4);
        elseif c(ii) == 2
            y(1:3, ii) = ydata{ii}(1:3, 1);
            TS_ID(1, ii) = ydata{ii}(4, 1);
            y(4:6, ii) = ydata{ii}(1:3, 2);
            TS_ID(2, ii) = ydata{ii}(4, 2);
        end
    end
end

No_Meas_index = isnan(TS_ID(1, :));

% EKF Loop
for ii = 2:steps
   % Prediction Step
   tspan = [tvec(ii-1) tvec(ii)];
```

```
  [~, NL_state] = ode45(@(Time, State) StatODNL_ODE(Time, State),
tspan, x.pos(:, ii-1)', options);
  x.neg(:, ii) = NL_state(end, :)';

  % NL and Jacobian Computaion (Part of Prediction Step)
  Atil = Atil_Solver(x.pos(:, ii-1));

  Ftil = eye(n) + dt*Atil;
  P.neg(:, :, ii) = Ftil*P.pos(:, :, ii-1)*Ftil' + Omega*Q*Omega';

  % Correction Step
  if No_Meas_index(ii) == 0 % There are Measurments
      if c(ii) == 1
          z1 = TS_state(ii, TS_ID(1, ii), 1);
          z2 = TS_state(ii, TS_ID(1, ii), 2);
          z3 = TS_state(ii, TS_ID(1, ii), 3);
          z4 = TS_state(ii, TS_ID(1, ii), 4);
          TS_stateK = [z1; z2; z3; z4];

          y_neg = StatOD_NLMeasurement(x.neg(:, ii), TS_stateK);

          Htil = Ctil_Solver(x.neg(:, ii), TS_stateK);

          eytil(1:3, ii) = y(1:3, ii) - y_neg;

          Ktil = P.neg(:, :, ii)*Htil'*(Htil*P.neg(:, :, ii)*Htil' +
R)^-1;

          x.pos(:, ii) = x.neg(:, ii) + Ktil*eytil(1:3, ii);
          P.pos(:, :, ii) = (eye(n) - Ktil*Htil)*P.neg(:, :, ii);

          S(1:p, 1:p, ii) = Htil*P.neg(:, :, ii)*Htil' + R;

      elseif c(ii) == 2
          % first measurment
          z1 = TS_state(ii, TS_ID(1, ii), 1);
          z2 = TS_state(ii, TS_ID(1, ii), 2);
          z3 = TS_state(ii, TS_ID(1, ii), 3);
          z4 = TS_state(ii, TS_ID(1, ii), 4);
          TS_stateK = [z1; z2; z3; z4];

          y_neg_1 = StatOD_NLMeasurement(x.neg(:, ii), TS_stateK);

          Htil_1 = Ctil_Solver(x.neg(:, ii), TS_stateK);

          eytil(1:3, ii) = y(1:3, ii) - y_neg_1;
          Ktil_1 = P.neg(:, :, ii)*Htil_1'*(Htil_1*P.neg(:, :,
ii)*Htil_1' + R)^-1;

          % second measurment
          z1 = TS_state(ii, TS_ID(2, ii), 1);
          z2 = TS_state(ii, TS_ID(2, ii), 2);
          z3 = TS_state(ii, TS_ID(2, ii), 3);
          z4 = TS_state(ii, TS_ID(2, ii), 4);
```

```
            TS_stateK = [z1; z2; z3; z4];

            y_neg_2 = StatOD_NLMeasurement(x.neg(:, ii), TS_stateK);

            Htil_2 = Ctil_Solver(x.neg(:, ii), TS_stateK);

            eytil(4:6, ii) = y(4:6, ii) - y_neg_2;

            Ktil_2 = P.neg(:, :, ii)*Htil_2'*(Htil_2*P.neg(:, :,
 ii)*Htil_2' + R)^-1;

            % Combined
            Pblock = blkdiag(P.neg(:, :, ii), P.neg(:, :, ii));
            Hblock = blkdiag(Htil_1, Htil_2);
            Rblock = blkdiag(R, R);

            x.pos(:, ii) = x.neg(:, ii) + Ktil_1*eytil(1:3, ii) +
 Ktil_2*eytil(4:6, ii);
            P.pos(:, :, ii) =  P.neg(:, :, ii) -
 Ktil_1*Htil_1*P.neg(:, :, ii) - Ktil_2*Htil_2*P.neg(:, :, ii);

            S(:, :, ii) = Hblock*Pblock*Hblock' + Rblock;
        end
    else % No Measurements
        x.pos(:, ii) = x.neg(:, ii);
        P.pos(:, :, ii) = P.neg(:, :, ii);
        eytil(:, ii) = NaN*ones(2*p, 1);
    end

    % Standard Deviations
    x_stds(:, ii) = sqrt(diag(P.pos(:, :, ii)));

end

end
```

LKF StatOD

```
function [P, dx, x_stds, eytil, S] = LKF_StatOD(dx0, P0, ydata, dt, Q,
 R, Gamma, TS_state, Nom_State)

% Matrix sizes and Steps
n = length(dx0); % number of states
p = length(ydata{1}) - 1; % number of measurments, subtract one since
 it has GND station ID
steps = length(ydata); % number of steps for problem; step 1 is the
 zero time vec

Omega = dt*Gamma; % Since CT Gamma matrix is LTI we can compute Omega
 outside EKF loop

% initilaize variables for speed
dx.neg(1:n, 1) = NaN*ones(n, 1);
dx.pos(:, 1) = dx0;

P.neg(1:n, 1:n, 1) = NaN*ones(n);
P.pos(:, :, 1) = P0;

x_stds(1:n, 1) = sqrt(diag(P0));

eytil = NaN*ones(2*p, 1);

S = NaN*ones(2*p, 2*p, steps);

y = NaN*ones(2*p, steps);
TS_ID = NaN*ones(2, steps);
c = NaN*ones(1, steps);

% Parse ydata into own data vector and GND station IDS
for ii = 1:steps
    if ~isempty(ydata{ii})
        [~, c(ii)] = size(ydata{ii});
        if c(ii) == 1
            y(1:3, ii) = ydata{ii}(1:3);
            TS_ID(1, ii) = ydata{ii}(4);
        elseif c(ii) == 2
            y(1:3, ii) = ydata{ii}(1:3, 1);
            TS_ID(1, ii) = ydata{ii}(4, 1);
            y(4:6, ii) = ydata{ii}(1:3, 2);
            TS_ID(2, ii) = ydata{ii}(4, 2);
        end
    end
end

No_Meas_index = isnan(TS_ID(1, :));

% LKF Loop
for ii = 2:steps
    % Prediction Step
    Atil = Atil_Solver(Nom_State(ii-1, :));
    Ftil = eye(n) + dt*Atil;

    dx.neg(:, ii) = Ftil*dx.pos(:, ii-1);

    P.neg(:, :, ii) = Ftil*P.pos(:, :, ii-1)*Ftil' + Omega*Q*Omega';

    % Correction Step %%% Deal with multiple measurments
    if No_Meas_index(ii) == 0 % There are Measurments
        if c(ii) == 1
        z1 = TS_state(ii, TS_ID(1, ii), 1);
        z2 = TS_state(ii, TS_ID(1, ii), 2);
        z3 = TS_state(ii, TS_ID(1, ii), 3);
        z4 = TS_state(ii, TS_ID(1, ii), 4);
        TS_stateK = [z1; z2; z3; z4];

        Htil = Ctil_Solver(Nom_State(ii, :), TS_stateK);
        Ktil = P.neg(:, :, ii)*Htil'*(Htil*P.neg(:, :, ii)*Htil' +
R)^-1;

        y_nom = StatOD_NLMeasurement(Nom_State(ii, :), TS_stateK);

        dy = y(1:3, ii) - y_nom;

        dx.pos(:, ii) = dx.neg(:, ii) + Ktil*(dy - Htil*dx.neg(:,
ii));

        P.pos(:, :, ii) = (eye(n) - Ktil*Htil)*P.neg(:, :, ii);

        S(1:3, 1:3, ii) = Htil*P.neg(:, :, ii)*Htil' + R;
        eytil(1:3, ii) = dy - Htil*dx.neg(:, ii);

        elseif c(ii) == 2
            % First measurment
            z1 = TS_state(ii, TS_ID(1, ii), 1);
            z2 = TS_state(ii, TS_ID(1, ii), 2);
            z3 = TS_state(ii, TS_ID(1, ii), 3);
            z4 = TS_state(ii, TS_ID(1, ii), 4);
            TS_stateK = [z1; z2; z3; z4];

            Htil_1 = Ctil_Solver(Nom_State(ii, :), TS_stateK);

            Ktil_1 = P.neg(:, :, ii)*Htil_1'*(Htil_1*P.neg(:, :,
ii)*Htil_1' + R)^-1;

            y_nom_1 = StatOD_NLMeasurement(Nom_State(ii, :),
TS_stateK);

            dy_1 = y(1:3, ii) - y_nom_1;

            eytil(1:3, ii) = dy_1 - Htil_1*dx.neg(:, ii);

            % Second measurment
            z1 = TS_state(ii, TS_ID(2, ii), 1);
            z2 = TS_state(ii, TS_ID(2, ii), 2);
            z3 = TS_state(ii, TS_ID(2, ii), 3);
            z4 = TS_state(ii, TS_ID(2, ii), 4);
            TS_stateK = [z1; z2; z3; z4];

            Htil_2 = Ctil_Solver(Nom_State(ii, :), TS_stateK);

            Ktil_2 = P.neg(:, :, ii)*Htil_2'*(Htil_2*P.neg(:, :,
ii)*Htil_2' + R)^-1;

            y_nom_2 = StatOD_NLMeasurement(Nom_State(ii, :),
TS_stateK);

            dy_2 = y(4:6, ii) - y_nom_2;

            eytil(4:6, ii) = dy_2 - Htil_2*dx.neg(:, ii);
            % Combined
            Pblock = blkdiag(P.neg(:, :, ii), P.neg(:, :, ii));
            Hblock = blkdiag(Htil_1, Htil_2);
            Rblock = blkdiag(R, R);

            dx.pos(:, ii) = dx.neg(:, ii) + Ktil_1*(dy_1 -
 Htil_1*dx.neg(:, ii)) + Ktil_2*(dy_2 - Htil_2*dx.neg(:, ii));

            P.pos(:, :, ii) = P.neg(:, :, ii) -
 Ktil_1*Htil_1*P.neg(:, :, ii) - Ktil_2*Htil_2*P.neg(:, :, ii);

            S(:, :, ii) = Hblock*Pblock*Hblock' + Rblock;

        end
    else
       dx.pos(:, ii) = dx.neg(:, ii);
       P.pos(:, :, ii) = P.neg(:, :, ii);
       eytil(:, ii) = NaN*ones(2*p, 1);
    end

    % Standard Deviations
    x_stds(:, ii) = sqrt(diag(P.pos(:, :, ii)));

end

end
```

StatOD NonLinear Measurements

```
function [y] = StatOD_NLMeasurement(Sat_state, TS_state)

X = Sat_state(1);
Xdot = Sat_state(2);
Y = Sat_state(3);
Ydot = Sat_state(4);

Xi = TS_state(1);
Xidot = TS_state(2);
Yi = TS_state(3);
Yidot = TS_state(4);

rho = sqrt((X - Xi)^2 + (Y - Yi)^2);
rhodot = ((X - Xi)*(Xdot - Xidot) + (Y - Yi)*(Ydot - Yidot))/sqrt((X -
 Xi)^2 + (Y - Yi)^2);
phi = atan2(Y - Yi, X - Xi);

y = [rho; rhodot; phi];

end
```

Stat OD NonLinear Noise ODE

```
function [State_Derivatives] = StatODNL_noise_ODE(Time, State)
```

```
u = 398600; % Earth's standard gravitational paremters [km^3/s^2]

X = State(1);
Xdot = State(2);
Y = State(3);
Ydot = State(4);

r = sqrt(X^2 + Y^2);

w1 = State(5);
w2 = State(6);

Xddot = -u*X/r^3 + w1;
Yddot = -u*Y/r^3 + w2;

State_Derivatives = [Xdot; Xddot; Ydot; Yddot; 0; 0];

end
```

StatOD NonLinear ODE

```
function [State_Derivatives] = StatODNL_ODE(Time, State)

u = 398600; % Earth's standard gravitational paremters [km^3/s^2]

X = State(1);
Xdot = State(2);
Y = State(3);
Ydot = State(4);

r = sqrt(X^2 + Y^2);

Xddot = -u*X/r^3;
Yddot = -u*Y/r^3;

State_Derivatives = [Xdot; Xddot; Ydot; Yddot];

end
```

Vary Color

```
function ColorSet=varycolor(NumberOfPlots)
% VARYCOLOR Produces colors with maximum variation on plots with
 multiple
% lines.
%
%     VARYCOLOR(X) returns a matrix of dimension X by 3.  The matrix
 may be
%     used in conjunction with the plot command option 'color' to vary
 the
%     color of lines.
%
```

```
%     Yellow and White colors were not used because of their poor
%     translation to presentations.
%
%     Example Usage:
%         NumberOfPlots=50;
%
%         ColorSet=varycolor(NumberOfPlots);
%
%         figure
%         hold on;
%
%         for m=1:NumberOfPlots
%             plot(ones(20,1)*m,'Color',ColorSet(m,:))
%         end
%Created by Daniel Helmick 8/12/2008
error(nargchk(1,1,nargin))%correct number of input arguements??
error(nargoutchk(0, 1, nargout))%correct number of output arguements??
%Take care of the anomolies
if NumberOfPlots<1
    ColorSet=[];
elseif NumberOfPlots==1
    ColorSet=[0 1 0];
elseif NumberOfPlots==2
    ColorSet=[0 1 0; 0 1 1];
elseif NumberOfPlots==3
    ColorSet=[0 1 0; 0 1 1; 0 0 1];
elseif NumberOfPlots==4
    ColorSet=[0 1 0; 0 1 1; 0 0 1; 1 0 1];
elseif NumberOfPlots==5
    ColorSet=[0 1 0; 0 1 1; 0 0 1; 1 0 1; 1 0 0];
elseif NumberOfPlots==6
    ColorSet=[0 1 0; 0 1 1; 0 0 1; 1 0 1; 1 0 0; 0 0 0];
else %default and where this function has an actual advantage
    %we have 5 segments to distribute the plots
    EachSec=floor(NumberOfPlots/5);

    %how many extra lines are there?
    ExtraPlots=mod(NumberOfPlots,5);

    %initialize our vector
    ColorSet=zeros(NumberOfPlots,3);

    %This is to deal with the extra plots that don't fit nicely into
 the
    %segments
    Adjust=zeros(1,5);
    for m=1:ExtraPlots
        Adjust(m)=1;
    end

    SecOne   =EachSec+Adjust(1);
    SecTwo   =EachSec+Adjust(2);
    SecThree =EachSec+Adjust(3);
    SecFour  =EachSec+Adjust(4);
    SecFive  =EachSec;
    for m=1:SecOne
        ColorSet(m,:)=[0 1 (m-1)/(SecOne-1)];
    end
    for m=1:SecTwo
        ColorSet(m+SecOne,:)=[0 (SecTwo-m)/(SecTwo) 1];
    end

    for m=1:SecThree
        ColorSet(m+SecOne+SecTwo,:)=[(m)/(SecThree) 0 1];
    end

    for m=1:SecFour
        ColorSet(m+SecOne+SecTwo+SecThree,:)=[1 0 (SecFour-m)/
(SecFour)];
    end
    for m=1:SecFive
        ColorSet(m+SecOne+SecTwo+SecThree+SecFour,:)=[(SecFive-m)/
(SecFive) 0 0];
    end

end
```

Published with MATLAB® R2020b