
HW 6: Setting Up the Term Project

Table of Contents

Initialize	1
Problem 1: Derive A, Htilde matrices	1
Problem 2: Integrate position, velocity, STM	3

Initialize

```
clearvars -except function_list pub_opt
ComputeA;
ComputeH_tilda;
clearvars -except function_list pub_opt
close all
```

```
stat_od_proj_init
ObsData = load('ObsData.txt');
```

Problem 1: Derive A, Htilde matrices

```
consts.Re = Re;
consts.area = drag.A;
consts.rho = compute_density(ri);
consts.theta_dot = theta_dot;
consts.m = drag.m;
consts.state_len = 18;

A_given = load('BatchA.mat');
A = stat_od_proj_A(state, consts);

relDiffA = abs((A(1:6,1:9) -A_given.A)./A_given.A);
fprintf('For A, elements that were exactly the same as the solution \n')
fprintf('cause Infs to show up in the log10 matrix. I replaced them\n')
fprintf('with zeros so that they were distinct from the other elements\n')
fprintf('in the histogram.\n')
hist_mat = log10(relDiffA);
hist_mat(hist_mat == -Inf) = 0;
hist(reshape(hist_mat,6*9,1));
title('A Matrix relative diff histogram.')
xlabel('Exponent')
ylabel('Num Elements')

H_given = load('BatchHtilde.mat');

consts.t = 0;
consts.site = 2;
H_tilda = stat_od_proj_H_tilda(state, consts);
relDiffHtilde = ...
    abs((H_tilda -H_given.Htildes(:, :, 1))./H_given.Htildes(:, :, 1));
```

relDiffHtilda

For A, elements that were exactly the same as the solution cause Infs to show up in the log10 matrix. I replaced them with zeros so that they were distinct from the other elements in the histogram.

relDiffHtilda =

1.0e-15 *

Columns 1 through 7

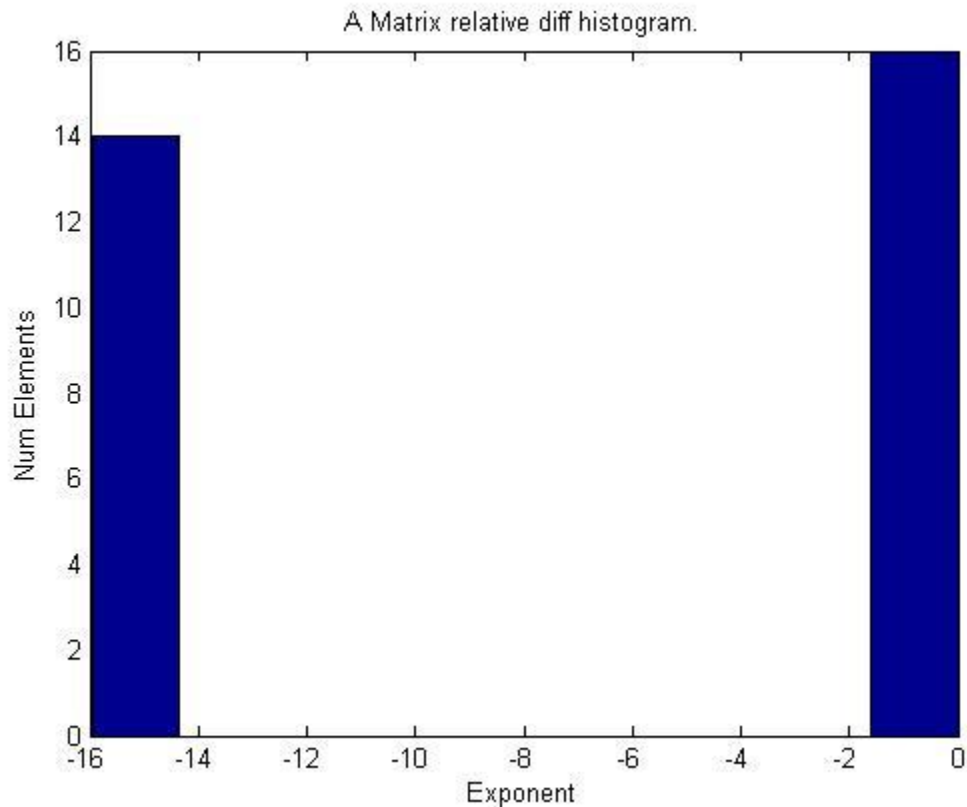
0.1361	0.2129	0	NaN	NaN	NaN	NaN
0.2591	0	0	0	0	0	NaN

Columns 8 through 14

NaN	NaN	NaN	NaN	NaN	0.1361	0.2129
NaN	NaN	NaN	NaN	NaN	0.2375	0

Columns 15 through 18

0	NaN	NaN	NaN
0	NaN	NaN	NaN

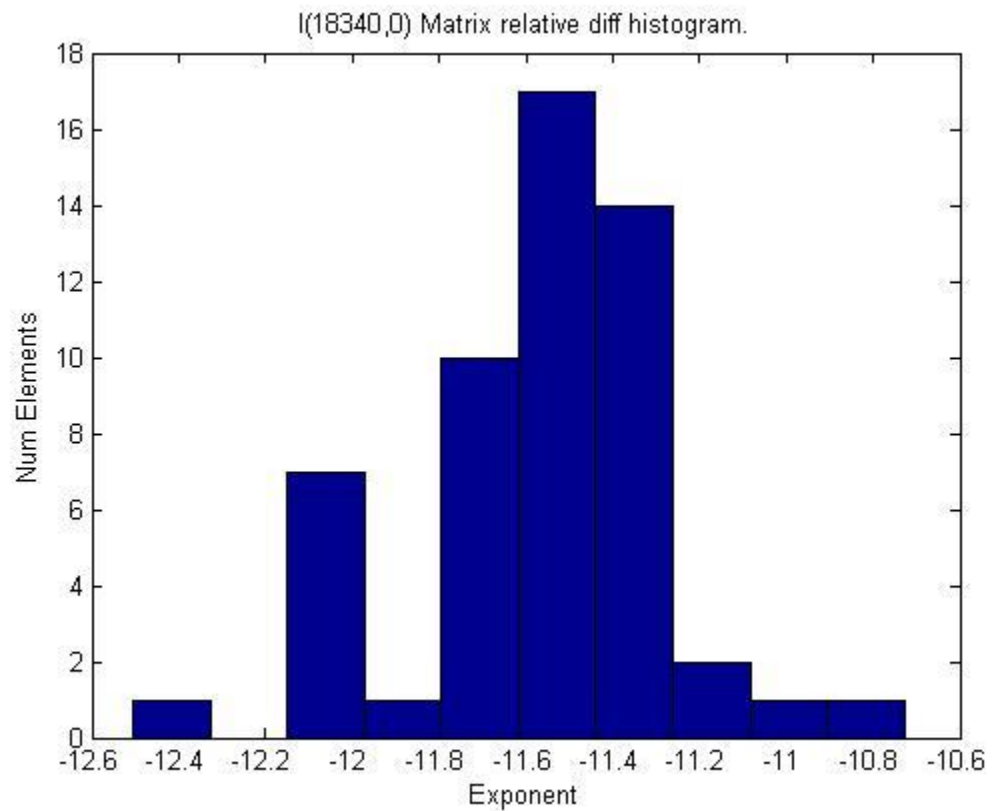
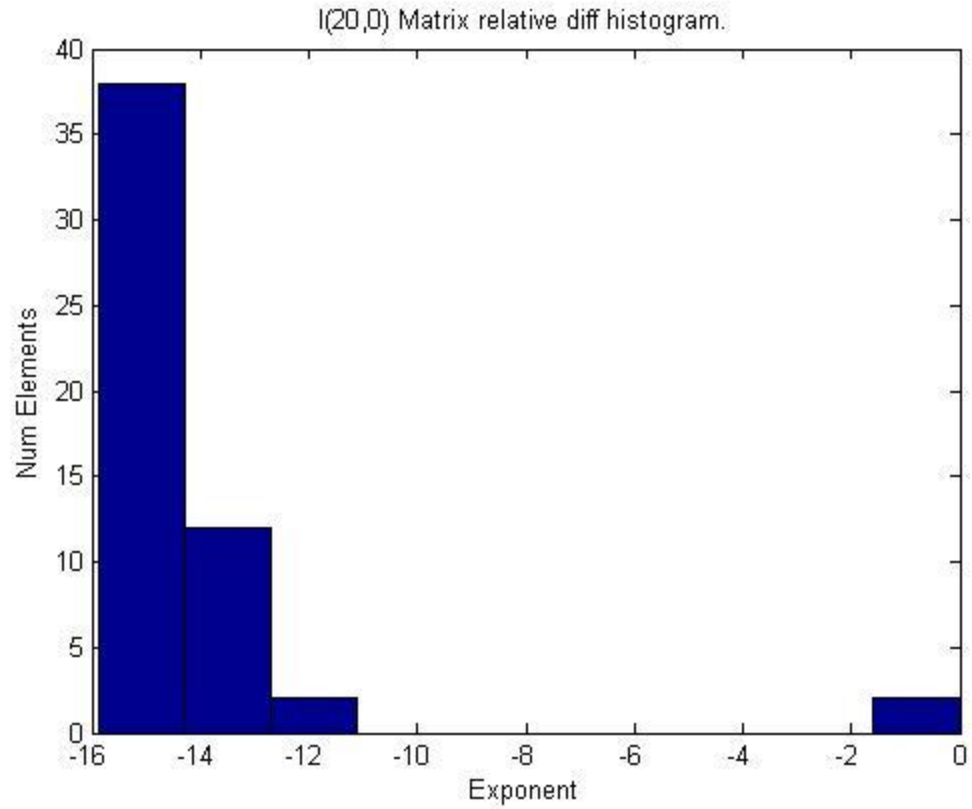


Problem 2: Integrate position, velocity, STM

```
dt = 0.1;
times = 0:dt:18340;
ode_opts = odeset('RelTol', 1e-12, 'AbsTol', 1e-20);
[T,X] = ode45(@two_body_state_dot, times, state, ode_opts, propagator_opts);

% Test the STMs against given ones
Phi_given = load('BatchPhi.mat');
STM_20 = eye(consts.state_len);
STM_20(1:important_block(1),1:important_block(2)) = ...
    reshape(X(20/dt+1,consts.state_len+1:end), ...
        important_block(1), important_block(2));
relDiffSTM20 = abs((STM_20-Phi_given.Phi_t20)./Phi_given.Phi_t20);
hist_mat = log10(relDiffSTM20);
hist_mat(hist_mat == -Inf) = 0;
figure
hist(reshape((hist_mat(1:6, 1:9)),6*9,1))
title('I(20,0) Matrix relative diff histogram.')
xlabel('Exponent')
ylabel('Num Elements')

STM_end = eye(consts.state_len);
STM_end(1:important_block(1),1:important_block(2)) = ...
    reshape(X(end,consts.state_len+1:end), ...
        important_block(1), important_block(2));
relDiffSTMend = abs((STM_end-Phi_given.Phi_t18340)./Phi_given.Phi_t18340);
% relDiffSTM20(1:6, 1:9)
figure
hist(reshape(log10(relDiffSTMend(1:6, 1:9)),6*9,1))
title('I(18340,0) Matrix relative diff histogram.')
xlabel('Exponent')
ylabel('Num Elements')
```



```

num_obs = length(ObsData(:,1));
r_residual_mat = [];
rr_residual_mat = [];
for ii = 1:num_obs
    site_num = 0;
    for jj = 1:3
        if ObsData(ii, 2) == site(jj).id
            site_num = jj;
            break
        end
    end
    t_obs = ObsData(ii,1);
    ostate = X(T(:,1)==t_obs,1:6);

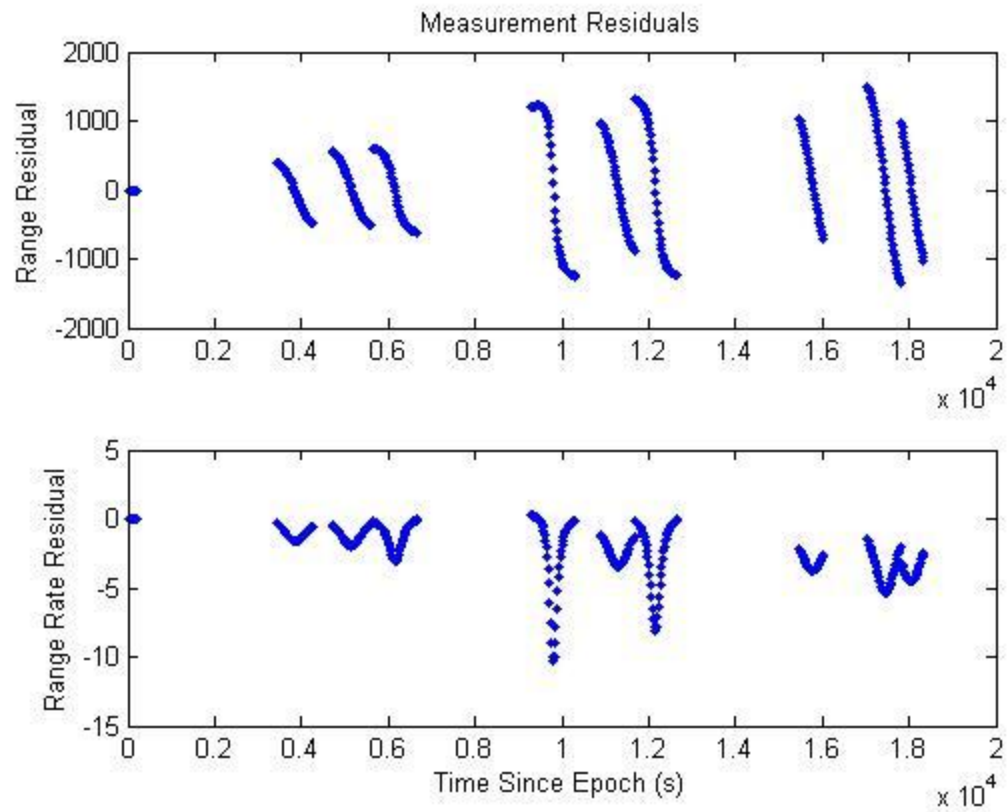
    r_comp = compute_range_ECFsite(ostate(1:3),...
        site(site_num).r,theta_dot*t_obs);
    rr_comp = compute_range_rate_ECFsite(ostate(1:6),...
        site(site_num).r,theta_dot*t_obs, theta_dot);

    r_residual_mat(ii) = (ObsData(ii,3)-r_comp);
    rr_residual_mat(ii) = (ObsData(ii,4)-rr_comp);

end
figure
subplot(2,1,1)
plot(ObsData(:,1),r_residual_mat,'.')
ylabel('Range Residual')
title('Measurement Residuals')
subplot(2,1,2)
plot(ObsData(:,1),rr_residual_mat,'.')
ylabel('Range Rate Residual')
xlabel('Time Since Epoch (s)')
RMS_r_residual = sqrt(sum(r_residual_mat.*r_residual_mat)/num_obs);
RMS_rr_residual = sqrt(sum(rr_residual_mat.*rr_residual_mat)/num_obs);
fprintf('RMS range residual: %f m \n', RMS_r_residual)
fprintf('RMS range rate residual: %f m/s \n', RMS_rr_residual)

    RMS range residual: 732.748307 m
    RMS range rate residual: 2.900165 m/s

```



Published with MATLAB® R2013b

```
function range = compute_range_ECFsite( inrtl_pos, ecf_site, theta )
%compute_range_ECFsite Summary of this function goes here
% Detailed explanation goes here
fcnPrintQueue(mfilename('fullpath'))

x = inrtl_pos(1);
y = inrtl_pos(2);
z = inrtl_pos(3);
xs = ecf_site(1);
ys = ecf_site(2);
zs = ecf_site(3);

range = sqrt(...
    (x-(xs*cos(theta)-ys*sin(theta)))*(x-(xs*cos(theta)-ys*sin(theta))) + ...
    (y-(xs*sin(theta)+ys*cos(theta)))*(y-(xs*sin(theta)+ys*cos(theta))) + ...
    (z-zs)*(z-zs));

end
```

Published with MATLAB® R2013b

```
function range_rate = compute_range_rate_ECFsite( inrtl_state, ...
    ecf_site, theta, theta_dot )
%UNTITLED2 Summary of this function goes here
% Detailed explanation goes here
fcnlPrintQueue(mfilename('fullpath'))

x = inrtl_state(1);
y = inrtl_state(2);
z = inrtl_state(3);
xdot = inrtl_state(4);
ydot = inrtl_state(5);
zdot = inrtl_state(6);
xs = ecf_site(1);
ys = ecf_site(2);
zs = ecf_site(3);

range_rate = (x*xdot+y*ydot+z*zdot...
    - (xdot*xs+ydot*ys)*cos(theta) + theta_dot*(x*xs + y*ys)*sin(theta) ...
    + (xdot*ys-ydot*xs)*sin(theta) + theta_dot*(x*ys-y*xs)*cos(theta) ...
    - zdot*zs)/compute_range_ECFsite(inrtl_state(1:3), ecf_site, theta);

end
```

Published with MATLAB® R2013b

```

%Symbolically Compute A matrix for project
fcnPrintQueue(mfilename('fullpath'))
syms x y z xdot ydot zdot mu J2 Cd slx sly slz s2x s2y s2z s3x s3y s3z
syms Re area m rho theta_dot H r0 rho0
r = sqrt(x^2+y^2+z^2);
v = sqrt(xdot^2+ymdot^2+zdot^2);
rel_wind = [xdot + theta_dot*y;
ydot - theta_dot*x;
zdot];
rel_wind_mag = sqrt(rel_wind(1)^2 + rel_wind(2)^2 + rel_wind(3)^2);
state = [x
y
z
xdot
ymdot
zdot
mu
J2
Cd
slx
sly
slz
s2x
s2y
s2z
s3x
s3y
s3z];
F = [xdot
ymdot
zdot
(-mu/r^3*x + 1.5*mu*J2*Re^2/(r^5)*(5*z^2/r^2 - 1)*x - 0.5*Cd*area/m*rho0*exp(-(
(-mu/r^3*y + 1.5*mu*J2*Re^2/(r^5)*(5*z^2/r^2 - 1)*y - 0.5*Cd*area/m*rho0*exp(-(
(-mu/r^3*z + 1.5*mu*J2*Re^2/(r^5)*(5*z^2/r^2 - 3)*z - 0.5*Cd*area/m*rho0*exp(-(
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0];
len = 18;
A = sym('A',[len len]);
for ii = 1:len
    for jj = 1:len
        A(ii,jj) = diff(F(ii),state(jj));
    end
end

```

end

% diff(state, F)

Published with MATLAB® R2013b

```

%Symbolically Compute Htilda matrix for project
fcnPrintQueue(mfilename('fullpath'))
syms x y z xdot ydot zdot mu J2 Cd slx sly slz s2x s2y s2z s3x s3y s3z
syms Re area m rho theta_dot H r0 rho0 xs ys zs theta
% syms t x(t) y(t) z(t) theta(t)
% xdot = diff(x(t),t);
% ydot = diff(y(t),t);
% zdot = diff(z(t),t);
r = sqrt(x^2+y^2+z^2);
v = sqrt(xdot^2+ydot^2+zdot^2);

state = [x
         y
         z
         xdot
         ydot
         zdot
         mu
         J2
         Cd
         slx
         sly
         slz
         s2x
         s2y
         s2z
         s3x
         s3y
         s3z];
range = sqrt((x-(xs*cos(theta)-ys*sin(theta)))^2 + ...
             (y-(xs*sin(theta) + ys*cos(theta)))^2 + (z-zs)^2);
% range = sqrt((x(t)-(xs*cos(theta(t))-ys*sin(theta(t))))^2 + ...
%             (y(t)-(xs*sin(theta(t)) + ys*cos(theta(t))))^2 + (z(t)-zs)^2);
range_rate = (x*xdot+y*ydot+z*zdot - (xdot*xs+ydot*ys)*cos(theta) + theta_dot*(x*xs
% range_rate = diff(range,t)
G = [range
     range_rate];
len = 18;
H_tilda = sym('H_tilda',[2 len]);
is_sitex = @(var) var == slx || var == s2x || var == s3x;
is_sitey = @(var) var == sly || var == s2y || var == s3y;
is_sitez = @(var) var == slz || var == s2z || var == s3z;
for ii = 1:2
    for jj = 1:len
        if is_sitex(state(jj))
            H_tilda(ii,jj) = diff(G(ii),xs);
        elseif is_sitey(state(jj))
            H_tilda(ii,jj) = diff(G(ii),ys);
        elseif is_sitez(state(jj))
            H_tilda(ii,jj) = diff(G(ii),zs);
        else
            H_tilda(ii,jj) = diff(G(ii),state(jj));

```

```
end
end
end
```

Published with MATLAB® R2013b

```
function accel = drag_accel( state, drag_data )
%drag_accel calculate drag on spacecraft
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

Cd = drag_data.Cd;
A = drag_data.A;
m = drag_data.m;

rho0 = 4e-13; %kg/m3
r0 = 7298.145; %km
H = 200.0; %km
theta_dot = 7.29211585530066e-5; %rad/s

if isfield(drag_data, 'model_params')
    if isfield(drag_data.model_params, 'rho0')
        rho0 = drag_data.model_params.rho0;
    end
    if isfield(drag_data.model_params, 'r0')
        r0 = drag_data.model_params.r0;
    end
    if isfield(drag_data.model_params, 'H')
        H = drag_data.model_params.H;
    end
    if isfield(drag_data.model_params, 'theta_dot')
        theta_dot = drag_data.model_params.theta_dot;
    end
end

r = norm(state(1:3));

rho = rho0*exp(-(r-r0)/H);
rel_wind = [state(4) + theta_dot*state(2);
            state(5) - theta_dot*state(1);
            state(6)];

accel = -0.5*Cd*A/m*rho*rel_wind*norm(rel_wind);

end
```

Published with MATLAB® R2013b

```
function fcnPrintQueue( filename )
global function_list;
if exist('function_list', 'var')
    file_in_list = 0;
    for idx = 1:length(function_list)
        if strcmp(function_list(idx), filename);
            file_in_list = 1;
            break
        end
    end
    if ~file_in_list
        %         fprintf('%s\n', filename);
        function_list = [function_list; filename];
    end
end
end
```

Published with MATLAB® R2013b

```
function accel = J2_accel( pos, params )
%J2_accel Acceleration due to J2
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Defaults (earth)
J2 = 0.00108248;
mu = 398600.4; %km3/s2
Re = 6378.145; %km

% Use input params if desired
if nargin > 1 % There are params
    if isfield(params, 'J2')
        J2 = params.J2;
    end
    if isfield(params, 'mu')
        mu = params.mu;
    end
    if isfield(params, 'Re')
        Re = params.Re;
    end
end

%Calculate accel
r = norm(pos);
z = pos(3);
const = 1.5*mu*J2*Re*Re/(r*r*r*r*r);
sin_sq_phi = z*z/(r*r);

accel = const*[5*sin_sq_phi - 1;
    5*sin_sq_phi - 1;
    5*sin_sq_phi - 3].*pos;
end
```

Published with MATLAB® R2013b

```

function A = stat_od_proj_A(state, consts)
%stat_od_proj_A Calculate A matrix for Stat OD project
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Init A, set up local vars
A = zeros(consts.state_len);
x = state(1);
y = state(2);
z = state(3);
xdot = state(4);
ydot = state(5);
zdot = state(6);
mu = state(7);
J2 = state(8);
Cd = state(9);

Re = consts.Re;
area = consts.area;
rho = consts.rho;
theta_dot = consts.theta_dot;
m = consts.m;

H = 88667; %m

% vars to reduce computations
x2 = x*x;
y2 = y*y;
z2 = z*z;
r = sqrt(x2+y2+z2);
sqrt_r = sqrt(r);
v = sqrt(xdot*xdot+ydot*ydot+zdot*zdot);
rel_wind_x = (xdot + theta_dot*y);
rel_wind_y = (ydot - theta_dot*x);
zdot2 = zdot*zdot;
rel_wind_mag = sqrt(rel_wind_x*rel_wind_x + rel_wind_y*rel_wind_y + zdot2);
Re2 = Re*Re;

rho0 = 3.614e-13; %kg/m3
r0 = 700000+6378136.3; %km
H = 88667.0; %km

% Only a few elements are populated
A(1,4) = 1;
A(2,5) = 1;
A(3,6) = 1;

A(4,1) = (3*mu*x^2)/(r*r*r*r*r) - ...
mu/(r*r*r) + ...
(3*J2*Re2*mu*((5*z2)/(r*r) - 1))/(2*(r)^(5)) - ...
(15*J2*Re2*mu*x2*z2)/(r)^(9) - ...
(15*J2*Re2*mu*x2*((5*z2)/(r*r) - 1))/(2*(r)^(2)) + ...
(Cd*area*theta_dot*rho*rel_wind_x*rel_wind_y)/(2*m*rel_wind_mag) + ...

```

```

        (Cd*area*x*rho*(xdot + theta_dot*y)*rel_wind_mag)/(2*H*m*r);
A1 = [
A2 = [
A3 = [
A4 = [          (3*mu*x^2)/(x^2 + y^2 + z^2)^(5/2) - mu/(x^2 + y^2 + z^2)^(3/2)
A5 = [  (3*mu*x*y)/(x^2 + y^2 + z^2)^(5/2) + (Cd*area*rho0*theta_dot*exp((r0 - (x^2
A6 = [
A(1,:) = A1;
A(2,:) = A2;
A(3,:) = A3;
A(4,:) = A4;
A(5,:) = A5;
A(6,:) = A6;
A4(4);

```

Published with MATLAB® R2013b

```
function H_tilda = stat_od_proj_H_tilda(state, consts)
%stat_od_proj_H_tilda Calculate H_tilda matrix for Stat OD project
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

% Init H_tilda, set up local vars
x = state(1);
y = state(2);
z = state(3);
xdot = state(4);
ydot = state(5);
zdot = state(6);
% mu = state(7);
% J2 = state(8);
% Cd = state(9);

theta_dot = consts.theta_dot;
theta = consts.t*consts.theta_dot;

%Identify the site the observation was from:
xs = state(9+(consts.site-1)*3+1);
ys = state(9+(consts.site-1)*3+2);
zs = state(9+(consts.site-1)*3+3);

H_tilda = zeros(2,18);
H_tilda(1,:) = [
H_tilda(2,:) = [ (xdot + theta_dot*ys*cos(theta) + theta_dot*xs*sin(theta))/((ys*c

% Zero out the site terms where there weren't observations.
for ii = 1:3
    if consts.site ~= ii
        H_tilda(:,9+(ii-1)*3+1:9+(ii-1)*3+3) = zeros(2,3);
    end
end
end
```

Published with MATLAB® R2013b

```
fcnPrintQueue(mfilename('fullpath'))
```

Initial data for stat OD project

```
mu = 3.986004415e14; %m3/s2
J2 = 1.082626925638815e-3;
Re = 6378136.3; %m

theta_dot = 7.2921158553e-5; %rad/s

site(1).name = 'Pacific Ocean Ship Sensor';
site(1).id = 101;
site(1).r = [-5127510.0 -3794160.0 0.0]'; % m

site(2).name = 'Pirincli, Turkey';
site(2).id = 337;
site(2).r = [3860910.0 3238490.0 3898094.0]'; % m

site(3).name = 'Thule, Greenland';
site(3).id = 394;
site(3).r = [549505.0 -1380872.0 6182197.0]'; % m

ri = [757700.0 5222607.0 4851500.0]';
vi = [2213.21 4678.34 -5371.30]';

drag.Cd = 2.0;
drag.A = 3.0; % m
drag.m = 970; %kg

% kilometerize everything
% mu = mu*1e-9;
% Re = Re*1e-3;
% site1.r = site1.r*1e-3; %km
% site2.r = site2.r*1e-3; %km
% site3.r = site3.r*1e-3; %km
% ri = ri*1e-3;%km
% vi = vi*1e-3;%km/s

state = [ri; vi; mu; J2; drag.Cd; site(1).r; site(2).r; site(3).r];

% Set up propagator options
propagator_opts.mu = mu;

propagator_opts.drag = drag;
propagator_opts.drag.use = 1;
propagator_opts.drag.model_params.rho0 = 3.614e-13; %kg/m3
propagator_opts.drag.model_params.r0 = 700000+6378136.3;
propagator_opts.drag.model_params.H = 88667;
propagator_opts.drag.model_params.theta_dot = theta_dot;

propagator_opts.J2.use = 1;
propagator_opts.J2.params.J2 = J2;
```

```
propagator_opts.J2.params.mu = mu;
propagator_opts.J2.params.Re = Re;

propagator_opts.OD.use = 1;
propagator_opts.OD.state_len = 18;
propagator_opts.OD.A_mat_handle = @stat_od_proj_A;
propagator_opts.OD.A_params.Re = Re;
propagator_opts.OD.A_params.area = drag.A;
propagator_opts.OD.A_params.rho = propagator_opts.drag.model_params.rho0;
propagator_opts.OD.A_params.theta_dot = theta_dot;
propagator_opts.OD.A_params.m = drag.m;
propagator_opts.OD.A_params.H = propagator_opts.drag.model_params.H;
important_block = [9 9];
propagator_opts.OD.A_params.important_block = important_block;
propagator_opts.OD.A_params.state_len = propagator_opts.OD.state_len;
STM_i = eye(propagator_opts.OD.state_len);
state = [state; reshape(STM_i(1:important_block(1),1:important_block(2)),...
    important_block(1)*important_block(2),1)];
```

Published with MATLAB® R2013b

```

function state_dot = two_body_state_dot(t, state, opts)
%two_body_state_dot    Return state_dot given state. Used for numerical
%integration
% The first 6 elements are assumed to be r and v. If a state transition
% matrix (STM) is to be calculated as well, opts.OD needs to be set up.
% Currently assumes r/v are the only state elements that have non-zero
% derivatives.
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

state_dot = zeros(length(state),1);
state_dot(1:3) = state(4:6);
mu = 3.986e5; % km3/s2
if isfield(opts, 'mu')
    mu = opts.mu;
end

r_vec = (state(1:3));
r = norm(r_vec);
state_dot(4:6) = -mu * r_vec/(r*r*r);

if isfield(opts, 'J2') && isfield(opts.J2, 'use')
    if opts.J2.use == 1
        if isfield(opts.J2, 'params')
            state_dot(4:6) = state_dot(4:6) + J2_accel(state(1:3), opts.J2.params)
        else
            state_dot(4:6) = state_dot(4:6) + J2_accel( state(1:3) );
        end
    end
end

if isfield(opts, 'drag') && isfield(opts.drag, 'use')
    if opts.drag.use == 1
        state_dot(4:6) = state_dot(4:6) + drag_accel( state, opts.drag );
    end
end

if isfield(opts, 'OD')
    if opts.OD.use == 1
        opts.OD.state_len;
        % The OD.state_len is the length of the estimation state. The rest
        % is the STM, numerically propagated with the A-Matrix
        A = opts.OD.A_mat_handle(state(1:opts.OD.state_len),opts.OD.A_params);

        % Block matrix multiplication
        %    long_dim = 9;
        %    STM = zeros(long_dim);
        STM = reshape(state(opts.OD.state_len+1:end),...
            opts.OD.A_params.important_block(1),...
            opts.OD.A_params.important_block(2));
        STM_dot = ...
            A(1:opts.OD.A_params.important_block(1),1:opts.OD.A_params.important_b
            *STM;
        % Pack up the important stuff

```

```
        state_dot(opts.OD.state_len+1:end) = reshape(STM_dot,...  
            opts.OD.A_params.important_block(1)*opts.OD.A_params.important_block(2  
    end  
end
```

Published with MATLAB® R2013b