FlexSat: Control Systems for Satellite Design

Advisor: Prof. Dr. Bernard Friedland

Christopher O'Hara (31459079)
cao36@njit.edu
May 4, 2019

Abstract—Control systems technology is a critical application of engineering and estimation theory in order to optimize various systems. Generally, the purpose and implementation of control systems are to improve the system throughput (manufacturing), performance, reliability, predictability, and safety. However, designing effective control systems is not a trivial task. This project will analyze the impacts of adding a compensator to an otherwise unstable system in the design of small, flexible satellites (FlexSat). For this project, a compensator is used with a star tracker sensor. Analysis of the state space and transfer functions are completed using common techniques including Nyquist, Bode, and Root Locus methods.

Index Terms—Control Systems, Space Technology, Satellite, FlexSat, Star Tracker, Military, Guidance, Navigation.

1 Introduction

Advances in control systems theory impacted every technological area of modern society. Currently, the emphasis is typically placed on robotics and automation, with entire industries being transformed in order to optimize the production, manufacturing, and distribution of products. Furthermore, control systems theory has been an extension of estimation theory, that has led to common algorithms to reduce cross-track error (e.g., PID) and accurately estimate the pose, location, and trajectory for vehicles (autonomous cars, consumer aircraft, etc.). Some of the most interesting and impressive advancements of control systems have been related to space technology and military operations.

Apart from NASA's Space Missions (i.e., Apollo) and ballistic missiles, control systems have played a crucial role in both the launching and operation of satellite missions. For launching, control feedback is able to dynamically incorporate the output of the system as it is changing. One such example is with rockets, that have variable dynamics due to changes in weight (fuel consumption) and gravitational force. For operation, drift can lead to the accumulation of errors that cause an object to deviate from its intended path. Control systems theory has been well established to handle these challenges.

A major benefit of control system design is that the system parameters and characteristics can be modeled beforehand. This allows for validation and verification of highly complex systems while minimizes costs due to device failure and misuse cases in operation (or component selection). The observability and controllability can be explicitly derived based on the initial conditions and characteristics of a system. Then, the stability can be analyzed throughout a range of varying behaviors and conditions. Approximate operational ranges then allow for robust and safety-critical systems to be created instead of blindly launching highly explosive (and expensive) devices from the ground with no confidence that the mission will succeed.

The focus on this project is to analyze typical control factors for satellites based on the concept of "flexible design" [Mazzini]." The parameters and state space characteristics for the included control system are derived from the textbook "Flexible Spacecraft Dynamics" [Mazzini]. With an emphasis on satellite design, an article on "Flexible Satellite Systems" was consulted [Lin]. Traditional control systems methods were chosen based on the instruction of "Control System Design" [Friedland]. As such, the controllability and observability are calculated for an open-loop system. Afterward, a compensator is designed using the Matlab *sisotool* in the Control Toolbox. For each implementation, Nyquist, Bode, and Root Locus criteria are observed.

2 Related Work

Sensor: Star Trackers

During the 1960s, *Inertial Navigation Systems* (INS) were not accurate enough to reliably guide the Apollo Missions or guide *Intercontinental Ballistic Missiles* (ICBM). *Angle Random Walk* (ARW) or "white noise" is a well-known limitation for sensors and actuators, that negatively affect performance and reliability. ICBMs, for example, require a tactical grade H^{-1} accuracy of better than 10^{-3} (gyroscope error/bias) [Nebylov]. *Kalman Filters* were employed to reduce the magnitude of the error via sensor fusion and estimation [Hobbs], but it was not until recently (2000s) that high-performance MEMS sensors made INS innately viable.

Prior to these MEMS sensors, *Star Trackers* were used. *Star Trackers* are a type of highly-accurate optical sensor, historically used for space flight and ballistic missile navigation. The general idea is that the locations of the stars have been measured with a high level of accuracy and the results are placed in catalogs [Fig. 1]. The locations and alignment of stars are static (relative) and thus a reliable method of tracking an objects location over time.

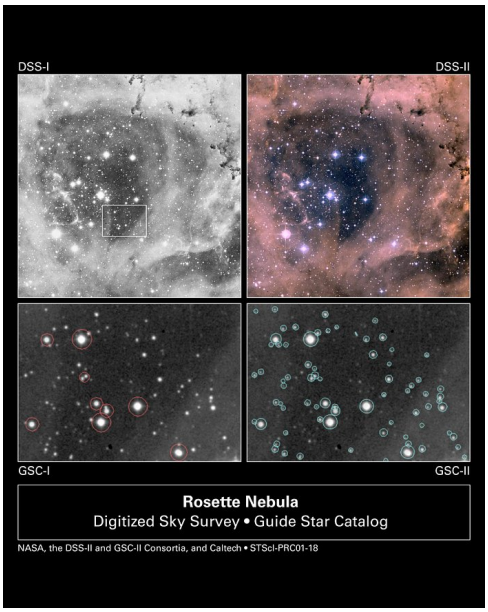

Fig. 1: Star Catalog example of the Rosette Nebula [6].

Root-Loci Method

The Root-Loci method is useful for plotting the complex plane of roots for $T(s) = 0$ (the poles of a closed-loop system). [Friedland] gives the basic root locus equation as:

$$1 + KG(s) = 1 + K\frac{\sum_{i=1}^{n_z}(s - z_i)}{\sum_{i=1}^{n_p}(s - p_i)} = 0 \tag{1}$$

where z and p represent the zeroes and poles, respectively, and the difference from the s-point results in a phasor. Essentially, the root-loci method is a graphical representation of the characteristic equation:

$$\Delta(s) = D(s) + KN(s) = 0 \tag{2}$$

Nyquist Method

A Nyquist diagram is a useful method of determining the stability of a system. [Friedland] notes that the *Nyquist Stability Criterion* is given as: 'A system having a return difference $1 + KG(s)$ is stable if and only if the Nyquist diagram, i.e., the map of the imaginary axis, does not encircle the point $1/K$ in the clockwise direction,' where $G(s) = -1/K$. Essentially, G(s) maps the right-half of the s plane onto the z-plane. If $G(s) \neq -1/K$, then the system is stable. The Nyquist diagram can also be used to analyze '*conditionally stable*' systems (i.e., the gain K is encapsulated in a stable region), though this is still problematic as changes to the gain under dynamic conditions can result in the system becoming unstable.

Bode Plot Method

An effective method for analyzing the *Gain* and *Phase* margins are to use a Bode plot. The gain margin describes the amount of variation in the game without making the system unstable whereas the phase margin describes the amount of 'shifting' permissible without inducing instability. Effectively, a Bode plot provides the same information as the Nyquist diagram, but in the dB scale (which is more intuitive for some engineers and scientist to analyze). It is customary to plot the Bode amplitude as:

$$D(\omega) = 10log_{10}|G(j\omega)| \tag{3}$$

where $|G(j\omega)|$ is the magnitude of the loop transmission G(s).

Proportional-Integral Compesators

One of the most common compensator designs is for a PI-compensator. A compensator places a transfer function between the plant input and the measured error. If D(s) has a pole, the open-loop system type will increase. [Friedland] gives the PI-compensator as:

$$D(s) = \frac{K_1}{s} + K_2 = \frac{K_1 + K_2 s}{s} \tag{4}$$

The phase lead (which is present in the root-loci method) is given as $T_{lead}(s) = \frac{s-z}{s-p}$. Both the pole and zero location should be close to the origin, in the left-hand plane. Since there is only one pole and one zero, they both should be located on the real axis. Phase lead compensators help to shift the poles of the transfer function to the left, which is beneficial for stability purposes. There are other types of compensators and conditions but they will not be described here as they are irrelevant for the project.

PROJECT

The purpose of the project is to explore real-world scenarios that utilize the aforementioned control systems methods. Such a case, that is relevant for both control systems and space technology, is small satellite design [Baumann et al.]. CubeSats are used by NASA and the ESA for many operations including data collection, network utilization, and other ICT tasks [Silva et al.]. The Institute of Aerospace department at TU Berlin (in cooperation with the DLR) has many small-sized, large-scale distributed satellite types including the BEESAT and CanSat [Fig. 2].

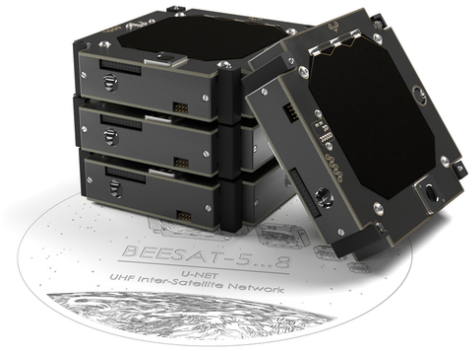

Fig. 2: BEESAT 8 Model/Design at TU Berlin [9].

Parameter Setup

Taking the specifications and state-space models from [Mazzini], Matlab can be used to simulate the system performance, stability, and feasibility. The dynamics/kinematic parameters are displayed below, in Table 1.

Variable	Value	Description
J_s	31.38	Satellite inertia $kg * m^2$
ω	2.6268	Flexible mode frequency $rad * s^{-1}$
D_r	0.00495	Flexible mode damping ratio
α	0.5736	Coupling coefficient
ϵ	0.00495	Flexible mode damping ratio
J_r	0.0004	Reaction wheel inertia $kg * m^2$
τc_{max}	0.05	Commanded torque saturation $N * m$
ωr_{max}	28005	Reaction wheel velocity saturation rpm
ω_{sat}	$\omega r_{max} \frac{2\pi}{60}$	Normalized saturation
T_m	0.8	Measurement delay second $N * m$
W	10^{-9}	Measurement noise variance
T_e	0.05	Estimator time constant second
τ_d	0.001	Disturbance torque $N * m$
n_0	0.0001	Initial condition of the state n

TABLE 1: Flexible Satellite Parameters.

Simulink Design

The first step it to utilize Simulink for developing the state-space. For the open-loop and closed-loop systems, an actuator, the satellite system, and a sensor (star tracker) are needed. Below, the designs are shown:

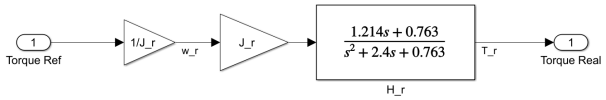

Fig. 3: Actuator (Component) Diagram.

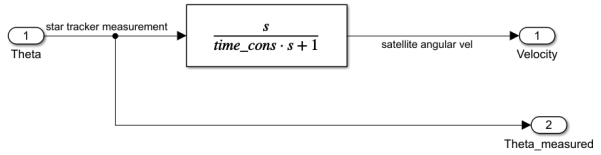

Fig. 4: Star Tracker (Component) Diagram.

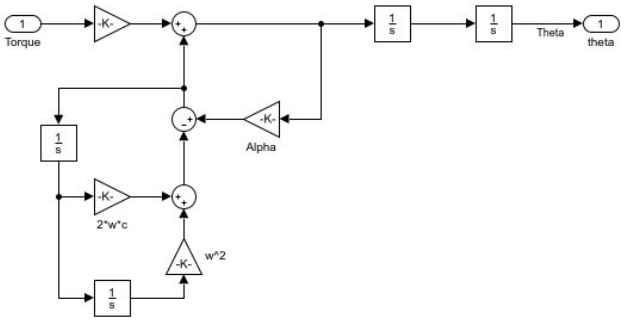

Fig. 5: Satellite (Component) Diagram.

Next, the open and closed-loop systems are designed with the aforementioned components. The diagrams below show the systems that utilize integrators (compensator).

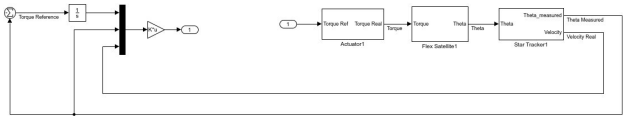

Fig. 6: Closed-Loop Integrator System. The input torque leads to an actuator which is fed to the satellite and then to the star tracker. The velocity and theta are measured and the result is fused with the torque reference.

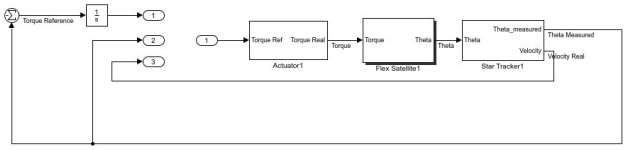

Fig. 7: Open-Loop Integrator System. The input torque leads to an actuator which is fed to the satellite and then to the star tracker. The velocity and theta are measured and the result is fused with the torque reference.

Matlab Computation

The initial transfer function can be written in the form:

$$a = [J_s\ 0\ 0]; b = [0\ 0\ 1] \rightarrow \frac{1}{31.38s^2}$$

The initial state-space can be written in the following form:

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix};\ B = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

$$C = \begin{bmatrix} 0 & 0.03191 \end{bmatrix};\ D = 0$$

Using Matlab to solve for the poles:

```
        Pole              Damping       Frequency

  0.00e+00                -1.00e+00      0.00e+00
  0.00e+00                -1.00e+00      0.00e+00
 -3.05e-02 + 4.02e+00i     7.58e-03      4.02e+00
 -3.05e-02 - 4.02e+00i     7.58e-03      4.02e+00
```

The state-space for the system can be written as:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-\omega_{mode}^2}{1-\alpha} & -2\frac{c_{damp}\omega_{mode}}{1-\alpha} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-\omega_{mode}^2}{1-\alpha} & -2\frac{c_{damp}\omega_{mode}}{1-\alpha} \end{bmatrix};$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{J_s(1-\alpha)} \\ 0 \\ \frac{\alpha-2}{J_s(1-\alpha)} \end{bmatrix};$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix};\ D = 0$$

Using Matlab to solve for variables:

```
A =
           x1         x2         x3         x4
 x1         0          1          0          0
 x2         0          0    -16.18  -0.06099
 x3         0          0          0          1
 x4         0          0    -16.18  -0.06099

B =
          u1
 x1        0
 x2  0.07474
 x3        0
 x4   0.4241

C =
     x1  x2  x3  x4
 y1   1   0   0   0

D =
     u1
 y1   0
```

Now, several control methods can be used to analyze the system:

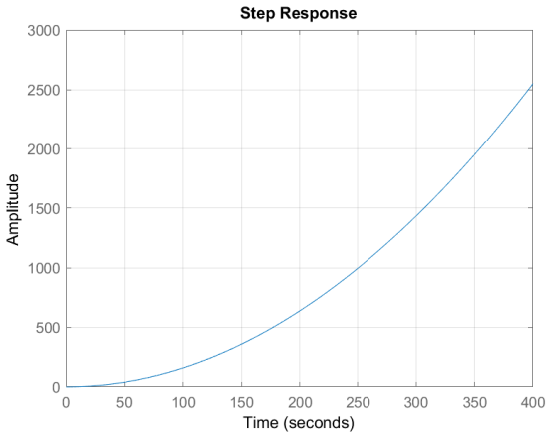

Fig. 8: The step response of the system.

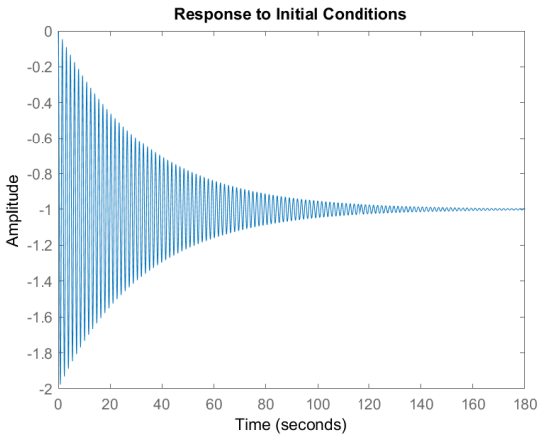

Fig. 9: The amplitude response over time. The dampening causes the system to eventually be conditionally stable (but requires nearly three minutes).

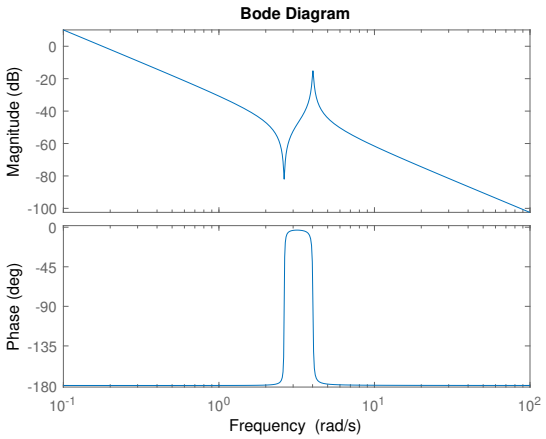

Fig. 10: Bode Plot demonstrating the magnitude and phase for the stability ranges. Note that the Magnitude (top) also matches the Single Value Analysis method for SISO systems.

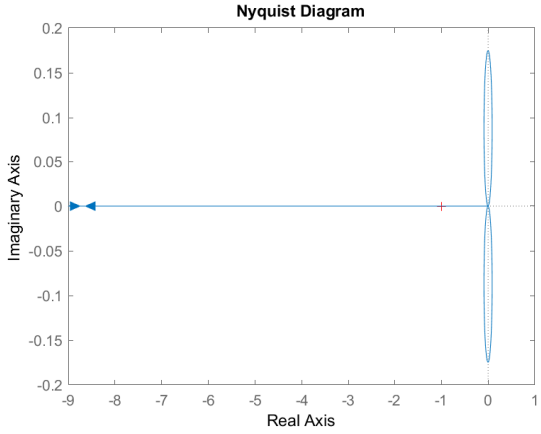

Fig. 11: Nyquist Diagram showing the relative stability of the system.

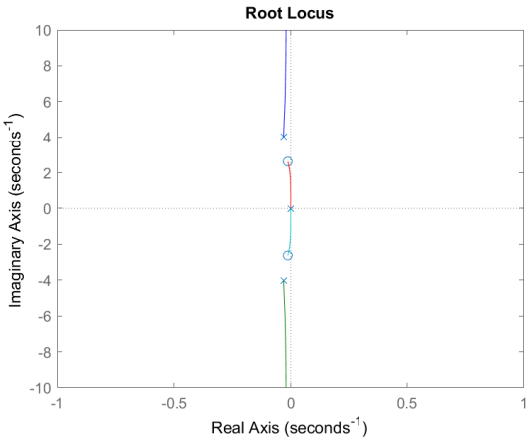

Fig. 12: Root Loci Diagram demonstrating the location of the poles and zeros.

Furthermore, the controllability and observability can be calculated. Using Matlab, we find:

```
Co =

         0    0.0747   -0.0026   -0.6935
    0.0747   -0.0026   -0.6935    0.0846
         0    0.0429   -0.0026   -0.6935
    0.0429   -0.0026   -0.6935    0.0846


Co_rank =

     4


Ob =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0  -16.1822   -0.0610
         0         0    0.9869  -16.1785


Ob_rank =

     4
```

Since the rank of both matrices is equal to its dimensions, this system is both controllable and observable.

Now, we can add a PI-compensator to improve the system's stability and performance. However, adding new parameters and components will change the state-space.

Using Matlab with `linmod` for the `'Modal_Control_Open'` model, we return the following:

```
sys =

  A =
       x1   x2       x3       x4       x5         x6   x7
   x1   0    0        0        0        0          0    1
   x2   1   -2        0        0        0          0    0
   x3   0    0     -2.4   -0.763        0          0    0
   x4   0    0        1        0        0          0    0
   x5   0    0  0.05204  0.03271 -0.06099     -16.18    0
   x6   0    0        0        0        1          0    0
   x7   0    0  0.09073  0.05702 -0.06099     -16.18    0

  B =
       u1
   x1   0
   x2   0
   x3   1
   x4   0
   x5   0
   x6   0
   x7   0

  C =
       x1  x2  x3  x4  x5  x6  x7
   y1   2  -4   0   0   0   0   0
   y2   1   0   0   0   0   0   0

  D =
       u1
   y1   0
   y2   0
```

Solving for the initial gain results in:

$$K = [6.1483 \;\; 0.5174]$$

Next, we will need to compensate the integrator component. Solving for the integrator gain leads to:

$$K_{int} = [0.0223 \quad -0.5572 \quad -5.7202]$$

Updating these new values to the gain results in an unstable system:

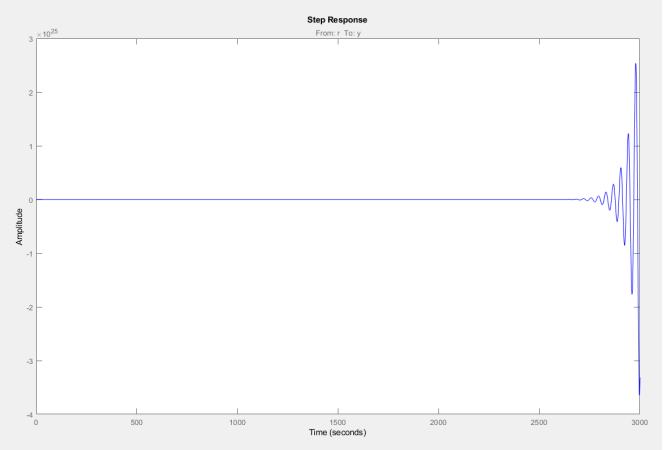

Fig. 13: Step Response of Open Integrator System.

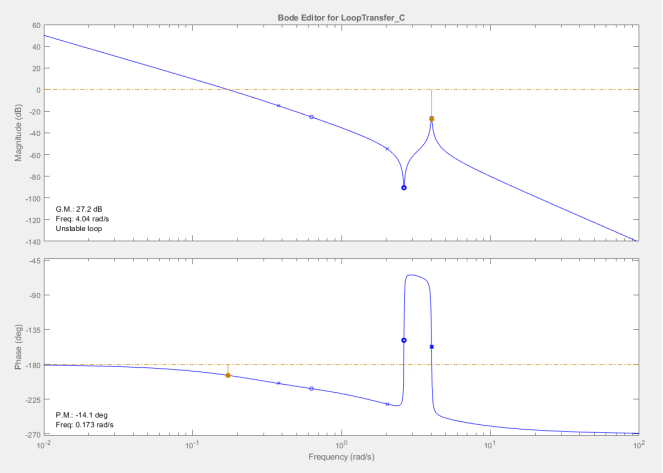

Fig. 14: Bode Plot of Open Integrator System.

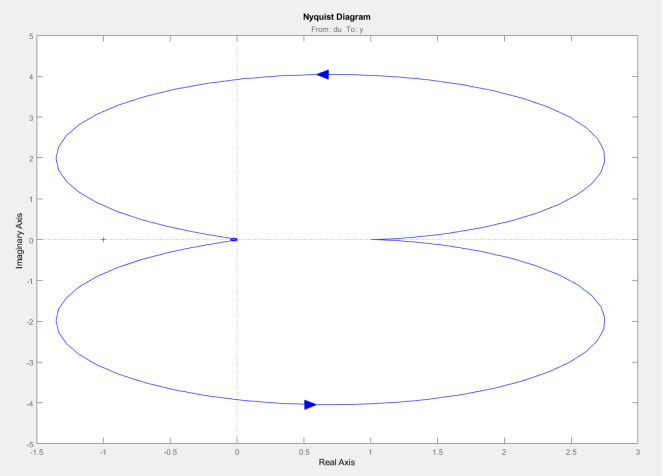

Fig. 15: Nyquist Diagram of Open Integrator System.

Using the automatic tuning feature in the Control Design GUI, we can automatically calculate values that will lead to a stable system (tuned around the bandwidth). The results can be used to calculate a new gain value relative to the transfer function:

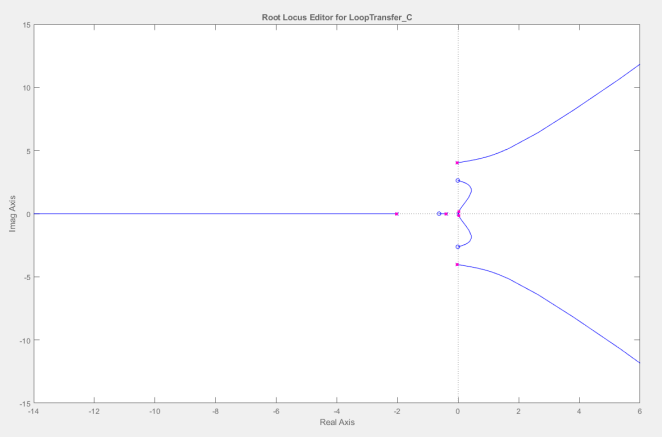

Fig. 16: Root Loci Diagram of Open Integrator System.

```
K1 =

  5.782 s + 0.23
  --------------
    1.33 s + 1
```

With the compensator:

```
C=

  4.347 (s+0.03978)
  -----------------
     (s+0.7519)
```

Afterward, the leading constants in the transfer functions can be scaled (effective gain) and the controlled system will be stable:

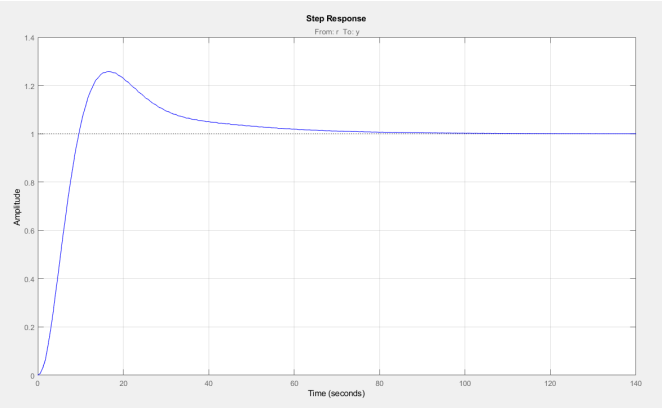

Fig. 17: Step Response of Controlled Integrator System.

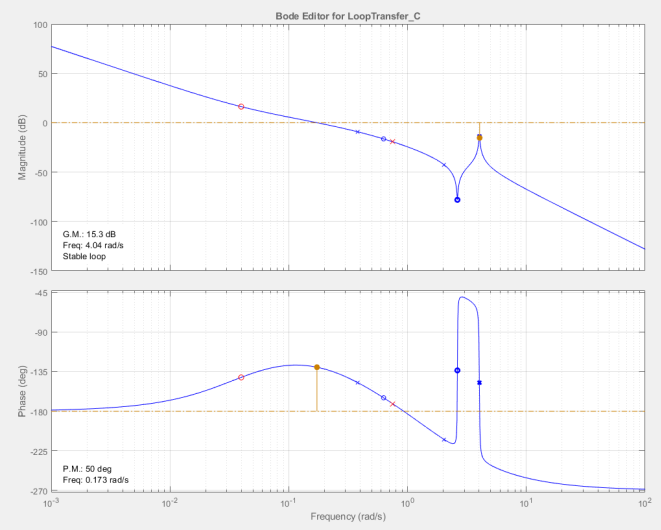

Fig. 18: Bode Plot of Controlled System.

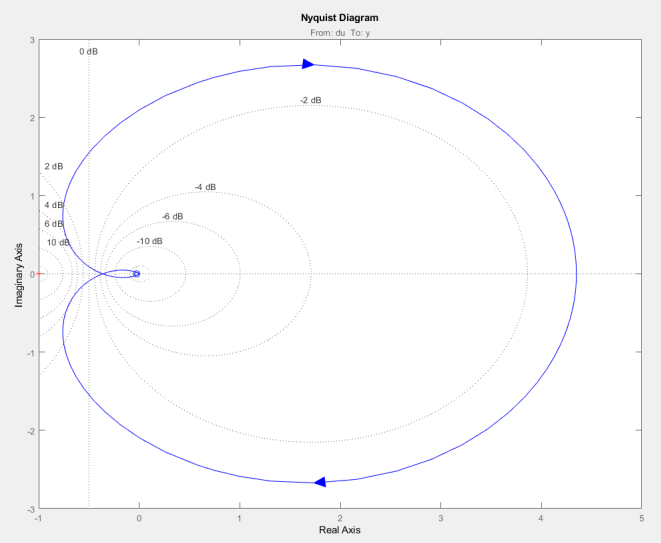

Fig. 19: Nyquist Diagram of Controlled System.

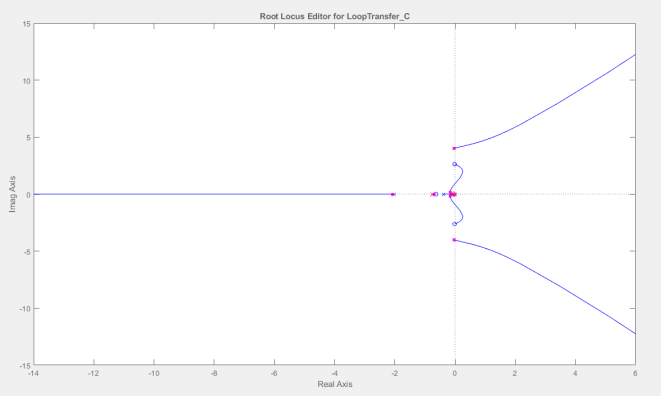

Fig. 20: Root Loci Diagram of Controlled System.

Now, the system has become stable and can be used in further scenarios and conditions.

CONCLUSIONS

The main task of this project has been to investigate the usage of relevant control theory analysis methods on a realistic system. The overall implementation is relatively at the intermediate level of control systems theory, as many other algorithms could have been added (i.e., Kalman Filter, Linear Quadratic Control). However, even an implementation such as provided is nontrivial and requires careful engineering. A major extension of this project is that the parameters values and specifications can be changed (thus flexible) and the control systems design toolbox and automated tuning features can also be used.

This leads to a major benefit of such a project. Once the core competencies have been established, they can be applied to many systems. The simulator star tracker sensor can be exchanged (or fused) with other sensors. The actuator can be designed to meet the specifications of the satellite system, or changed completely to match an assembly line, robotic system, or something completely novel.

Future Work

As alluded in the Conclusions above, it would be nice to explore some additional control systems methods. Furthermore, the system described is not immediately suitable for validating the control of a satellite under real conditions. Additional dynamics need to be incorporated related to the environment and temporal changes in parameters. It is intended to continue this work in the Summer of 2019 working on a CanSat system that will be launched out of the DECAN rocket at the Institute of Aerospace, Space Technology Department, TU Berlin (to a distance of 10km).

At NJIT, this project might be extended to an MSc Mechatronics Course, ECE666 - Control Systems II. Furthermore, it might be possible to continue this work as part of an Independent Study module.

3 APPENDIX

Below contains the .m files for the project:

Listing 1: constants.m

```
J_s = 31.38;
w_mode = 2.6268;
c_damp = 0.00495;
alpha = 0.5736;
J_r = 0.0004;
T_sat = 0.05;
w_sat = 2800*2*pi/60;
delay = 0.8;
noise = 10^-9;
time_cons = 0.5;
dis_torque = 0.001;
in_nu = 0.0001;
```

Listing 2: FlexSat_Open_Loop.m

```
%% Open-Loop Init
constants;
T = 1;
b = [0 0 1];
a = [J_s 0 0];
TF = tf(b,a)
[A,B,C,D] = tf2ss(b,a)
Poles = pole(TF)
%% SS and Dampening
constants;
A = [0 1 0 0 ; 0 0 -w_mode^2/(1-alpha) -2*c_damp*w_mode/(1-alpha) ; 0 0 0 1 ; 0 0 -w_mode^2/(1-alpha) -2*c_damp*w_mode/(1-
    alpha)];
B = [0 ; 1/(J_s*(1-alpha)) ; 0 ; alpha-2/(J_s*(1-alpha))];
C = [1 0 0 0];
D = [0];
sys = ss(A,B,C,D);
damp(sys)
%% linmod
constants;
[A,B,C,D] = linmod('Flex_Satellite');
sys = ss(A,B,C,D)
damp(sys);
%% Step, Initialize, Controllability, and Observability
constants;
A = [0 1 0 0 ; 0 0 -w_mode^2/(1-alpha) -2*c_damp*w_mode/(1-alpha) ; 0 0 0 1 ; 0 0 -w_mode^2/(1-alpha) -2*c_damp*w_mode/(1-
    alpha)];
B = [0 ; 1/(J_s*(1-alpha)) ; 0 ; alpha/(J_s*(1-alpha))];
C = [1 0 0 0];
D = [0];
sys = ss(A,B,C,D);
figure(1);
step(sys)
grid;
x0 = [0 ; 0 ; 1 ; 0];
figure(2);
initial(sys,x0)
figure(3);
bode(sys)
figure(4);
nichols(sys)
figure(5);
nyquist(sys)
figure(6);
rlocus(sys)
Co = ctrb(sys)
Co_rank = rank(Co)
Ob = obsv(sys)
Ob_rank = rank(Ob)
```

Listing 3: FlexSat_Closed_Loop.m

```
%% Modal Open Init
constants;
[A,B,C,D] = linmod('Modal_Control_Open');
sys = ss(A,B,C,D)
pole(sys)
%% Eigenstructure
constants;
sys = linmod('Modal_Control_Open');
delta = 0.7;
omega = 0.2;
lambda = roots([1 2*delta*omega omega^2]);
VW1 = null([sys.a-lambda(1)*eye(7) sys.b]);
```

```
VW2 = null([sys.a-lambda(2)*eye(7) sys.b]);
V = [VW1(1:7) VW2(1:7)];
W = [VW1(8) VW2(8)];
K = -real(W/(sys.c*V))
damp(sys.a-sys.b*K*sys.c)
%% Modal Close Bode
constants;
[A,B,C,D] = linmod('Modal_Control_Closed');
sys = ss(A,B,C,D)
bode(sys)
%nyquist(sys)
%% Open Integrator Init
constants;
[A,B,C,D] = linmod('Modal_Control_Open_Integ');
sys = ss(A,B,C,D)
pole(sys)
%% Eigenstructure Integral
constants;
sys = linmod('Modal_Control_Open_Integ');
delta = 0.7;
omega = 0.13;
lambda = [roots([1 2*delta*omega omega^2 ]); -omega]
VW1 = null([sys.a-lambda(1)*eye(8) sys.b]);
VW2 = null([sys.a-lambda(2)*eye(8) sys.b]);
VW3 = null([sys.a-lambda(3)*eye(8) sys.b]);
V = [VW1(1:8) VW2(1:8) VW3(1:8)];
W = [VW1(9) VW2(9) VW3(9)];
Kint = real(W/(sys.c*V))
damp(sys.a+sys.b*Kint*sys.c);
%% Close Bode Integral
constants;
[A,B,C,D] = linmod('Modal_Control_Closed_Integ');
sys = ss(A,B,C,D)
bode(sys)
%nyquist(sys)
%% Phase-lead SISO
constants;
[A,B,C,D] = linmod('Phase_Control_Open');
G = ss(A,B,C,D)
bode(G)
sisotool (G);
%%
constants;
[A,B,C,D] = linmod('Phase_Control_K1');
sys = ss(A,B,C,D)
%bode(sys)
dcgain(sys)
%%
constants;
[A,B,C,D] = linmod('Phase_Control_Open');
G = ss(A,B,C,D)
K1 = tf([18.9*1.33*0.23 0.23],[1.33 1]);
sisotool(G,K1)
%%
constants;
t2 = 67;
[A,B,C,D] = linmod('Phase_Control_K2');
sys = ss(A,B,C,D)
bode(sys)
```

Below contains the terminal printouts for both open loop and closed looped systems:

```
>> FlexSat_Open_Loop

TF =

      1
  ---------
  31.38 s^2

Continuous-time transfer function.


A =

     0     0
     1     0


B =

     1
     0


C =

         0    0.0319


D =

     0


Poles =

     0
     0


         Pole              Damping       Frequency      Time Constant
                                        (rad/seconds)      (seconds)

  0.00e+00                -1.00e+00       0.00e+00             Inf
  0.00e+00                -1.00e+00       0.00e+00             Inf
 -3.05e-02 + 4.02e+00i     7.58e-03       4.02e+00         3.28e+01
 -3.05e-02 - 4.02e+00i     7.58e-03       4.02e+00         3.28e+01

sys =

  A =
             x1        x2        x3        x4
   x1         0         0         0         1
   x2         0  -0.06099    -16.18         0
   x3         0         1         0         0
   x4         0  -0.06099    -16.18         0

  B =
           u1
   x1       0
   x2  0.04287
   x3       0
   x4  0.07474

  C =
      x1  x2  x3  x4
   y1   1   0   0   0

  D =
      u1
   y1   0

Continuous-time state-space model.


         Pole              Damping       Frequency      Time Constant
                                        (rad/seconds)      (seconds)

  0.00e+00                -1.00e+00       0.00e+00             Inf
  0.00e+00                -1.00e+00       0.00e+00             Inf
 -3.05e-02 + 4.02e+00i     7.58e-03       4.02e+00         3.28e+01
 -3.05e-02 - 4.02e+00i     7.58e-03       4.02e+00         3.28e+01

Co =
```

```
         0    0.0747   -0.0026   -0.6935
    0.0747   -0.0026   -0.6935    0.0846
         0    0.0429   -0.0026   -0.6935
    0.0429   -0.0026   -0.6935    0.0846


Co_rank =

     4


Ob =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0  -16.1822   -0.0610
         0         0    0.9869  -16.1785


Ob_rank =

     4
```

```
>> FlexSat_Closed_Loop

sys =

  A =
             x1        x2        x3        x4        x5        x6        x7
   x1         0         0         0         0         0         0         1
   x2         1        -2         0         0         0         0         0
   x3         0         0      -2.4    -0.763         0         0         0
   x4         0         0         1         0         0         0         0
   x5         0         0   0.05204   0.03271  -0.06099    -16.18         0
   x6         0         0         0         0         1         0         0
   x7         0         0   0.09073   0.05702  -0.06099    -16.18         0

  B =
       u1
   x1   0
   x2   0
   x3   1
   x4   0
   x5   0
   x6   0
   x7   0

  C =
       x1  x2  x3  x4  x5  x6  x7
   y1   2  -4   0   0   0   0   0
   y2   1   0   0   0   0   0   0

  D =
       u1
   y1   0
   y2   0

Continuous-time state-space model.


ans =

  -2.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
  -0.3772 + 0.0000i
  -2.0228 + 0.0000i
  -0.0305 + 4.0226i
  -0.0305 - 4.0226i


K =

    6.1483    0.5174


         Pole              Damping       Frequency       Time Constant
                                      (rad/TimeUnit)     (TimeUnit)

 -1.93e-02 + 4.03e+00i     4.79e-03       4.04e+00          5.17e+01
 -1.93e-02 - 4.03e+00i     4.79e-03       4.04e+00          5.17e+01
 -2.49e+00                 1.00e+00       2.49e+00          4.01e-01
 -1.48e+00                 1.00e+00       1.48e+00          6.76e-01
 -1.40e-01 + 1.43e-01i     7.00e-01       2.00e-01          7.14e+00
 -1.40e-01 - 1.43e-01i     7.00e-01       2.00e-01          7.14e+00
 -1.70e-01                 1.00e+00       1.70e-01          5.90e+00
```

```
sys =

  A =
           x1        x2        x3        x4        x5        x6        x7
   x1       0         0         0         0         0         0         1
   x2       1        -2         0         0         0         0         0
   x3       0         0      -2.4    -0.763       0         0         0
   x4       0         0         1         0         0         0         0
   x5       0         0   0.05204   0.03271  -0.06099    -16.18       0
   x6       0         0         0         0         1         0         0
   x7       0         0   0.09073   0.05702  -0.06099    -16.18       0

  B =
       u1
   x1   0
   x2   0
   x3   1
   x4   0
   x5   0
   x6   0
   x7   0

  C =
          x1      x2      x3      x4      x5      x6      x7
   y1   12.81  -24.59       0       0       0       0       0

  D =
       u1
   y1   0

Continuous-time state-space model.


sys =

  A =
           x1        x2        x3        x4        x5        x6        x7        x8
   x1       0        -1         0         0         0         0         0         0
   x2       0         0         0         0         0         0         0         1
   x3       0         1        -2         0         0         0         0         0
   x4       0         0         0      -2.4    -0.763       0         0         0
   x5       0         0         0         1         0         0         0         0
   x6       0         0         0   0.05204   0.03271  -0.06099    -16.18       0
   x7       0         0         0         0         0         1         0         0
   x8       0         0         0   0.09073   0.05702  -0.06099    -16.18       0

  B =
       u1
   x1   0
   x2   0
   x3   0
   x4   1
   x5   0
   x6   0
   x7   0
   x8   0

  C =
       x1  x2  x3  x4  x5  x6  x7  x8
   y1   1   0   0   0   0   0   0   0
   y2   0   1   0   0   0   0   0   0
   y3   0   2  -4   0   0   0   0   0

  D =
       u1
   y1   0
   y2   0
   y3   0

Continuous-time state-space model.


ans =

   0.0000 + 0.0000i
  -2.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
  -0.3772 + 0.0000i
  -2.0228 + 0.0000i
  -0.0305 + 4.0226i
  -0.0305 - 4.0226i


lambda =

  -0.0910 + 0.0928i
```

```
  -0.0910 - 0.0928i
  -0.1300 + 0.0000i


Kint =

    0.0223   -0.5572   -5.7202


         Pole              Damping       Frequency       Time Constant
                                      (rad/TimeUnit)     (TimeUnit)

 -2.00e-02 + 4.03e+00i     4.96e-03       4.03e+00          5.00e+01
 -2.00e-02 - 4.03e+00i     4.96e-03       4.03e+00          5.00e+01
 -2.48e+00                 1.00e+00       2.48e+00          4.04e-01
 -1.50e+00                 1.00e+00       1.50e+00          6.67e-01
 -9.10e-02 + 9.28e-02i     7.00e-01       1.30e-01          1.10e+01
 -9.10e-02 - 9.28e-02i     7.00e-01       1.30e-01          1.10e+01
 -1.30e-01                 1.00e+00       1.30e-01          7.69e+00
 -1.32e-01                 1.00e+00       1.32e-01          7.58e+00

sys =

  A =
             x1        x2        x3        x4        x5        x6        x7        x8
   x1         0        -1         0         0         0         0         0         0
   x2         0         0         0         0         0         0         0         1
   x3         0         1        -2         0         0         0         0         0
   x4         0         0         0      -2.4    -0.763         0         0         0
   x5         0         0         0         1         0         0         0         0
   x6         0         0         0   0.05204   0.03271  -0.06099    -16.18         0
   x7         0         0         0         0         0         1         0         0
   x8         0         0         0   0.09073   0.05702  -0.06099    -16.18         0

  B =
       u1
   x1   0
   x2   0
   x3   0
   x4   1
   x5   0
   x6   0
   x7   0
   x8   0

  C =
           x1       x2       x3       x4       x5       x6       x7       x8
   y1  -0.0224     12.2   -23.26        0        0        0        0        0

  D =
       u1
   y1   0

Continuous-time state-space model.


G =

  A =
             x1        x2        x3        x4        x5        x6
   x1         0         0         0         0         0         1
   x2         0      -2.4    -0.763         0         0         0
   x3         0         1         0         0         0         0
   x4         0   0.05204   0.03271  -0.06099    -16.18         0
   x5         0         0         0         1         0         0
   x6         0   0.09073   0.05702  -0.06099    -16.18         0

  B =
       u1
   x1   0
   x2   1
   x3   0
   x4   0
   x5   0
   x6   0

  C =
       x1  x2  x3  x4  x5  x6
   y1   1   0   0   0   0   0

  D =
       u1
   y1   0

Continuous-time state-space model.
```

```
sys =

  A =
           x1        x2        x3        x4        x5        x6        x7
   x1       0         0         0         0         0         0         1
   x2    0.23   -0.7519         0         0         0         0         0
   x3       0         0      -2.4    -0.763         0         0         0
   x4       0         0         1         0         0         0         0
   x5       0         0   0.05204   0.03271  -0.06099    -16.18         0
   x6       0         0         0         0         1         0         0
   x7       0         0   0.09073   0.05702  -0.06099    -16.18         0

  B =
       u1
   x1   0
   x2   0
   x3   1
   x4   0
   x5   0
   x6   0
   x7   0

  C =
          x1      x2      x3      x4      x5      x6      x7
   y1   4.347  -13.46       0       0       0       0       0

  D =
       u1
   y1   0

Continuous-time state-space model.


ans =

   Inf


G =

  A =
           x1        x2        x3        x4        x5        x6
   x1       0         0         0         0         0         1
   x2       0      -2.4    -0.763         0         0         0
   x3       0         1         0         0         0         0
   x4       0   0.05204   0.03271  -0.06099    -16.18         0
   x5       0         0         0         1         0         0
   x6       0   0.09073   0.05702  -0.06099    -16.18         0

  B =
       u1
   x1   0
   x2   1
   x3   0
   x4   0
   x5   0
   x6   0

  C =
       x1  x2  x3  x4  x5  x6
   y1   1   0   0   0   0   0

  D =
       u1
   y1   0

Continuous-time state-space model.


sys =

  A =
           x1        x2        x3        x4        x5        x6        x7        x8
   x1       0         0         0         0         0         0         0         1
   x2    0.23   -0.7519         0         0         0         0         0         0
   x3       0         0         0         0         0         0         0         0
   x4       0         0   0.01493      -2.4    -0.763         0         0         0
   x5       0         0         0         1         0         0         0         0
   x6       0         0         0   0.05204   0.03271  -0.06099    -16.18         0
   x7       0         0         0         0         0         1         0         0
   x8       0         0         0   0.09073   0.05702  -0.06099    -16.18         0

  B =
       u1
   x1   0
   x2   0
   x3   1
```

```
  x4   1
  x5   0
  x6   0
  x7   0
  x8   0

C =
         x1      x2      x3      x4      x5      x6      x7      x8
  y1  4.347  -13.46       0       0       0       0       0       0

D =
      u1
  y1   0

Continuous-time state-space model.
```

REFERENCES

[1] B. Friedland, Observer-Based Control System Design Lecture Notes for ECE660.

[2] B. Friedland, Control System Design: An Introduction to State Space Methods, McGraw-Hill, 1985. ISBN:0070224412 (Reprinted by Dover Publications May 2005, ISBN: 0-486-44278-0.)

[3] M. Hobbs, Basics of Missile Guidance and Space Techniques, v. 1, ISBN:9781434421258 (Wildside Press, LLC, 2010)

[4] A. Nebylov, Aerospace Sensors, 2012, ISBN: 9781606500613 (Momentum Press, 2012)

[5] A. Gelb, Applied Optimal Estimation, ISBN-13: 978-0262570480 (M.I.T. Press, 1974)

[6] NASA/ESA, the DSS-II and GSC-II Consortia (with images from the 'Palomar Observatory-STScI Digital Sky Survey of the northern sky, based on scans of the Second Palomar Sky Survey are copyright) 1993-1999 by the California Institute of Technology,
URL=https://www.spacetelescope.org/images/opo0118a/

[7] L. Mazzini (2016) The Dynamics of the Flexible Satellite. In: Flexible Spacecraft Dynamics, Control and Guidance. Springer Aerospace Technology. Springer, Cham

[8] P. Lin & T. J. Kostas, (2007). Flexible Satellite Systems. 1 - 7. 10.1109/MILCOM.2007.4455053.

[9] M. Lehmann, TU Berlin, BEESAT 8
URL=https://www.raumfahrttechnik.tu-berlin.de/menue/forschung/aktuelle_projekte/beesat_5_6_7_8/

[10] Baumann, F .; Korn, N .; Pirschel, K .; Wolf, R .; Brie, K. (2017). A Picosatellite Swarm for Technology Demonstration (IAC-17-B4.6B.7). 68th International Astronautical Congress (IAC) , Adelaide (Australia), 25.-29. September.

[11] Korn, N .; Baumann, F .; Wolf, R .; Brie, K. (2017). Multifunctional Optical Attitude Determination Sensor for Picosatellites (IAC-17-C1.1.9). 68th International Astronautical Congress (IAC) , Adelaide (Australia), 25.-29. September.

[12] da Silva, Adenilson Roberto; Gadelha de Souza, Luiz Carlos, Control System and Flexible Satellite Interaction During Orbit Transfer Maneuver (AAS 98-343), Spaceflight Dynamics 1998, Volume 100 Part 1, Advances in Astronautical Sciences. Proceedings of the AAS/GSFC International Symposium on Space Flight Dynamics held 11-15 May, 1998, Greenbelt Maryland. Edited by Thomas H. Stengle. American Astronautical Society Publication, 1998., p.541

[13] TUB -ILR, DECAN Rocket Project,
URL=https://www.raumfahrttechnik.tu-berlin.de/menue/forschung/aktuelle_projekte/decan/