Demo 2 : Robust attitude control design of a spacecraft with 2 symmetrical solar arrays ($\approx 10\,mn$ run).

Table of Contents

by D. Alazard, F. Sanfedino - ISAE-SUPAERO and E. Kassarian - DyCSyT.

1. Description

The objective is to design the Attitude Control System of the spacecraft presented in the Demo1 : ***a spacecraft with 2 symmetrical solar arrays.***

The SIMULINK model is the same, except that 20% of relative uncertainty is taken into account on :

- the mass of the main body $\mathcal{B}$: **`mB,`**
- the 3 terms on the diagonal of the main body inertia: $\mathbf{J}_B^{\mathcal{B}}$: **`IBx, IBy, IBz,`**
- the frequency of the 3 flexible modes of the solar array (identical for the 2 solar arrays): **`w1, w2, w3`**.

The mathematical framework used for the representation of parametric uncertainties is the Li<near Fractional Transformation (LFT).

```
SCwith2SAcantilevered_U
Gu=ulinearize('SCwith2SAcantilevered_U')
```

```
Gu =

  Uncertain continuous-time state-space model with 6 outputs, 6 inputs, 12 states.
  The model uncertainty consists of the following blocks:
    IBx: Uncertain real, nominal = 75, variability = [-20,20]%, 1 occurrences
    IBy: Uncertain real, nominal = 40, variability = [-20,20]%, 1 occurrences
    IBz: Uncertain real, nominal = 80, variability = [-20,20]%, 1 occurrences
    mB: Uncertain real, nominal = 1e+03, variability = [-20,20]%, 3 occurrences
    tan_Theta_div4: Uncertain real, nominal = 0, range = [-1,1], 32 occurrences
    w1: Uncertain real, nominal = 5.6, variability = [-20,20]%, 4 occurrences
    w2: Uncertain real, nominal = 19.3, variability = [-20,20]%, 4 occurrences
    w3: Uncertain real, nominal = 35.4, variability = [-20,20]%, 4 occurrences

Type "Gu.NominalValue" to see the nominal value, "get(Gu)" to see all properties, and "Gu.Uncertainty" to
```

This demo aims to show how the **`SIMULINK`** models made with **`SDTlib`** can be used for control design with the *robust control toolbox*. The numerical data are given as an example and are not necessarily representative of an actual application.

We consider the robust design of a 3- axis structured attitude control law to meet:

- (**Req1**) the pointing requirement (defined by the 3×1 vector **APE**, rad) in spite of low frequency orbital disturbances (characterized by the 3×1 upper bound on the magnitude **Tpert**, Nm),
- (**Req2**) stability margins characterized by an upper bound γ (**gamma**) on the H_∞ norm of the input sensitivity function.

while minimizing (**Req3**) the variance on the torque applied by the reaction wheel system on the spacecraft in response to the star sensor and gyrometer noises characterized by their PSD (Power Spectral Density **PSD_SST** and **PSD_GYRO** (assumed to be equal for the 3 components).

Numerical application:

```
% APE (Absolute Pointing Error) requirement
APE=[4 4 20]'*0.001*pi/180; % (rad)

% Orbital  disturbance magnitude:
Tpert=[0.03 0.01 0.02]'; %( Nm)

% PSD of sensor noises:
PSD_GYRO=1e-10;  %(rd^2/s),
PSD_SST=1e-8;    % (rd^2.s),

% Upper bound on the input sensitivity function
gamma=1.5;
```

The value $\gamma = 1.5$ ensures on each of the 3 axes:

- a modulus margin $> 1/\gamma = 0.6667$,
- a gain margin $> \frac{\gamma}{\gamma - 1} = 3\ (9.5\ dB)$,
- a phase margin $> 2 \arcsin \frac{1}{2\gamma} = 38.9\,(deg)$.

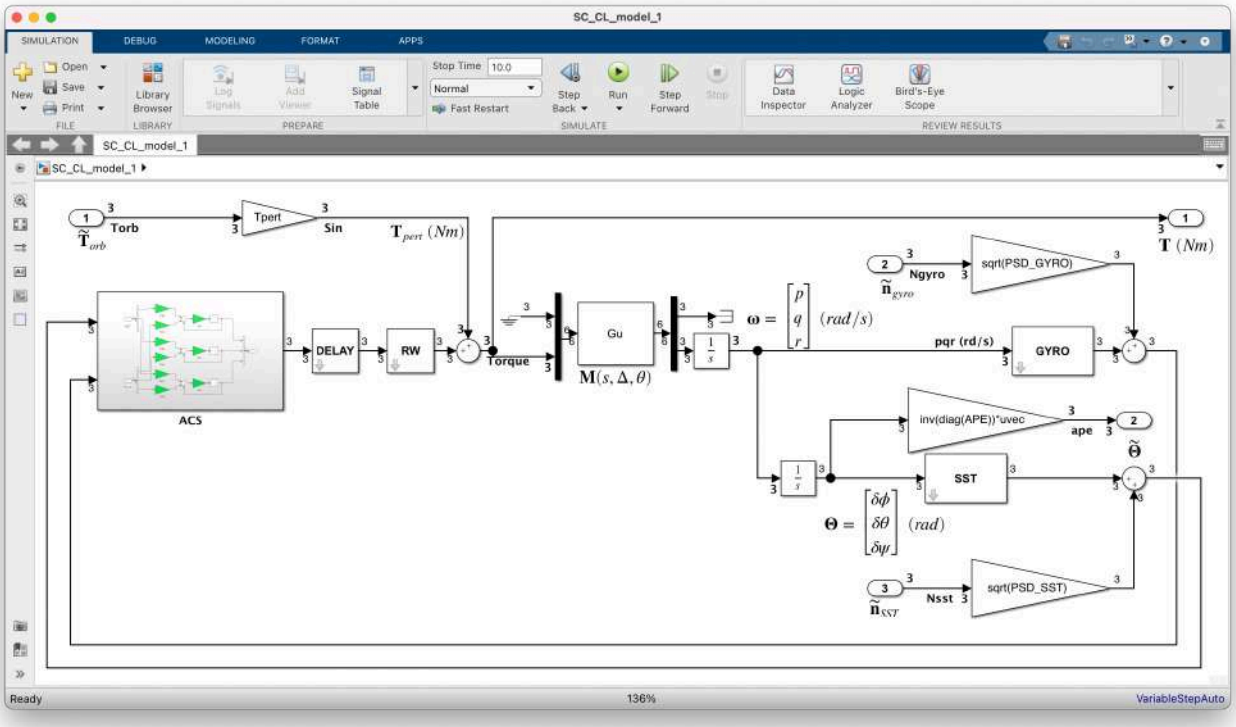
The requirements **Req1** and **Req2** must be met for any values of:

- the uncertain mechanical parameters (**mB, IPx, IBy, IBz w1, w2, w3)**, regrouped in a block $\boldsymbol{\Delta}$ in reference to the LFT formalism,
- the geometrical configuration θ of the solar arrays.

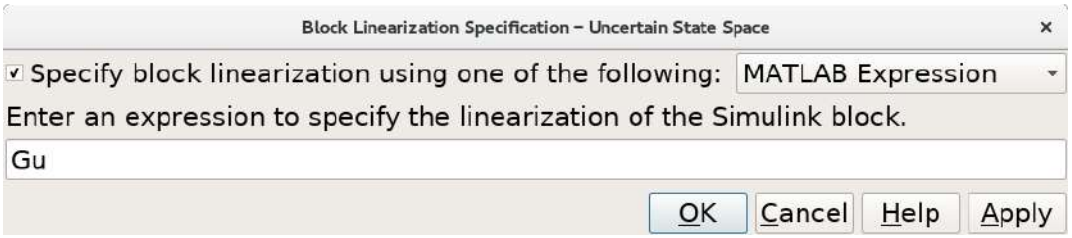
The performance index **Req3** is measured on the worst-case parametric configuration in $\boldsymbol{\Delta}$, θ.

2. The design model

The design model is the SIMULINK file **SC_CL_model_1.slx** depicted in the following Figure.



The model previously computed $\mathbf{M}(s, \mathbf{\Delta}, \theta)$ (**Gu**) is introduced with an **`Uncertain State Space`** from the **`Robust Control toolbox`** sub-library. Since the design will be based on the **`SIMULINK`** model and the **`Matlab/Simulink slTuner`** interface, it is very important to specify again **Gu** in the **`Linear Analysis > Specify Selected Block Linearization`** menu (obtained by a right click on the block `Gu`) according to the following Figure.



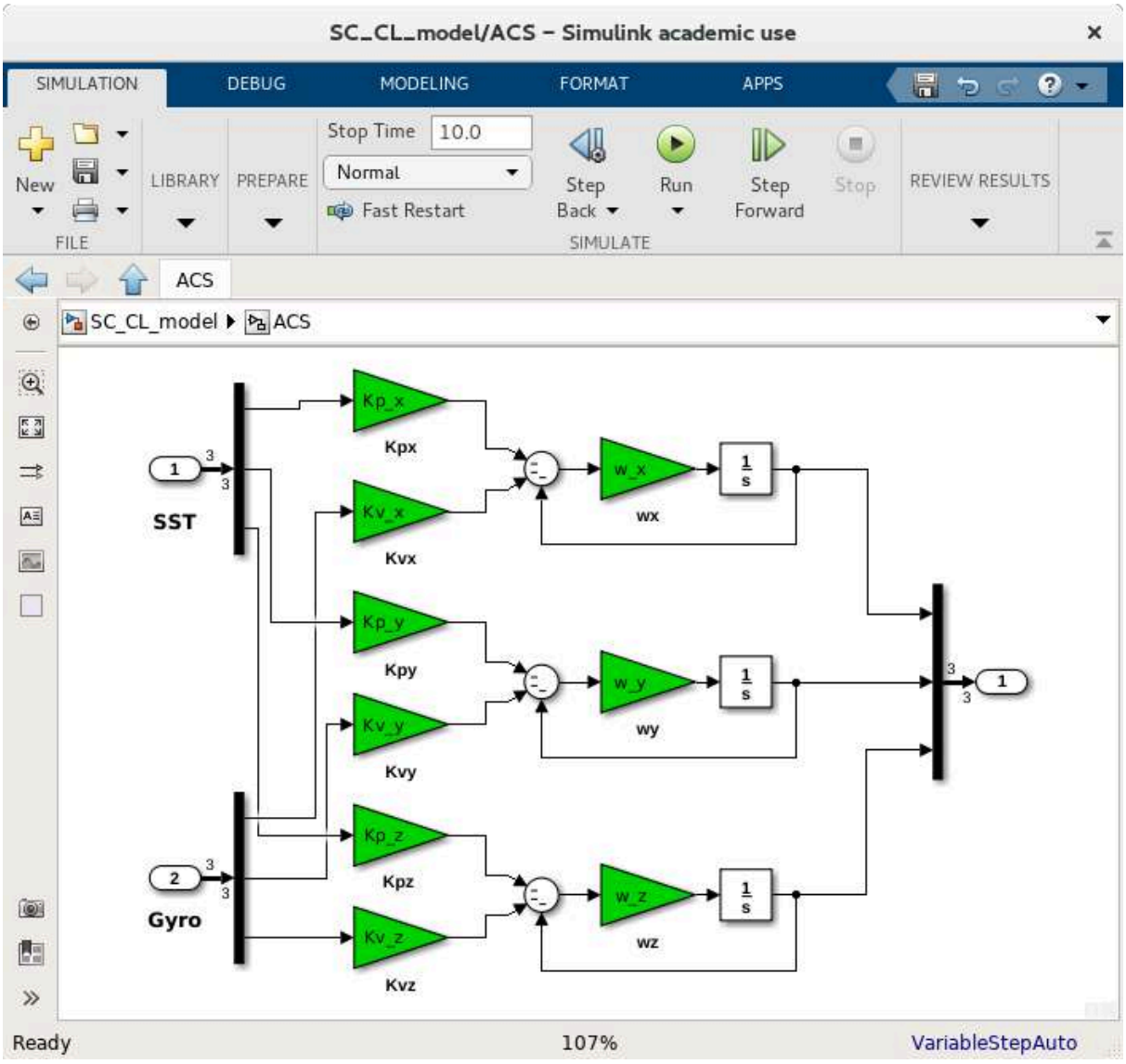
Then the design model is completed with the 3×3 avionics models for:

- the reaction wheel system dynamics: **`RW`**. A second order transfer with a 0.7 damping ratio and a $100\ Hz$ frequency is assumed on each of the 3 axes,
- the star sensor dynamics: **`SST`**. A first order transfer with a $8\ Hz$ cut-off frequency is assumed on each of the 3 axes,
- the gyrometer dynamics: **`GYRO`**. A first order transfer with a $200\ Hz$ cut-off frequency is assumed on each of the 3 axes,
- a loop delay **`DELAY`**. A $0.01\ s$ delay, modelled by a second order ***Pade*** approximation, is assumed on each of the 3 axes.

```
% Avionics:
```

```
% RWs: 2nd order low pass at 100 Hz
RW=tf((200*pi)^2,[1 1.4*200*pi (200*pi)^2])*eye(3);
% SST: 1st order low pass at 8 hz
SST=tf(8*2*pi,[1 8*2*pi])*eye(3);
% GYRO: 1sr order low pass at 200 Hz
GYRO=tf(400*pi,[1 400*pi])*eye(3);
% Loop delay
delay=0.01; %(s) : delay due to discretizaton, sensors dynamics, etc.
[num,den]=pade(delay,2);
DELAY=tf(num,den)*eye(3);
```

The structured 3×6 attitude control system (ACS) is depicted in the next figure.



It is a decentralised controller composed of a proportional-derivative controller (the 2 gains **`Kpi`** and **`Kvi`**, **`i=x,y,z`**) with a first order low pass filter (characterized by the cut-off frequency **`wi`**, **`i=x,y,z`**) per axis.

The set of the 9 tunable parameters (**`Kpi, wi`** and **`Kvi`**, **`i=x,y,z`**) is denoted $\mathbf{K}$. They are roughly initialized assuming a rigid 3-axis decoupled spacecraft and in order to:

- reject a constant orbital disturbance $\mathbf{T}_{pert}(i)$ with a steady state pointing error $\Delta\theta_i$ lower that the absolute pointing error requirement $\mathbf{APE}(i)$ on each axis $i = x, y, z$.
- tune the 2-nd order closed-loop dynamics of each axis with a damping ratio of $\xi = 0.7$ and a given frequency (bandwith) ω_i.

Indeed, under these assumptions, the open loop model between the control torque and the pointing error on each axis is $\frac{1}{\mathbf{J}_B^{\mathcal{S}/\mathcal{C}}(i,i)\mathrm{s}^2}$ where the nominal 3×3 inertia $\mathbf{J}_B^{\mathcal{S}/\mathcal{C}}$ (or **`IatG`** in the following **`MATLAB`** sequence) of the whole spacecraft at the point B can be computed from the model **`Gu`**:

```
% Tunable parameter initialization:
% Total mass on the nominal model:
Mtot=inv(dcgain(Gu.NominalValue));
IatG=Mtot(4:6,4:6);
```

Then the tuning $K_{p_i} = \mathbf{J}_B^{\mathcal{S}/\mathcal{C}}(i,i)\,\omega_i^2$ and $K_{v_i} = 2\,\xi\,\mathbf{J}_B^{\mathcal{S}/\mathcal{C}}(i,i)\,\omega_i$ ensures the wanted closed-loop dynamics and a distubance rejection function expressed as:

$$\frac{\Delta\theta_i}{\mathbf{T}_{pert}(i)}(\mathrm{s}) = \frac{1}{\mathbf{J}_B^{\mathcal{S}/\mathcal{C}}(i,i)(\mathrm{s}^2 + 2\,\xi\,\omega_i\mathrm{s} + \omega_i^2)}.$$

Thus the minimal bandwith required to meet the absolute pointing error requirement in steady state ($\Delta\theta_i \leq \mathbf{APE}(i)$) is $\omega_i = \sqrt{\frac{\mathbf{T}_{pert}(i)}{\mathbf{J}_B^{\mathcal{S}/\mathcal{C}}(i,i)\mathbf{APE}(i)}}$.

The frequency of the first order low pass filter is tuned at $20\omega_i$ on each axis.

This initial tuning, based on simplified assumptions (it will be shown that it does not verify the requirement **`Req2`** and does not minimize **`Req3`**), is useful to initialize the non-convex optimization problem presented in next section. It can be implemented according to the following **`MATLAB`** sequence:

```
% Required bandwidth on each axis:
wACS=sqrt(Tpert./(diag(IatG).*APE));
% with a damping ratio:
xiACS=0.7;
% Tunable parameter initialization:
Kp_x=IatG(1,1)*wACS(1)^2;
Kv_x=IatG(1,1)*2*xiACS*wACS(1);
Kp_y=IatG(2,2)*wACS(2)^2;
Kv_y=IatG(2,2)*2*xiACS*wACS(2);
Kp_z=IatG(3,3)*wACS(3)^2;
Kv_z=IatG(3,3)*2*xiACS*wACS(3);
w_x=20*wACS(1);  w_y=20*wACS(2);  w_z=20*wACS(3);
```

The inputs and outputs of the closed-loop SIMULINK model **`SC_CL_model_1.slx`**, denoted $\mathbf{P}(\mathrm{s}, \mathbf{\Delta}, \theta, \mathbf{K})$, are:

- $\widetilde{\mathbf{T}}_{orb}$ (**Torb**: 3 component input # 1): the normalized (w.r.t. **Tpert**) orbital disturbance (adimensional),
- $\widetilde{\mathbf{n}}_{gyro}$ (**Ngyro**: 3 component input # 4): the normalized (w.r.t. **PSD_GYRO**) gyro noise ($\sqrt{Hz}$),
- $\widetilde{\mathbf{n}}_{SST}$ (**Nsst**: 3 component input # 5): the normalized (w.r.t. **PSD_SST**) star sensor noise ($\sqrt{Hz}$),
- $\mathbf{T}$ (**Torque**: 3 component output # 1): the torque applied on the spacecraft by the reactions wheels (Nm),
- $\widetilde{\mathbf{\Theta}}$ (**ape**: 3 component output # 2): the normalized (w.r.t. **APE**) attitude error (adimensional).

$\mathbf{P}_{\widetilde{\mathbf{T}}_{orb}\to\widetilde{\mathbf{\Theta}}}(\mathrm{s},\mathbf{\Delta},\theta,\mathbf{K})$ denotes the closed-loop transfer from $\widetilde{\mathbf{T}}_{orb}$ to $\widetilde{\mathbf{\Theta}}$ for a given parametric uncertainty vector $\mathbf{\Delta}$, a given geometric configuration θ and a given tunable parameter vector $\mathbf{K}$. $\mathbf{P}_{\widetilde{\mathbf{n}}\to\mathbf{T}}(\mathrm{s},\mathbf{\Delta},\theta,\mathbf{K})$ denotes the closed-loop transfer from the 6 component noise $\widetilde{\mathbf{n}}=[\widetilde{\mathbf{n}}_{gyro}^{T}\quad\widetilde{\mathbf{n}}_{SST}]^{T}$ to $\mathbf{T}$.

In addition, the SIMULINK model has some labelled internal signals: $\mathbf{T}_{pert}\ (Nm)$ (**Sin**) is the input disturbance used to define the input sensitivity function $\bar{\mathbf{S}}(\mathrm{s},\mathbf{\Delta},\theta,\mathbf{K})=\mathbf{P}_{\mathbf{T}_{pert}\to\mathbf{T}}(\mathrm{s},\mathbf{\Delta},\theta,\mathbf{K})$.

3. Robust control design

The design is made using the **MATLAB/SIMULINK** interface **slTuner** by specifying the names of the tunable blocks and the names of the signals used to express the various requirements:

```
SC_CL_model_1
P0=slTuner('SC_CL_model_1',{'Kpx','Kvx','wx','Kpy','Kvy','wy','Kpz','Kvz','wz'});
addPoint(P0,{'Torb','Sin','Ngyro','Nsst','Torque','ape'});
```

The first requirement or hard contraint **Req1** on the pointing error reads:

$$\max_{\mathbf{\Delta},\theta}\|\mathbf{P}_{\widetilde{\mathbf{T}}_{orb}\to\widetilde{\mathbf{\Theta}}}(\mathrm{s},\mathbf{\Delta},\theta,\widehat{\mathbf{K}})\|_{\infty}\le 1$$

and the corresponding MATLAB syntax is:

```
% Hard requirement on the closed-loop transfert:
%  The gain from the normalized orbital disturbances to normalized APE:
Req1=TuningGoal.Gain('Torb','ape',1);
Req1.Name='ape/Spec';
```

The second requirement or hard constraint **Req2** on the input sensitivity function reads:

$$\max_{\mathbf{\Delta},\theta}\|\bar{\mathbf{S}}(\mathrm{s},\mathbf{\Delta},\theta,\widehat{\mathbf{K}})\|_{\infty}\le\gamma$$

and the corresponding MATLAB syntax is:

```
% Hard requirement on the input sensitivity function:
Req2=TuningGoal.Gain('Sin','Torque',gamma);
```

```
Req2.Name='Input sensitivity';
```

The performance index or soft constraint **Req3** is:

$$\widehat{J} = \max_{\boldsymbol{\Delta},\theta} \|\mathbf{P}_{\tilde{\mathbf{n}}\to\mathbf{T}}(s,\boldsymbol{\Delta},\theta,\widehat{\mathbf{K}})\|_2 \text{ with } \widehat{\mathbf{K}} = \arg\min_{\mathbf{K}} \max_{\boldsymbol{\Delta},\theta} \|\mathbf{P}_{\tilde{\mathbf{n}}\to\mathbf{T}}(s,\boldsymbol{\Delta},\theta,\mathbf{K})\|_2$$

and the corresponding MATLAB syntax is:

```
% Soft requirement on the variance from sensor noises to
% the applied torque:
Req3=TuningGoal.Variance({'Ngyro','Nsst'},'Torque',1);
Req3.Name='Actuator Variance';
```

The optimization is done thanks to the function **systune**:

(Remark: if you do not wish to run systune, please load the variable OptimalDesign instead, which contains the outputs that we obtained when running systune. The computation of systune can take several minutes.)

```
% Design optimisation:
rng(1) % to freeze the random number sequence from one run to the next.
[Popt,fBest,gBest,Info] = systune(P0,Req3,[Req1,Req2]);
```

```
Soft: [0.106,Inf], Hard: [1,1.62], Iterations = 135
Soft: [0.11,Inf], Hard: [1,1.16], Iterations = 128
Soft: [0.113,Inf], Hard: [1,1.09], Iterations = 129
Soft: [0.121,Inf], Hard: [1,1.01], Iterations = 119
Soft: [0.122,Inf], Hard: [1,1], Iterations = 117
Soft: [0.122,0.122], Hard: [1,1], Iterations = 131
Soft: [0.122,0.122], Hard: [1,1], Iterations = 162
Final: Soft = 0.122, Hard = 0.99975, Iterations = 921
```

```
gBest
```

```
gBest = 1×2
    0.9998    0.9997
```

Since the 2 components of **gBest** are lower than 1, the requirements **Req1** and **Req2** are met for any parametric uncertainties $\boldsymbol{\Delta}$ and any geometric configurations θ (or at least according to the worst-case parametric and geometric configurations found by **systune**).

```
fBest
```

```
fBest = 0.1222
```

fBest gives the worst-case performance index: $\widehat{J} = \text{fBest}\ Nm/\sqrt{Hz}$

Then, one can get the optimal tuning of the 9 controller gains $\widehat{\mathbf{K}}$:

```
% Optimal tuning:
w_x=getBlockValue(Popt,'wx');w_x=w_x.d;
w_y=getBlockValue(Popt,'wy');w_y=w_y.d;
w_z=getBlockValue(Popt,'wz');w_z=w_z.d;
Kp_x=getBlockValue(Popt,'Kpx');Kp_x=Kp_x.d;
Kp_y=getBlockValue(Popt,'Kpy');Kp_y=Kp_y.d;
```

```
Kp_z=getBlockValue(Popt,'Kpz');Kp_z=Kp_z.d;
Kv_x=getBlockValue(Popt,'Kvx');Kv_x=Kv_x.d;
Kv_y=getBlockValue(Popt,'Kvy');Kv_y=Kv_y.d;
Kv_z=getBlockValue(Popt,'Kvz');Kv_z=Kv_z.d;
```

4. Optimal design analysis

It is now possible to analyse the frequency-domain responses of 2 constrained transfers (**Req1** and **Req2**) from random samples in the parametric space $\{\mathbf{\Delta}, \theta\}$ and the worst-case identified during the optimization process (**systune**):

```
% Optimal design analysis:
PB1=getIOTransfer(Popt,'Torb','ape');
PB2=getIOTransfer(Popt,'Sin','Torque');

% Worst-cases identified by systune
% Best run (if the option 'randomestarts' is used):
[g,index]=min([Info.g]);
% Worst cases identified during the best run:
WCs=Info(index).wcPert;
```

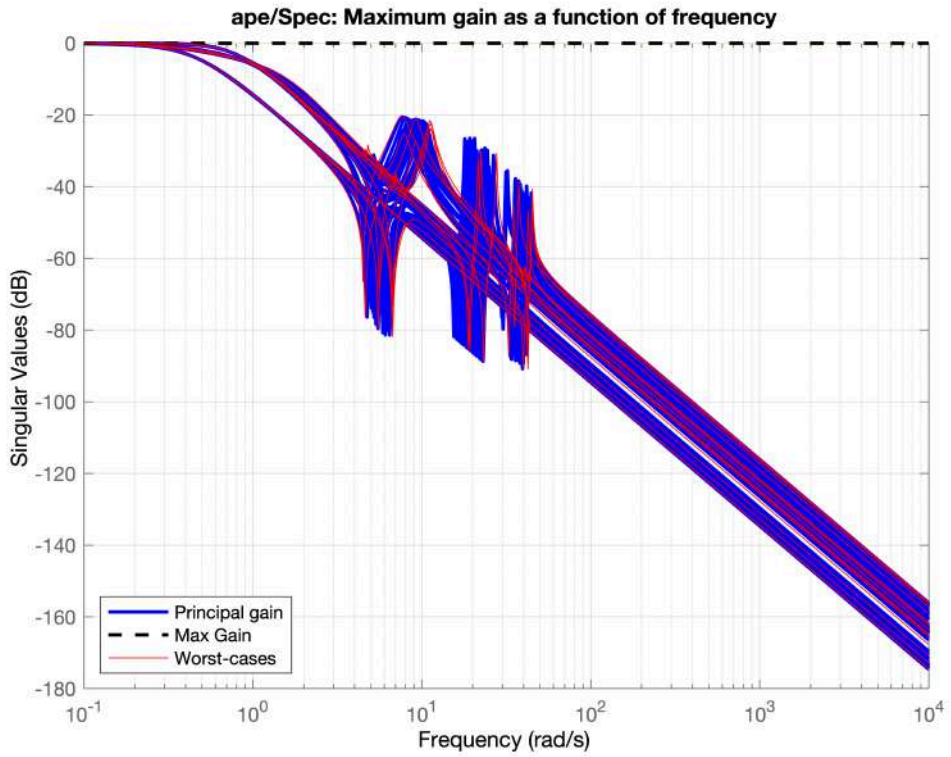
The routine systune identified 7 worst-case parametric configurations. For example, the first one is :

```
WCs(1)
```

```
ans = struct with fields:
               IBx: 60.0000
               IBy: 32
               IBz: 64
                mB: 800
    tan_Theta_div4: 1
                w1: 4.4800
                w2: 23.1600
                w3: 42.4800
```

The frequency domain response relative to **Req1** (absolute pointing error in response to the orbital disturbance) is plotted:

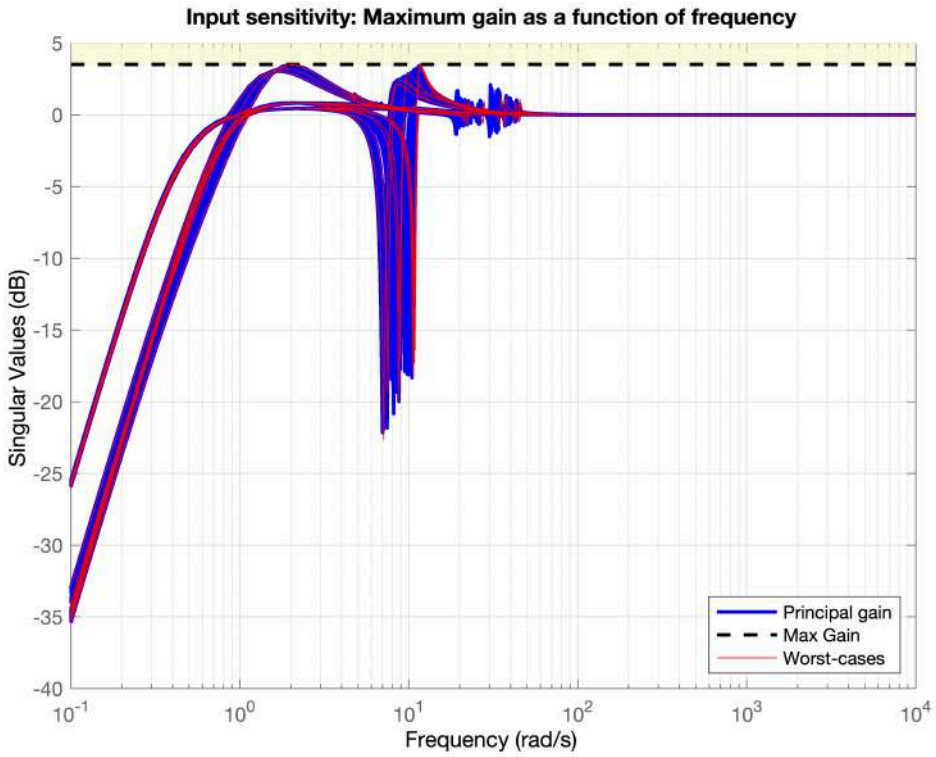
```
figure
figReq1=gcf;
viewGoal(Req1,PB1)
hold on
sigma(usubs(PB1,WCs),'-r')   % worst performance
legend('Principal gain','Max Gain','Worst-cases','Location','southwest')
```



The hard constraint **`Req1`**, representing the absolute pointing error in response to the orbital disturbance, is met over all parametric configurations detected by systune, and saturated at the null frequency for any values of the parametric vector and the geometric configuration.

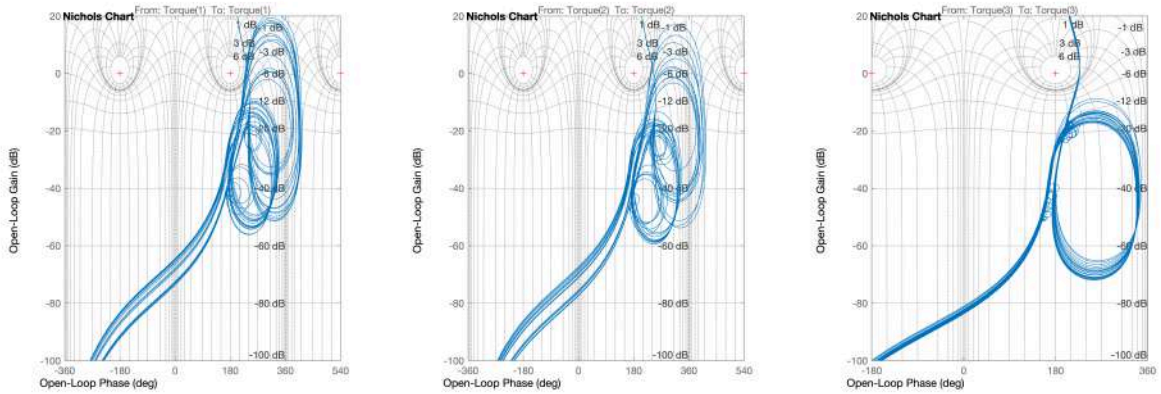
The frequency domain response relative to **`Req2`** (sensitivity function, determining the stability margins) is plotted:

```
figure
figReq2=gcf;
viewGoal(Req2,PB2)
hold on
sigma(usubs(PB2,WCs),'-r')  % worst performance:
legend('Principal gain','Max Gain','Worst-cases','Location','southeast')
```



The hard constraint **Req2**, representing the sensitivity function and imposing a lower bound on the disc margin, is met over all parametric configurations detected by systune, and saturated in low frequency (around 2 rad/s) and at the frequency of the first flexible mode (around 12 rad/s). Note the axis-per-axis stability margins (gain margins and phase margins) are largely better than the convervative lower bounds induced from **Req2** (gain margin $> \frac{\gamma}{\gamma-1} = 3\ (6\ dB)$, phase margin $> 2\arcsin\frac{1}{2\gamma} = 38.9\,deg$) , as shown on the following Nichols frequency-domain responses of the open-loop transfer function (when the loop is opened on the system input):

```
% Nichols plot of the open-loop transfer function:
% the loop is opened at the system input:
L = -getLoopTransfer(Popt,'Torque') ;
figure
for ii=1:3
    FDBCK=eye(3);
    FDBCK(ii,ii)=0;           % only the loop on channel # ii is opened!
    BFii=feedback(L,FDBCK);
    subplot(1,3,ii);
    nichols(BFii(ii,ii));
    ngrid
    set(gca,'Ylim',[-100 20]);
end
set(gcf,'Units','normalized','Position',[0 0 1 0.5])
```



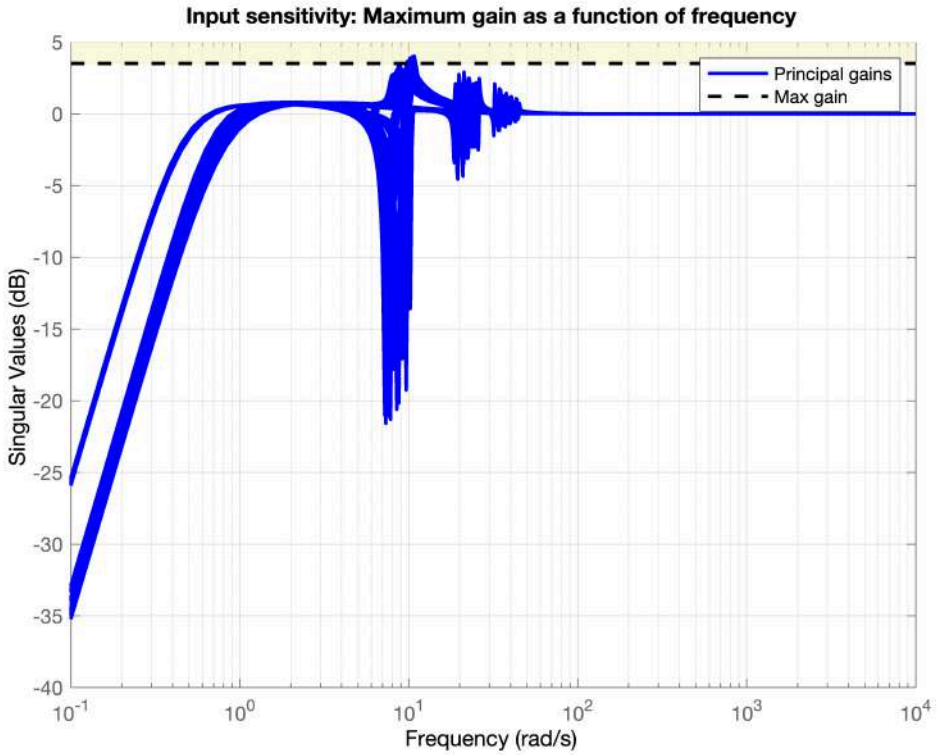
Finally, the performance index **`Req3`** (variance of the actuator's torque in response to the measurement noise), measured by `fBest`, can also be retrieved as the H2 norm of the transfer:

```
max(norm(usubs(getIOTransfer(Popt,{'Ngyro','Nsst'}, 'Torque'), WCs), 2))
```

```
ans = 0.1222
```

Remark: the controller computed in section *2* and which was used to initialize the optimization problem, does not robustly verify **`Req2`** and yields a slightly worse **`Req3`**:

```
figure
viewGoal(Req2,P0)
```



```
max(norm(usubs(getIOTransfer(P0,{'Ngyro','Nsst'}, 'Torque'), WCs), 2))%%%%%%%%%%%%%%%%%
```

```
ans = 0.1274
```

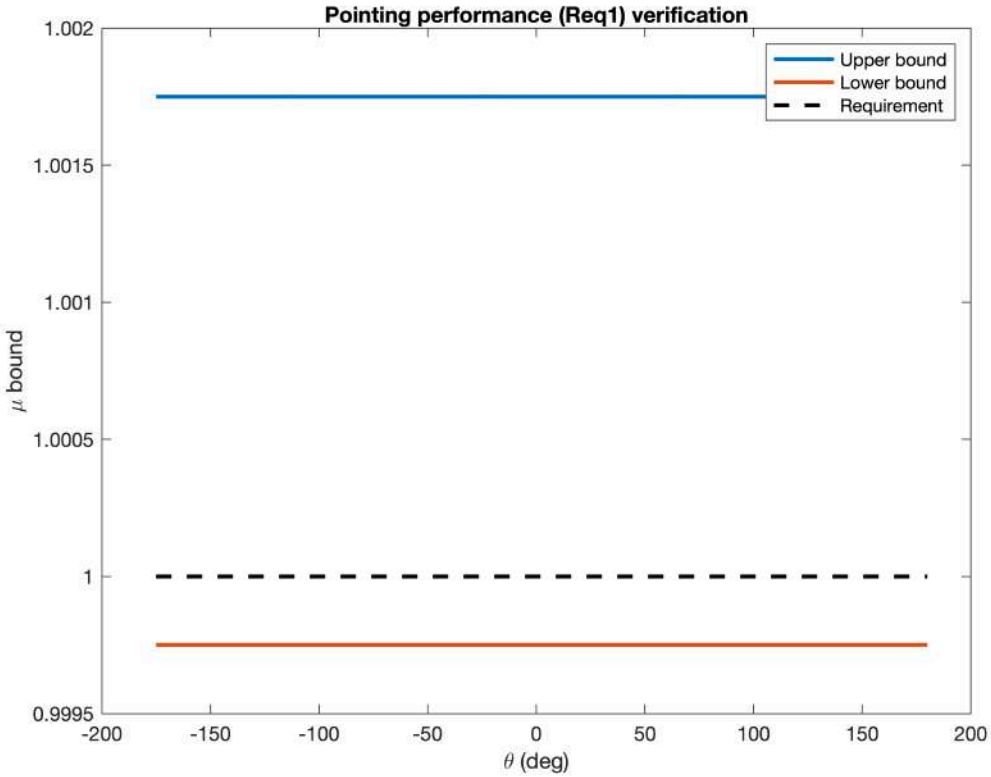
5. V&V (Verification and Validation)

When a robust performance problem is submitted to **systune**, it is solved thanks to an efficient heuristic allowing a set of worst-case models to be selected and used in a multi-model control design approach. Nevertheless, a μ-analysis can be directly applied on the parametrized model **Gu** to give a performance guarantee certificate. While previous section analyzed the robustness based on a sampling of the parametric space helped with some worst-case configurations detected by **systune**, the results provided by the μ-analysis guarantee that no worst-case configuration is missed over the whole parametric space. All the robustness and worst-case analysis tools provided by the **MATLAB Robust Control Toolbox (mussv, robstab, robgain, wcgain, ...)** can be direcly applied on the models provided by the **SDTlib** thanks to the LFT (Linear Fractional Transformation) formalism used to represent these models.

Although this example is quite simple (7 uncertain parameters and 1 varying parameter θ), the high repetition (32 occurrences) of θ makes the μ-analysis quite challenging. In the following **MATLAB** sequences, θ is sampled over a grid with 72 points (every 5 degrees between -175 and 180) and the 2 hard requirements are verified over the whole parametric space of the 7 other uncertain parameters. The **MATLAB** code is commented because its run is about 45 minutes. The results of the analyses are saved in the data file **Mu_analysis_data.mat**

Worst-case pointing performance:

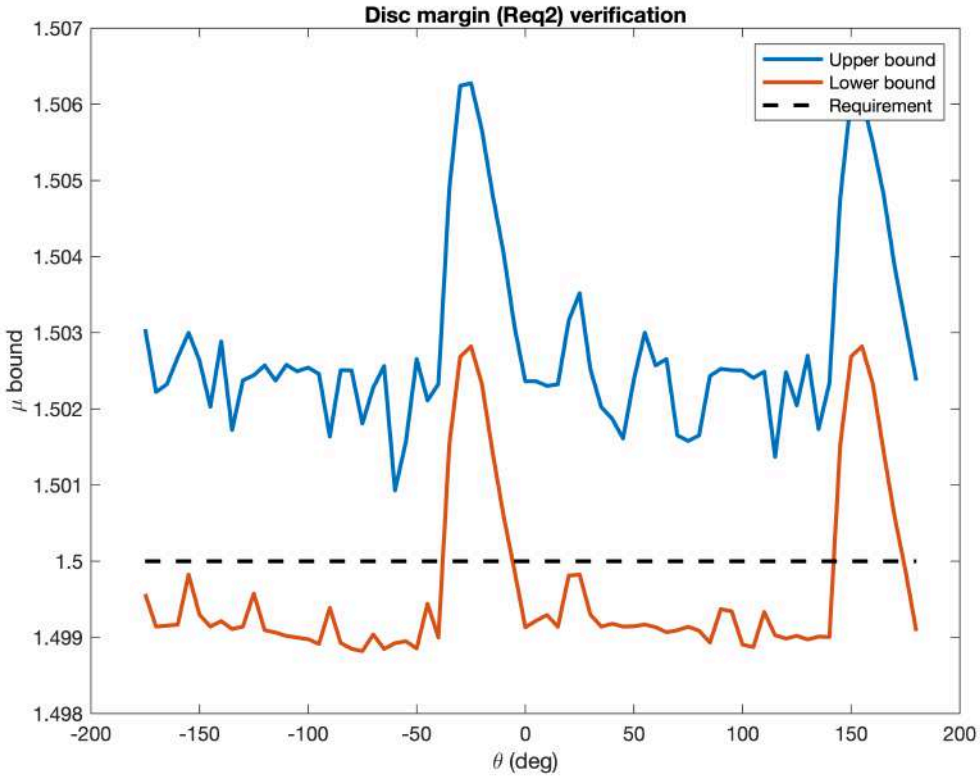
```
%OPT = wcOptions;
%OPT.MussvOptions='a';
%ii=0;
theta_grid=[-175:5:180];
% for theta=theta_grid;
%     ii=ii+1;
%     PB1ii=usubs(uss(PB1),'tan_Theta_div4',tan(theta*pi/180/4));
%     [WCG1(ii),WCU1(ii)]=wcgain(PB1ii,OPT);
% end
% save Mu_analysis_data WCG1 WCU1
load Mu_analysis_data
figure
plot(theta_grid,[WCG1(:).UpperBound],'Linewidth',2);
hold on
plot(theta_grid,[WCG1(:).LowerBound],'Linewidth',2);
plot(theta_grid,1*ones(size(theta_grid)),'k--','Linewidth',2)
xlabel('\theta (deg)')
ylabel('\mu bound')
legend('Upper bound', 'Lower bound', 'Requirement')
title('Pointing performance (Req1) verification')
```



The μ bounds are independent of the solar array configuration θ . The μ upperbound is just above 1 but no worst-case parametric configuration violating the pointing requirement was isolated (the μ lower bound is always under 1). On the frequency-response of the **ape/Spec**, it can be also verified that the pointing requirement is actually saturated in low frequency for all parametric configurations. The next analysis result on the disc margin is more interesting.

Worst-case disc margin:

```
% for theta=theta_grid;
%     ii=ii+1;
%     PB2ii=usubs(uss(PB2),'tan_Theta_div4',tan(theta*pi/180/4));
%     [WCG2(ii),WCU2(ii)]=wcgain(PB2ii,OPT);
% end
% save Mu_analysis_data WCG2 WCU2 WCG1 WCU1
load Mu_analysis_data
figure
plot(theta_grid,[WCG2(:).UpperBound],'Linewidth',2);
hold on
plot(theta_grid,[WCG2(:).LowerBound],'Linewidth',2);
plot(theta_grid,1.5*ones(size(theta_grid)),'k--','Linewidth',2)
xlabel('\theta (deg)')
ylabel('\mu bound')
legend('Upper bound', 'Lower bound', 'Requirement')
title('Disc margin (Req2) verification')
```



Here we can conclude that some worst-case parametric configurations violating the disc margin requirement were isolated around $\theta = -25\,deg$ and $\theta = 155\,deg$. However the disc margin guarrenteed by the upper bound is very close to the requirement and highlights that these worst-cases are only marginal.

The worst case configuration can be characterized

```
[wc,index]=max([WCG2(:).LowerBound]);
WCU2(index)
```

```
ans = struct with fields:
    IBx: 60.0000
    IBy: 32
    IBz: 64
     mB: 808.3372
     w1: 6.7200
     w2: 15.5501
     w3: 42.4800
```

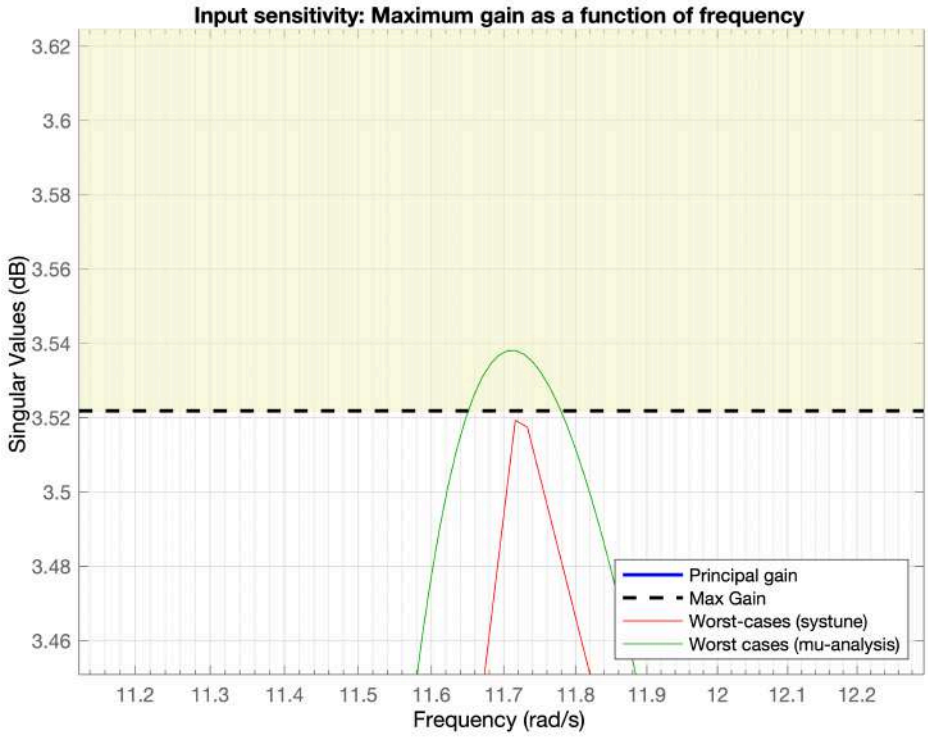
```
theta_grid(index)
```

```
ans = 155
```

and analyzed in the frequency domain:

```
figure(figReq2)
WCplant=usubs(PB2,WCU2(index));
WCplant=usubs(WCplant,'tan_Theta_div4',tan(theta_grid(index)*pi/180/4));
sigma(WCplant,...
    linspace(WCG2(index).CriticalFrequency*0.9,WCG2(index).CriticalFrequency*1.1,200),'
```

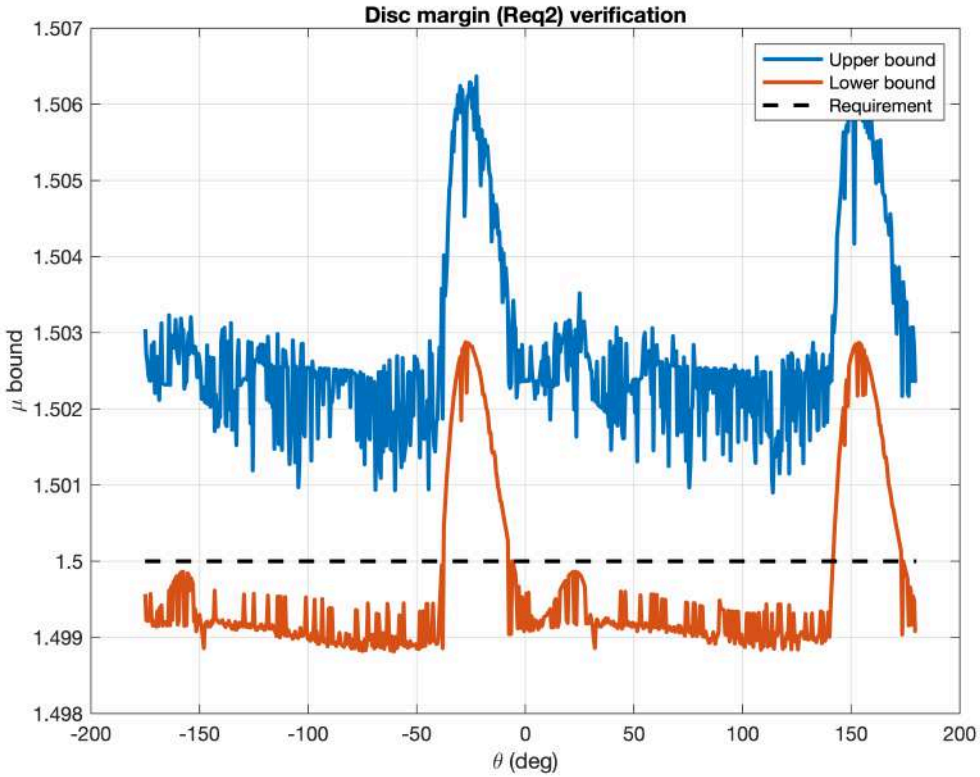
```
legend('Principal gain','Max Gain','Worst-cases (systune)',...
    'Worst cases (mu-analysis)','Location','southeast')
axis([WCG2(index).CriticalFrequency*0.95,WCG2(index).CriticalFrequency*1.05,...
    20*log10(WCG2(index).LowerBound*0.99),20*log10(WCG2(index).LowerBound*1.01)])
```



This Figure focuses the frequency domain response of the worst case identified by the analysis around the critical frequency. The worst-case configuration (with regard to the stability margin requirement **`Req2`**) is reached when the mass/inertia of the main body are close to their lower bounds (this is quite logic).

The grid of θ could be refined for even more precise results, and the computation time would only increase linearly with the size of the grid (since there is only one parameter that is gridded). For example, the figure **`Mu_req2_finer.fig`** was computed with a finner grid on θ (sampled every degree), and took 8 hours of computation (10 times longer). Let us note that a Monte-Carlo analysis would require to refine the grid over all 8 parameters, resulting in a much higher computational cost.

```
open('Mu_req2_finer.fig')
```



6. Comparison with a non robust synthesis

One main interest of the parametric models derived with the **SDTlib** is to perform directly control design with robust performances thanks to **systune**, as presented in section 3.

The following **Matlab** sequence computes a controller on the nominal model $\mathbf{M}(s,\mathbf{0},0)$. Then, analyses are performed *a posteriori* to validate it on the uncertain and varying model $\mathbf{M}(s,\boldsymbol{\Delta},\theta)$ (**Gu**). It is shown that the requirements are not robustly met.

Controller design on the nominal model with the set of requirements:

```
Gu_save=Gu;
Gu=Gu.NominalValue;
Kp_x=IatG(1,1)*wACS(1)^2;
Kv_x=IatG(1,1)*2*xiACS*wACS(1);
Kp_y=IatG(2,2)*wACS(2)^2;
Kv_y=IatG(2,2)*2*xiACS*wACS(2);
Kp_z=IatG(3,3)*wACS(3)^2;
Kv_z=IatG(3,3)*2*xiACS*wACS(3);
w_x=20*wACS(1);  w_y=20*wACS(2);  w_z=20*wACS(3);
P0=slTuner('SC_CL_model_1',{'Kpx','Kvx','wx','Kpy','Kvy','wy','Kpz','Kvz','wz'});
addPoint(P0,{'Torb','Sin','Ngyro','Nsst','Torque','ape'});
[Popt,fBest,gBest,Info] = systune(P0,Req3,[Req1,Req2]);
```

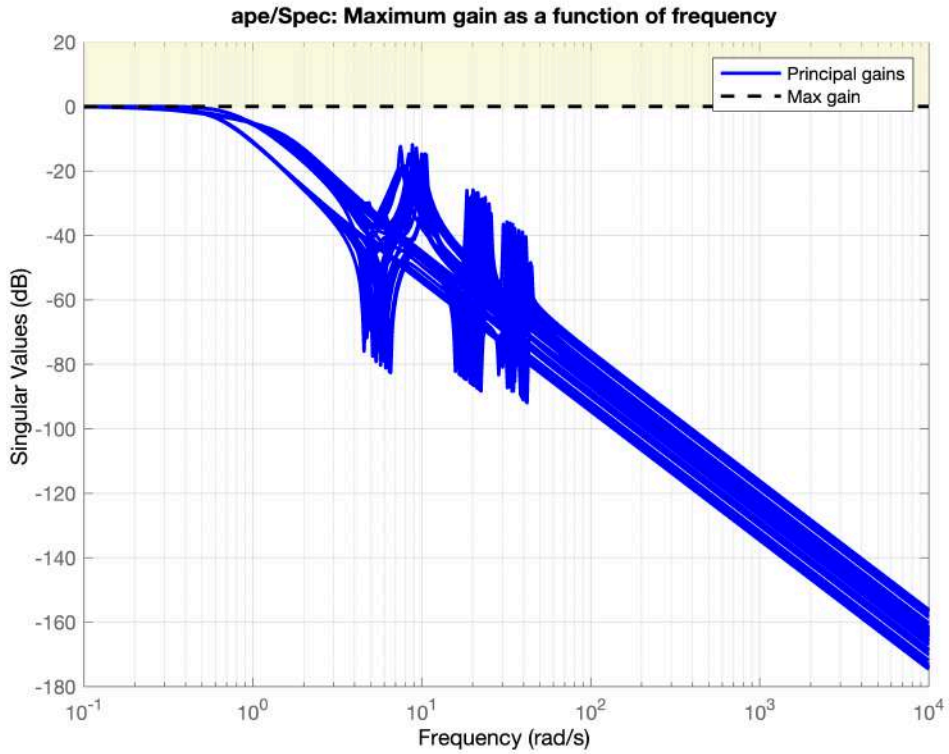
```
Final: Soft = 0.106, Hard = 0.99983, Iterations = 135
```

```
% Optimal tuning:
```

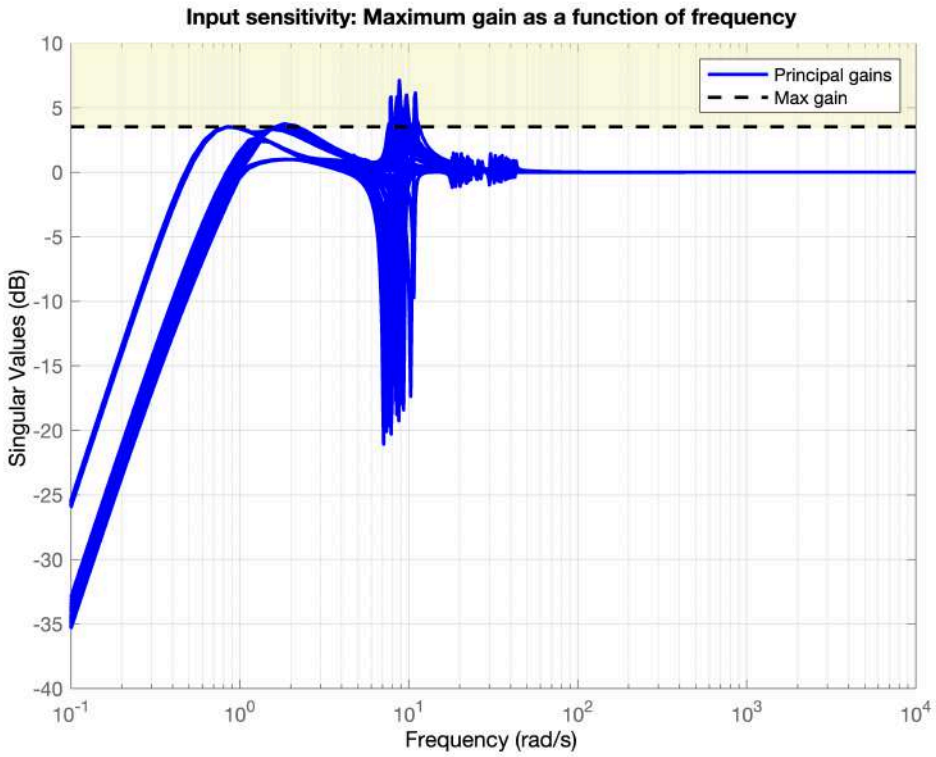
```
w_x=getBlockValue(Popt,'wx');w_x=w_x.d;
w_y=getBlockValue(Popt,'wy');w_y=w_y.d;
w_z=getBlockValue(Popt,'wz');w_z=w_z.d;
Kp_x=getBlockValue(Popt,'Kpx');Kp_x=Kp_x.d;
Kp_y=getBlockValue(Popt,'Kpy');Kp_y=Kp_y.d;
Kp_z=getBlockValue(Popt,'Kpz');Kp_z=Kp_z.d;
Kv_x=getBlockValue(Popt,'Kvx');Kv_x=Kv_x.d;
Kv_y=getBlockValue(Popt,'Kvy');Kv_y=Kv_y.d;
Kv_z=getBlockValue(Popt,'Kvz');Kv_z=Kv_z.d;
```

Validation on the uncertain and varying model Gu:

```
Gu=Gu_save;
Pval=slTuner('SC_CL_model_1',{'Kpx','Kvx','wx','Kpy','Kvy','wy','Kpz','Kvz','wz'});
addPoint(Pval,{'Torb','Sin','Ngyro','Nsst','Torque','ape'});
PB1=getIOTransfer(Pval,'Torb','ape');
PB2=getIOTransfer(Pval,'Sin','Torque');
figure
viewGoal(Req1,PB1)
```




```
figure
viewGoal(Req2,PB2)
```



These analyses allow to conclude that, when designing the controller on the nominal plant:

- the pointing error requirement is still (almost) met when adding parametric uncertainties,
- but the disc margin requirement is largely exceeded when adding parametric uncertainties.

Thus, a robust control design approach is strongly recommended on this type of problems, and therefore, parametric models are needed as early as possible in the preliminary design process. It is the main goal of SDTlib to provide such models dependent on uncertain or variable parameters.

7. Summary

In summary, the following points were addressed :

- ***Modeling:*** A complete dynamical model of the spacecraft was proposed. The model was easily computed by assembling the individual blocks representing the different elements of the spacecraft. Actuators dynamics, and sensors noise and delays were also implemented. Moreover, all parametric configurations of the uncertain parameters (mechanical parameters) and variable parameters (angular configuration of the solar arrays) are regrouped in one single model, based on the LFT framework. This parametric model enables the tools from the *robust control toolbox*, as follows.

- ***Control architecture:*** A classical structure, based on proportionnal-derivative controllers with a low-pass filter, was fixed. An initial guess was proposed, which did not fulfill all the requirements but was helpful to initialize the controller design as a non-convex optimization problem.
- ***Controller design:*** The control problem was formulated as an H-infinity optimization problem, which was solved with the routine systune. It allowed to minimize the variance of the actuator in response to the measurement noise, while pushing the requirements in pointing performance and stability margins to the admissible limit. Formulating all these constraints in one single optimization problem allowed us to optimize the performance while taking full advantage of the system's capacity, without iterating on the controller design while verifying the requirements a posteriori, as it would be the case with more traditionnal methods.
- ***Robustness:*** Moreover, this optimization was carried out while dynamically detecting the worst-case parametric configurations using reliable heuristics. As a consequence, the requirements are fulfilled even in the worst-case configurations detected by systune, which gives very good robustness properties to the controller. We also showed that the same optimization problem, when solved on the nominal plant (not taking into account the parametric uncertainties), largely exceeded the stability margins requirement. Taking into account the parametric uncertainties directly in the controller design allowed us to avoid tedious iterations with the V&V process.
- ***Validation:*** Although the μ-analysis could not be directly applied over the whole parametric space because of the number of repetitions of the parameter θ, an analysis was performed over a grid of values of θ. For each value of the grid, the analysis was performed over the whole parametric space of the 7 other uncertain parameters (without any gridding approximation for these 7 parameters). The lower bound showed that the requirement was exceeded for some parametric configurations. However, the upper bound provided a formal guarantee (for each angular configuration of the grid) that this excess is only marginal. This analysis was performed in reasonable time (less than 1 hour); the grid of θ could be refined for even more precise results, and the computation time would only increase linearly with the size of the grid (since there is only one parameter that is gridded). Let us note that a Monte-Carlo analysis would require to refine the grid over all 8 parameters, resulting in a much higher computational cost.
- ***General methodology:*** This whole procedure can be done early in the project phase, allowing to finely model the pointing error budget and guarantee the performance requirements with an adequate controller. The procedure can easily be repeated in case of a design change or when the uncertainties related to the mechanical design are narrowed down as the project advances. Essentially, it reduces the design iterations to reduce the time and increase the reliability of the project.

Final remark: this demo file illustrated the capacity of the ***`SDTlib`*** *for the modeling, analysis and control of space systems in a relatively simple application case. More complexe scenarios are highlighted in the file* `GetStarted.mlx`*, including for example other sources of microvibrations (reaction wheel imbalances, solar array drive mechanism), control/structure co-design approaches, on-orbit servicing scenarios, or experimental validations.*