# *Advanced Aerospace Control*

## *Tuning of a Quadrotor Controller*

*Exam Project A.Y.2020*

*Gianluca Lardo 945857*
*Giovanni Maria Maneschi 953386*
*Sebastiano Vianello 952633*

POLITECNICO
MILANO 1863

DIPARTIMENTO DI
SCIENZE E TECNOLOGIE
AEROSPAZIALI

# Presentation Overview

# Premise: objectives and work procedure

The project, developed in Matlab, has pursued the following steps:

- Parameters setting, nominal plant, and dynamic model
- Evaluation of tuning possibilities
- Analysis on both nominal and uncertain system, achieving nominal performance, robust stability, and robust performance
    - Redesign process in order to achieve the requirements
- A final verification on the accuracy of the solution has been implemented

# 1. Nominal System Analysis

## 1.1 Nominal Lateral Dynamics

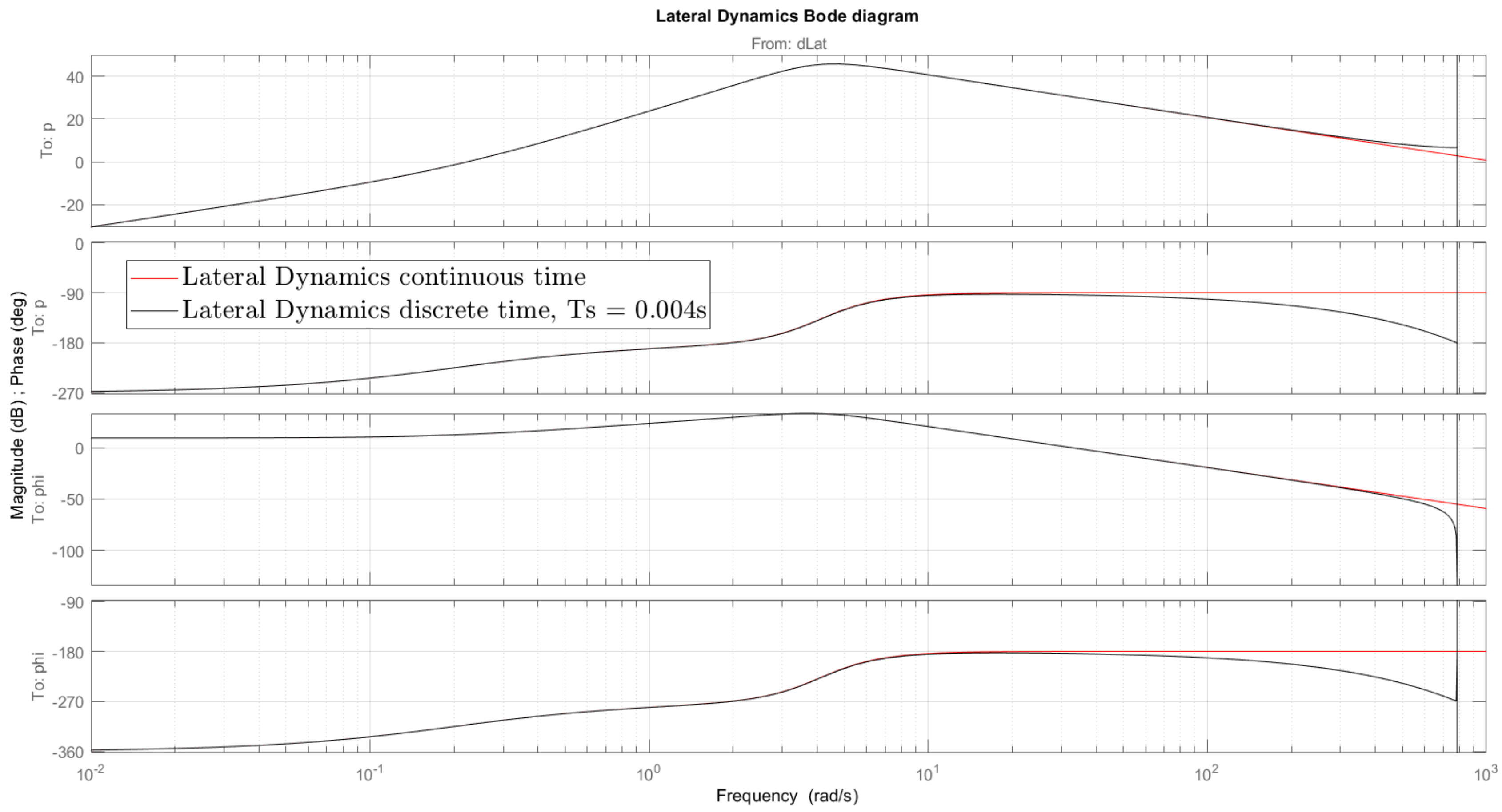
At first, the nominal model of the quadrotor (grey box model) has been obtained through experimental analysis.

The model itself is in state space form as follow:

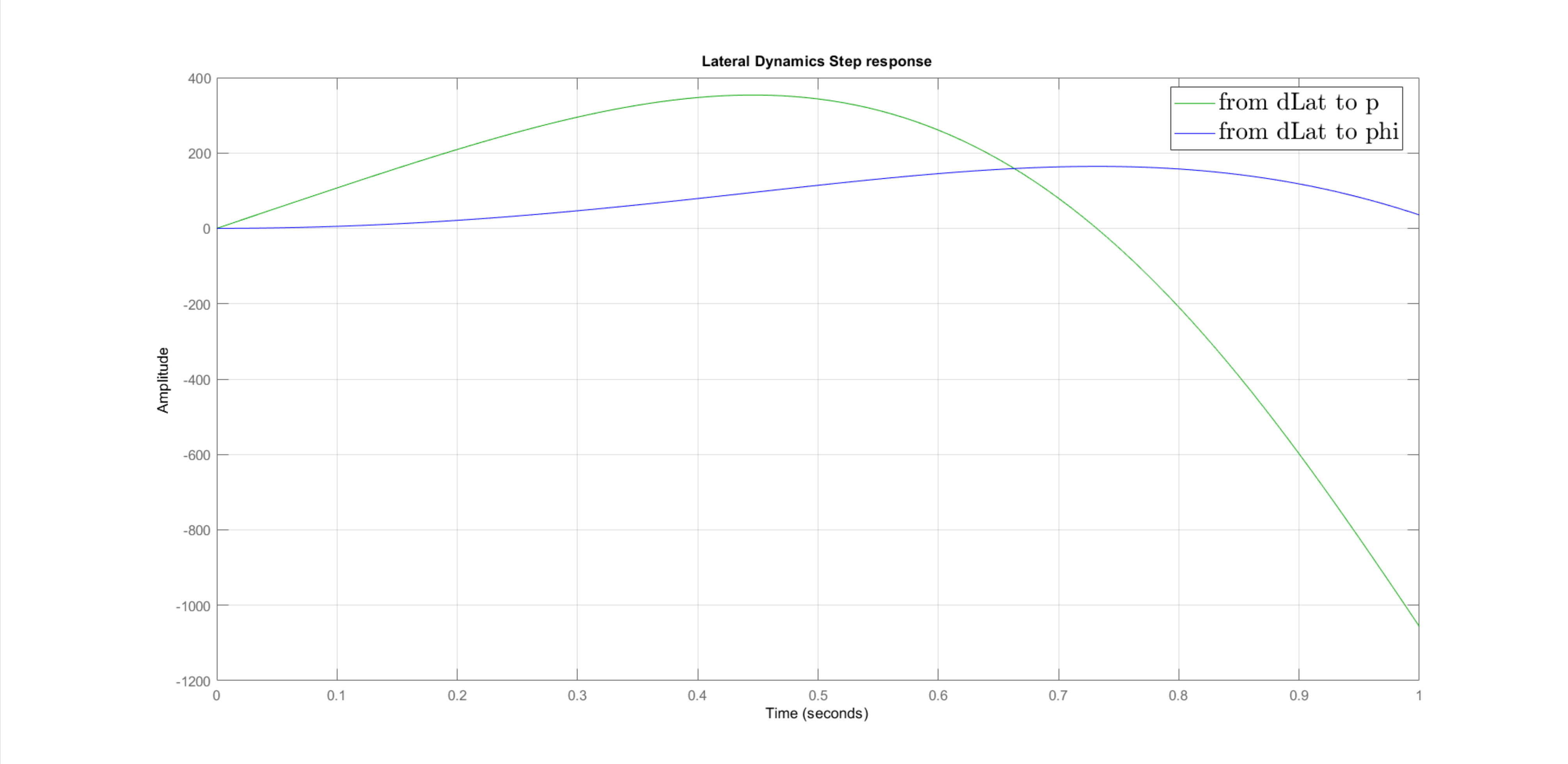
$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

$$u = \delta_{lat}, \qquad y = \begin{bmatrix} p \\ \varphi \end{bmatrix}$$
$$x = \begin{bmatrix} v \\ p \\ \varphi \end{bmatrix}$$

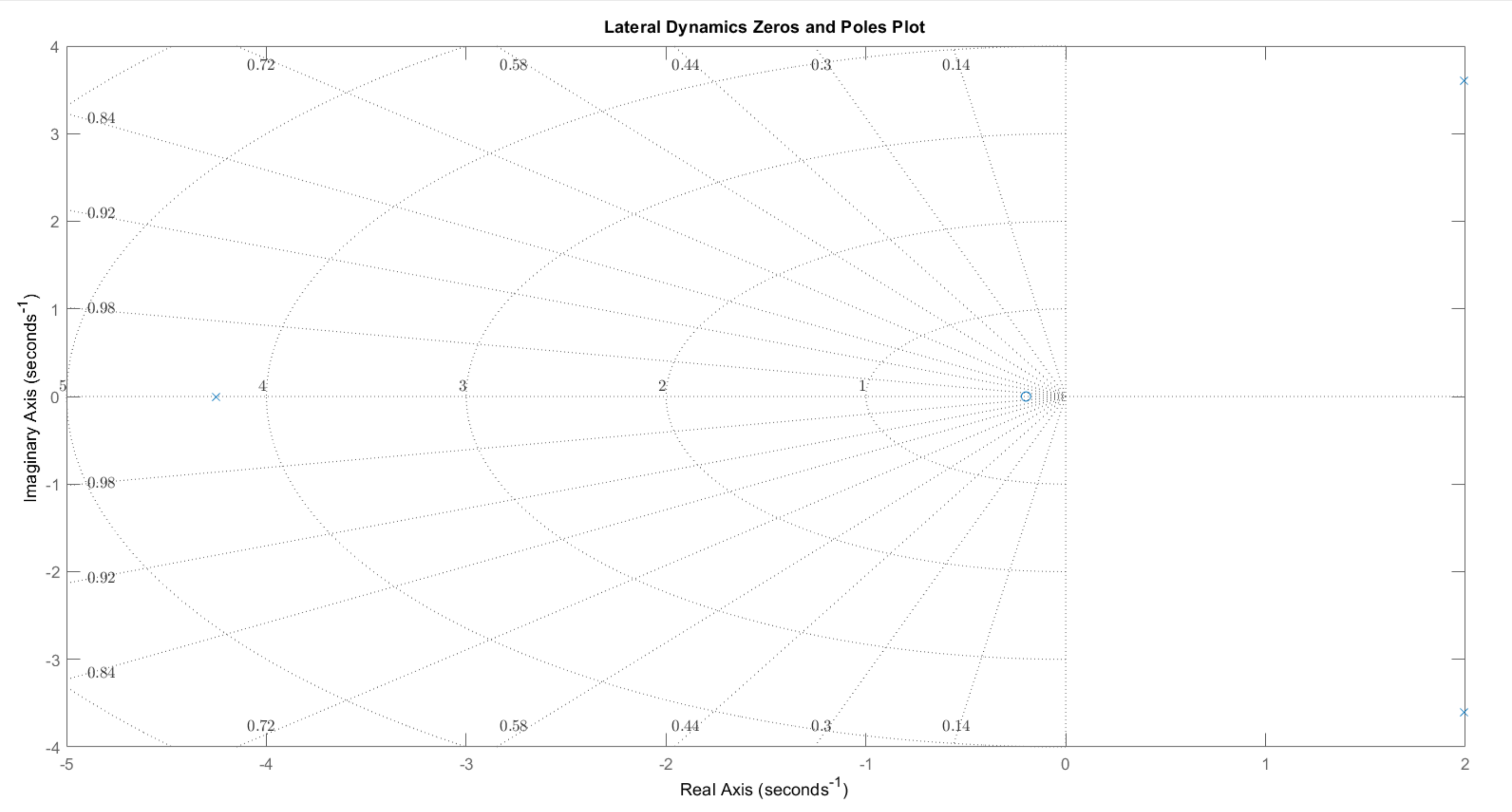
$$A = \begin{bmatrix} Y_v & Y_p & g \\ L_v & L_p & 0 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} Y_\delta \\ L_\delta \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$




The bode diagram shows the system in continuous and discrete time representation (with a sampling time Ts = 0.004s). It is possible to notice that there are computation errors in the discretization process. It occurs only at high frequencies tough, which in our case is acceptable.




This is the step response of the lateral dynamics system. It is clearly notable the bond between the outputs: $\dot{\varphi} = p$




$$P1 = -4.2514 + 0.0000i$$
$$P2 = 1.9937 + 3.6024i$$
$$P3 = 1.9937 - 3.6024i$$

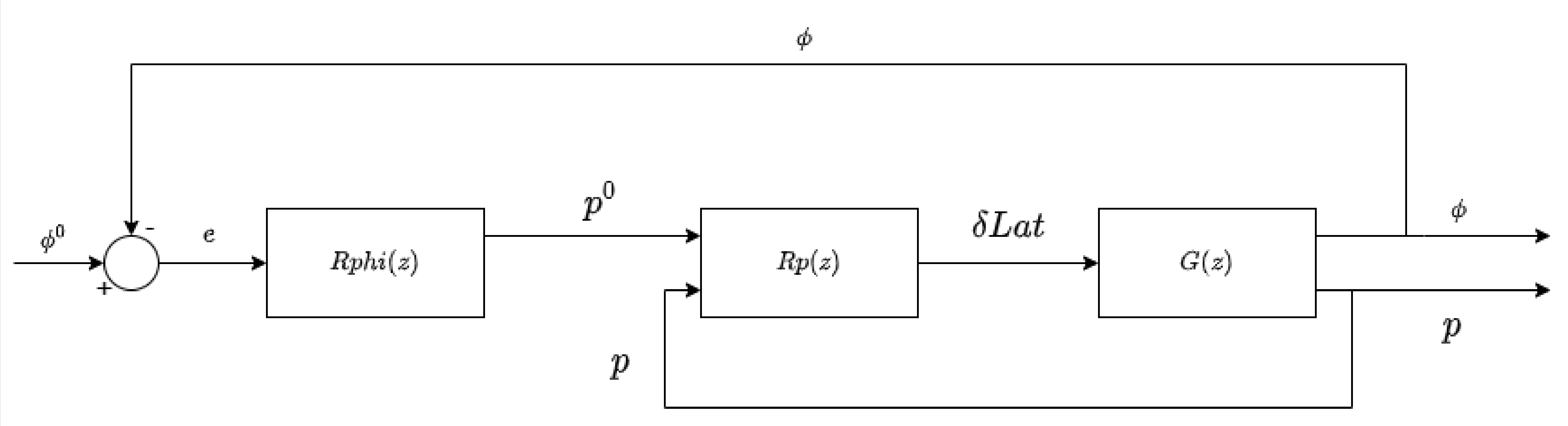
The graphs plots poles and zeros of the lateral dynamics. It is possible to see two complex conjugates poles with positive real part: P2 and P3. Not only do this makes the system unstable, but it is known from literature that this limits the performance of the system. Indeed, the $\omega_c$ pulsation must satisfy the equation:

$\omega_c > p\left(\frac{M}{M-1}\right)$, where: $p = Re(P2, P3)$, $M$ is a performance parameter shown later.

In other words, the system must be faster proportionally to p.

# 1.2 Control System Structure

The structure of the system is the one shown in figure.




It is composed of two feedback loops, with the respective controllers $R\varphi$ and $Rp$.

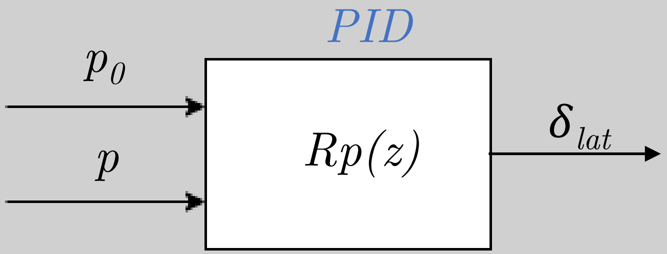
A roll angle $\varphi_0$ is imposed as input, then the system would ideally track it to reach the condition $\varphi_0 = \varphi$. Performance requirements, such as transition shape, are given. Roll rate $p$ is an output of the system too.

The double ringed-structure allows the control system to work in either with the roll angle or, opening the loop, with the roll rate.

The $\boldsymbol{Rp}$ controller acts on the roll rate $p$.

It is basically a **PID** controller and the structure is fixed as follows:

Parameters $b, c1, c2, d1, d2$ are tunable. (Ts = 0.004s).




$$x_p(k+1) = A_p x_p(k) + B_p u_p(k)$$
$$\delta_{lat}(k) = C_p x_p(k) + D_p u_p(k)$$

$$u_p = \begin{bmatrix} p^0 \\ p \end{bmatrix}$$

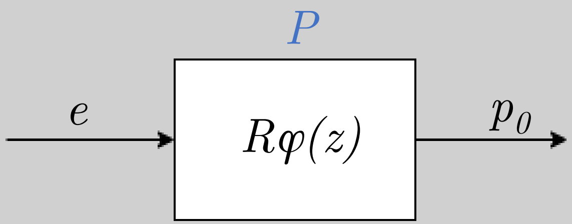
$$A_p = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, B_p = \begin{bmatrix} b & -b \\ 0 & 0.5 \end{bmatrix}, C_p = \begin{bmatrix} c_1 & c_2 \end{bmatrix}, D_p = \begin{bmatrix} d_1 & d_2 \end{bmatrix}$$

The inputs $p_0$ and $p$ have been separated to allow, in the implementation phase, to have less sensitive into the output $\delta_{lat}$ (the control variable), which is a non-dimensional rolling moment.

The $\boldsymbol{R\varphi}$ controller acts on the roll angle $\varphi$.
It has a fixed structure too as follows. (Ts = 0.004s).
It is a simple **P** controller, with a tunable gain $k\varphi$.

$$p^0(k) = D_\varphi e_\varphi(k)$$

$$e_\varphi = \varphi^0 - \varphi$$



- Tunable parameters initialized as struct
  - Full parameters initialization control
- Plant State Space created with following parameters:
  - Nominal modeling
  - Deterministic uncertainty modeling
  - Gaussian distribution modeling

```
%Uncertain Stability Derivatives

%Yv
params.Yv.nominal = -0.264; %1/s
params.Yv.Gaussian = (-0.264/100)*4.837*randn(1);
params.Yv.Name = 'Yv';
params.Yv.Type = 'Percentage';
params.Yv.Range = 3*4.837;

%Yp
params.Yp.nominal = 0; %m/(s*rad)
params.Yp.Gaussian = 0;
params.Yp.Name = 'Yp';
params.Yp.Type = 'Percentage';
params.Yp.Range = 0;

%Lv
params.Lv.nominal = -7.349; %rad*s/m
params.Lv.Gaussian = (-7.349/100)*4.927*randn(1);
params.Lv.Name = 'Lv';
params.Lv.Type = 'Percentage';
params.Lv.Range = 3*4.927;

%Lp
params.Lp.nominal = 0; %1/s
params.Lp.Gaussian = 0;
params.Lp.Name = 'Lp';
params.Lp.Type = 'Percentage';
params.Lp.Range = 0;

%Uncertain Control Derivatives

%Yd
params.Yd.nominal = 9.568; %m/s^2
params.Yd.Gaussian = (9.568/100)*4.647*randn(1);
params.Yd.Name = 'Yd';
params.Yd.Type = 'Percentage';
params.Yd.Range = 3*4.647;

%Ld
params.Ld.nominal = 1079.339; %rad/s^2
params.Ld.Gaussian = (1079.339/100)*2.762*randn(1);
params.Ld.Name = 'Ld';
params.Ld.Type = 'Percentage';
params.Ld.Range = 3*2.762;
```

# • Untuned plant build and discretization

```
%Plant build

A = plant.A;
B = plant.B;
C = plant.C;
D = plant.D;

% Build Lateral dynamics

G = ss(A,B,C,D);
G = c2d(G,Ts,'tustin');
G.InputName = 'dLat';
G.Outputname{1} = 'p';
G.Outputname{2} = 'phi';

% Build 2dof PID controller Rp(s)

Ap = parameters.A;
Bp = parameters.B;
Cp = parameters.C;
Dp = parameters.D;

Rp = ss(Ap,Bp,Cp,Dp,Ts);

Rp.InputName{1} = 'p0';
Rp.InputName{2} = 'p';
Rp.OutputName = 'dLat';
```

```
% Build 1dof P controller (gain) Rphi

Rphi = tf(parameters.Kphi,1,Ts);
Rphi.InputName = 'e';
Rphi.OutputName = 'p0';

Sum = sumblk('e = phi0 - phi');

% System assembly

system = (connect(G,Rp,Rphi,Sum,'phi0',{'p','phi'}));
```

# 1.3 Controller Design Process

The tuning process for the controllers has the following nominal performance requirement:

A. Nominal Performance:

   i. Response of $\varphi$ to variations in $\varphi^0$: equivalent to a second-order response with $\omega_n \geq 10\ rad/s$ and $\xi \geq 0.9$.

Tuning the plant has been performed with Matlab integrated command "*systune*":

```
options = systuneOptions('RandomStart',30,'UseParallel',true,'SoftTol',1e-7);
[tunedSystem,~,~] = systune(system,Req1,Req2,options);
```

Chosen requirements:

```
Req1 = TuningGoal.StepTracking('phi0','phi',tf2);
```
(as a soft req.)

```
Req2 = TuningGoal.Tracking('phi0','phi',responsetime,dcerror,peakerror);
```
(as a hard req.)

Possible to insert "soft goals" and "hard constraints", as requisites.

'RandomStart' is used to avoid biased solution due to local minimas.

Requirement 1 (on input $\varphi_0$ on output $\varphi$):

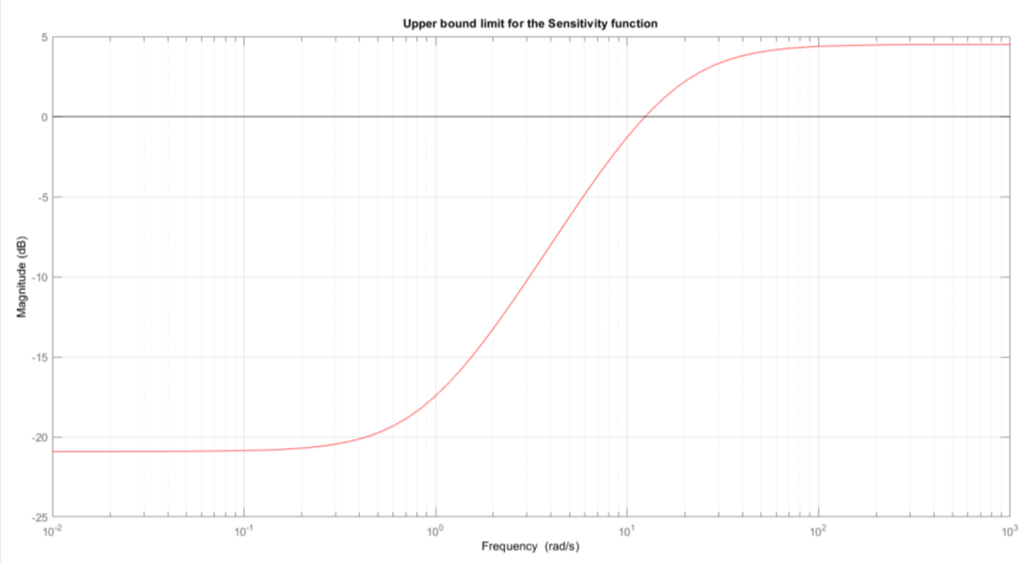
Impose a step tracking goal of a second order transfer function, with $\omega_n = 10 rad/s$ $\xi = 0.9$.


Requirement 2 (on input $\varphi_0$ on output $\varphi$):

The constraint is to impose an upper bound for the sensitivity function.

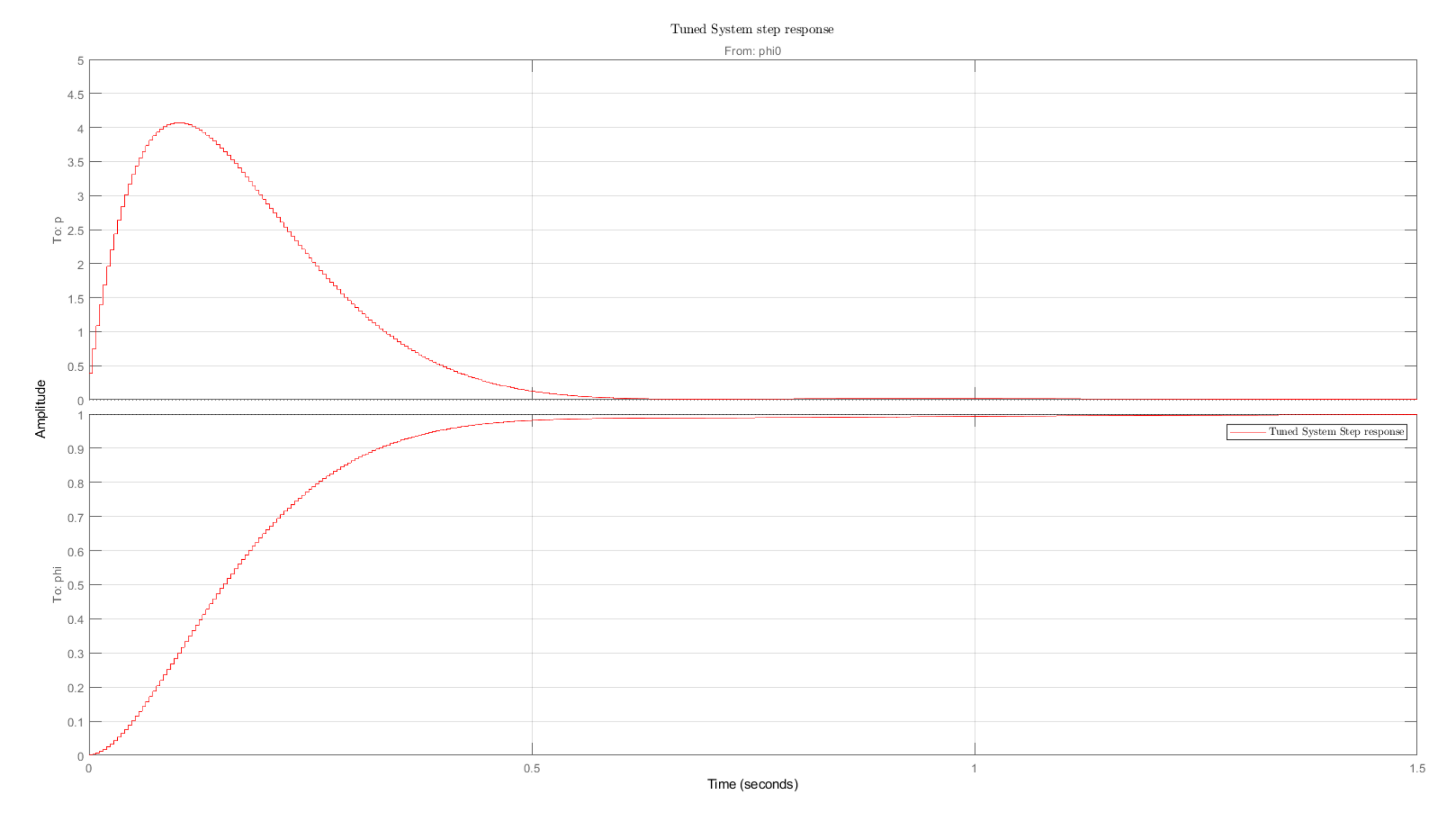
Weighting function $w_p$ is defined as follows:

$$w_p = \frac{\frac{s}{M}+\omega_b}{s+A\omega_b} \qquad A=0.1 \quad \boldsymbol{\omega_b = 10} \qquad M=1.68$$




Note that we do not choose to have a requirement on output $p$, due to bond between it and $\varphi$

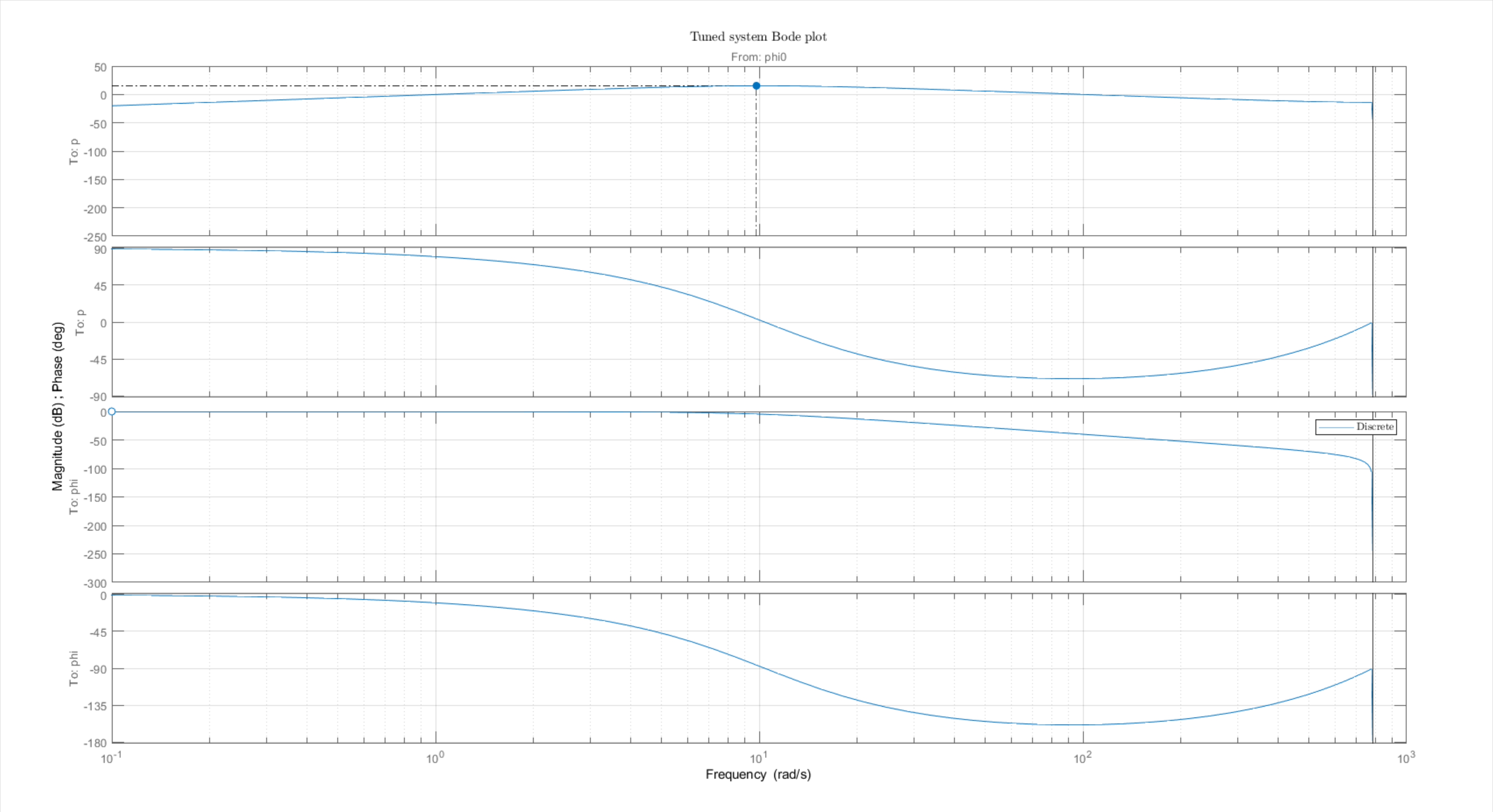
# 1.4 Nominal Tuned System Analysis

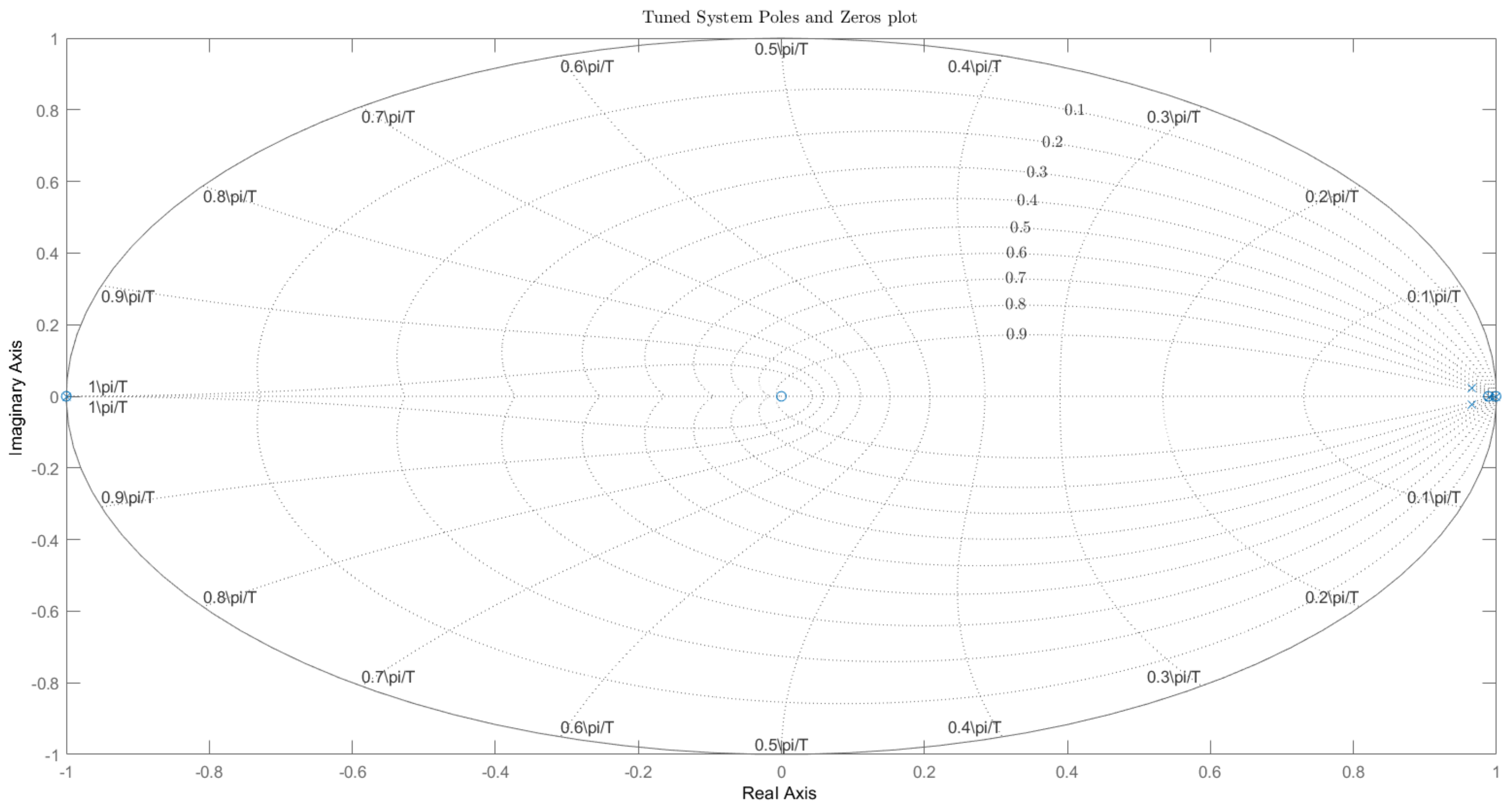


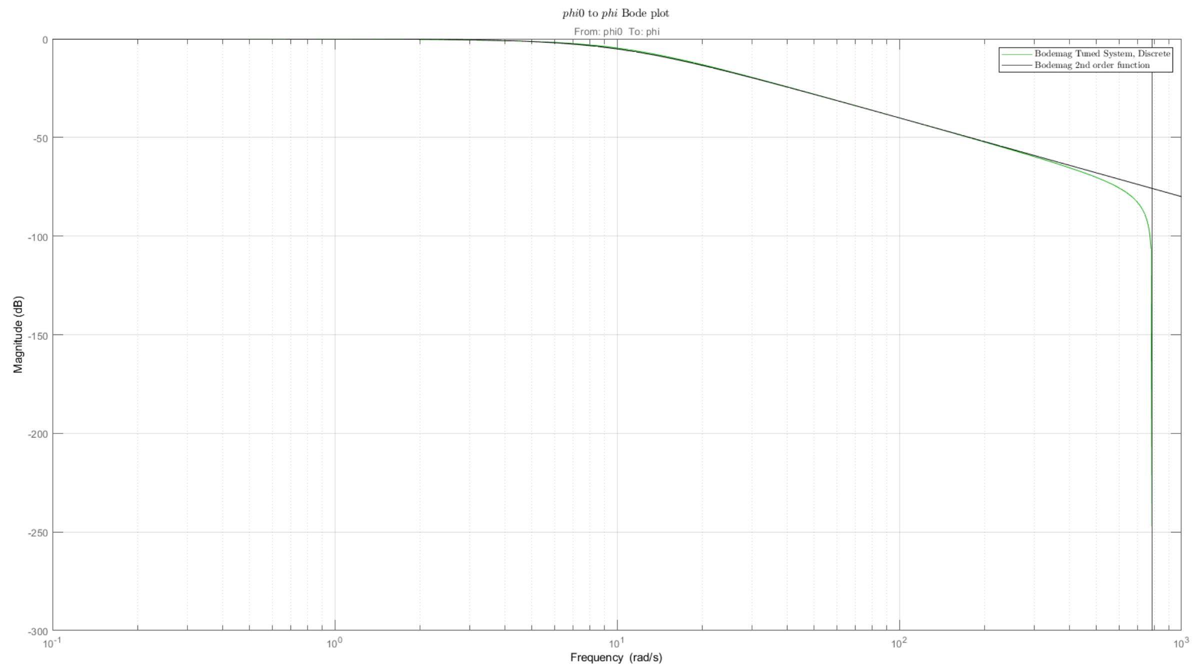

Nominal time domain response exposes the desired behavior, with $\varphi_0$ to $\varphi$ tracking following a 2[nd] order response shape and asymptotic stability.

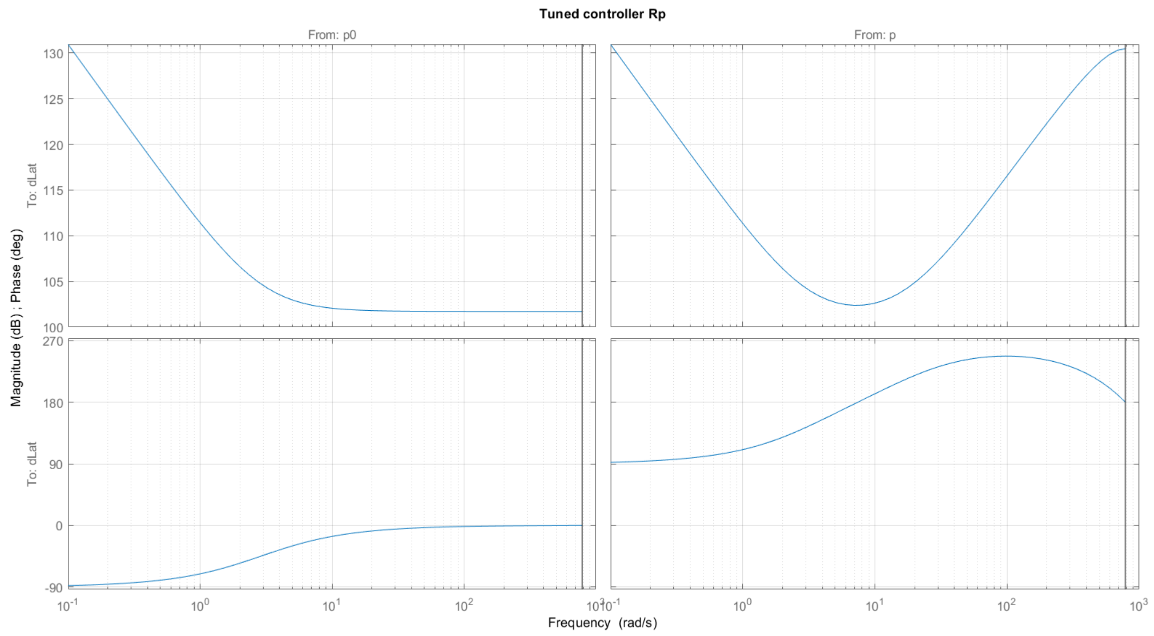
A convergence test of the parameters, over multiple runs, has been done. The results are listed below.

```
%Tuned parameters

SystemTuned.Blocks.b= -5.48
SystemTuned.Blocks.c1 = -5.24
SystemTuned.Blocks.c2 = -0.64
SystemTuned.Blocks.d1 = 0.14
SystemTuned.Blocks.d2 = -1.45
```


Tuned system Bode plot
From: phi0
Magnitude (dB) ; Phase (deg)
To: p
To: p
To: phi
To: phi
Discrete
Frequency (rad/s)
50
0
-50
-100
-150
-200
-250
90
45
0
-45
-90
0
-50
-100
-150
-200
-250
-300
0
-45
-90
-135
-180
$10^{-1}$
$10^{0}$
$10^{1}$
$10^{2}$
$10^{3}$

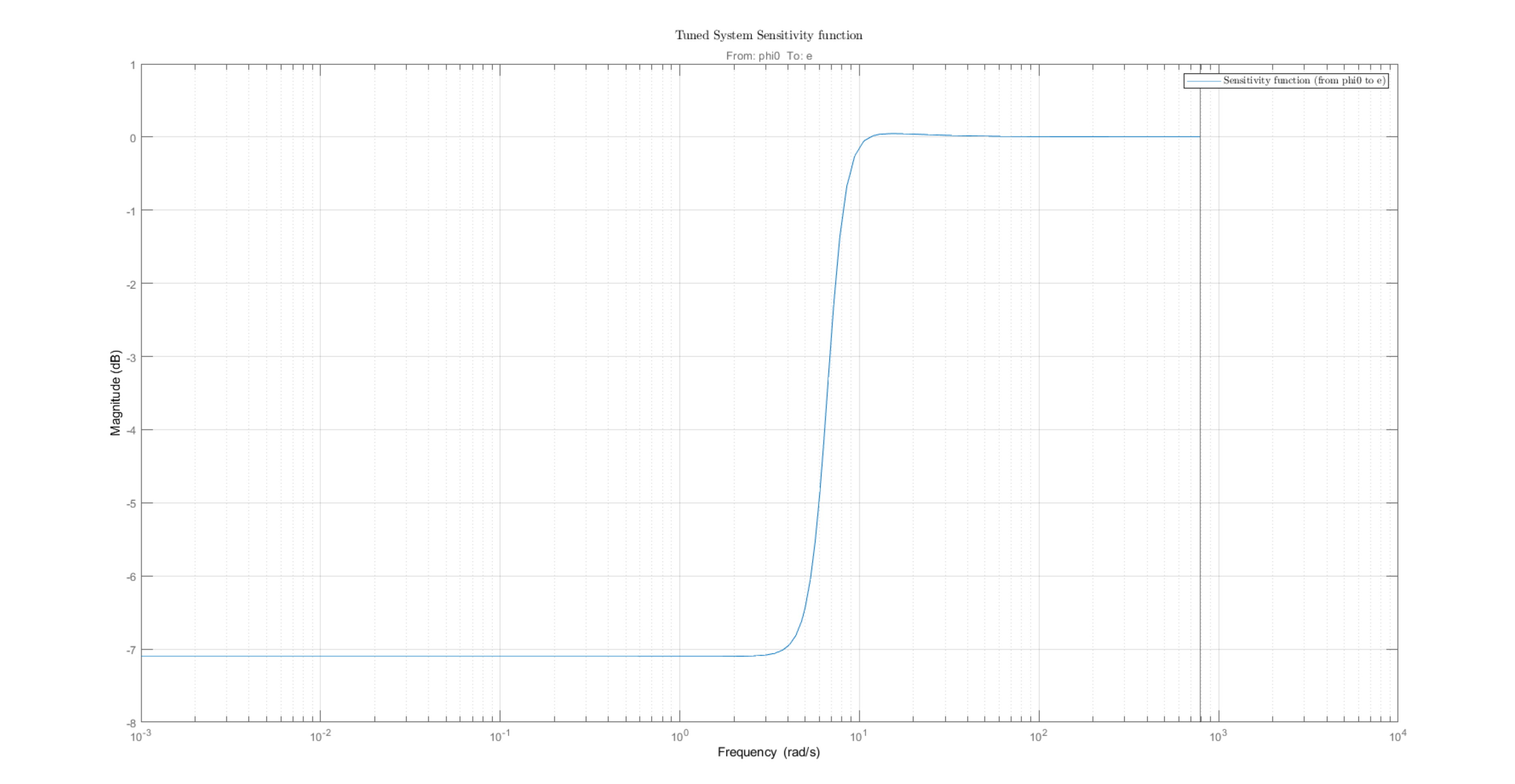

Tuned System Poles and Zeros plot
Imaginary Axis
Real Axis
0.5\pi/T
0.6\pi/T
0.4\pi/T
0.7\pi/T
0.3\pi/T
0.8\pi/T
0.2\pi/T
0.9\pi/T
0.1\pi/T
1\pi/T
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9



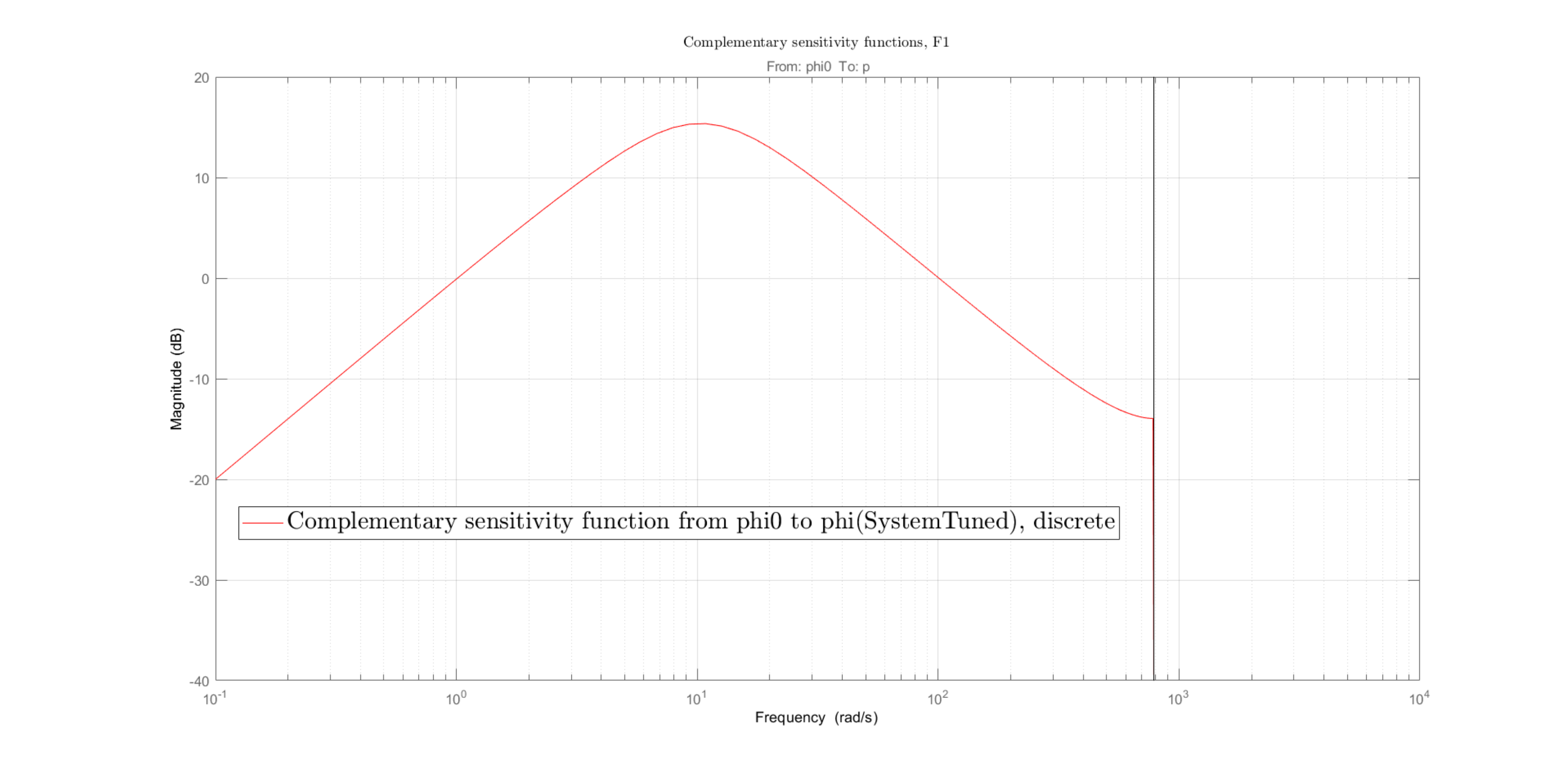




$\varphi_0$ to $\varphi$ transfer function for both discrete plant and reference tf.
On the right-hand side the PID controller tf is highlighted for both inputs.

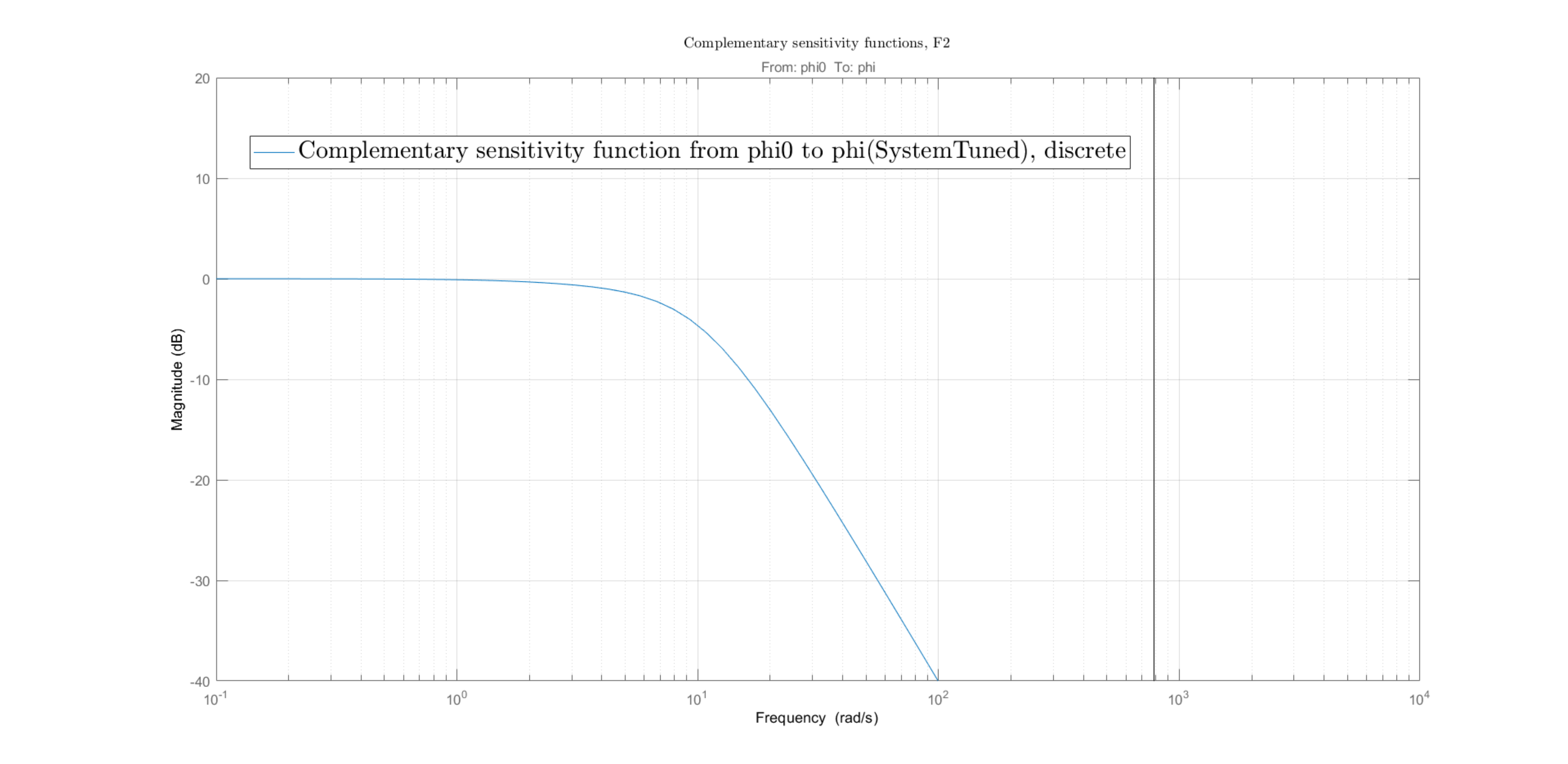
# 1.5 Nominal Tuned System Sensitivities Analysis



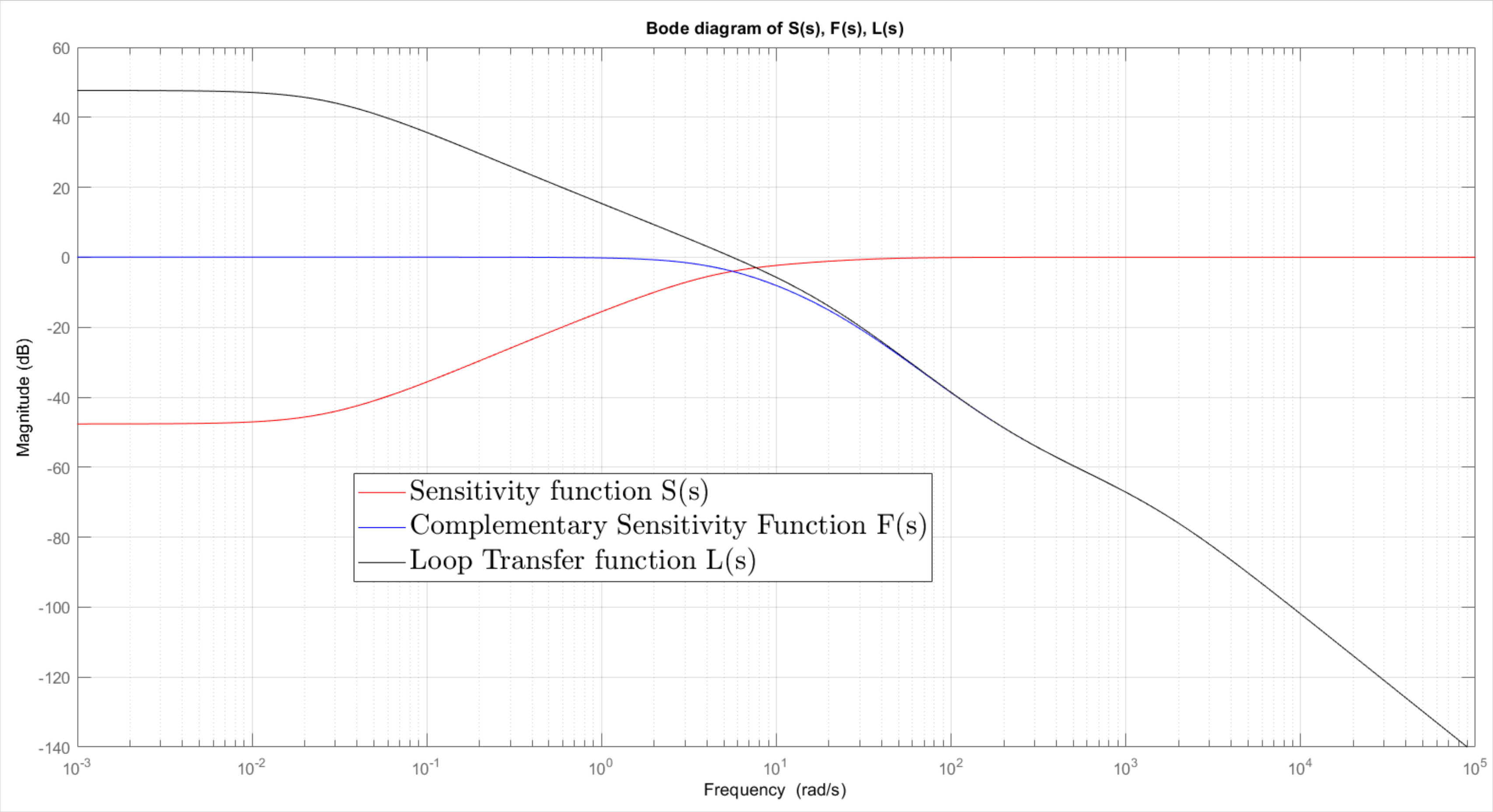

Tuned system sensitivity function exposes a good attenuation of disturbances at lower frequencies

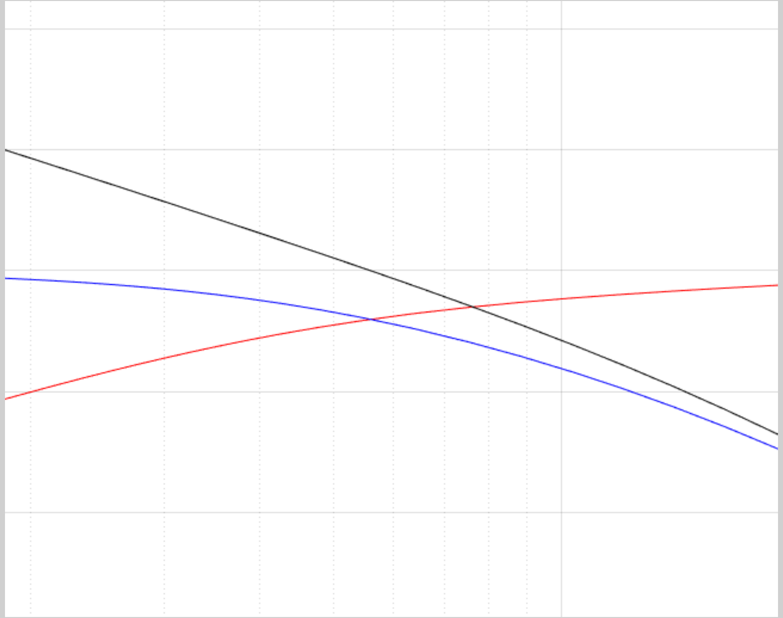



Nominal time domain response exposes the desired behavior, with phi0 to phi tracking following a $2^{nd}$ order response shape and asymptotic stability.




Nominal time domain response exposes the desired behavior, with phi0 to phi tracking following a 2$^{nd}$ order response shape and asymptotic stability.



Note that, as expected, the cross frequency of F(s) is placed slightly before L(s) cross frequency. On the contrary, cross frequency of S(s) come slightly after L(s) cross frequency. However, $\omega_c$ is around 10 rad/s, as design requirement.

# 2. Uncertain System Analysis

## 2.1 Robust Stability

The nominal system design has been done assuming that system model and parameters are perfectly known. However, in real applications, things are not so easy.

The uncertainty can be:

- Parametric, if it affects numerical parameters of the model.
- Nonparametric, if it affects the structure of transfer functions itself.

Guarantee the Robust Stability (RS) means guarantee the stability also if the model is not perfectly known.

In this project, the uncertainty is given on model parameters with a Gaussian distribution.

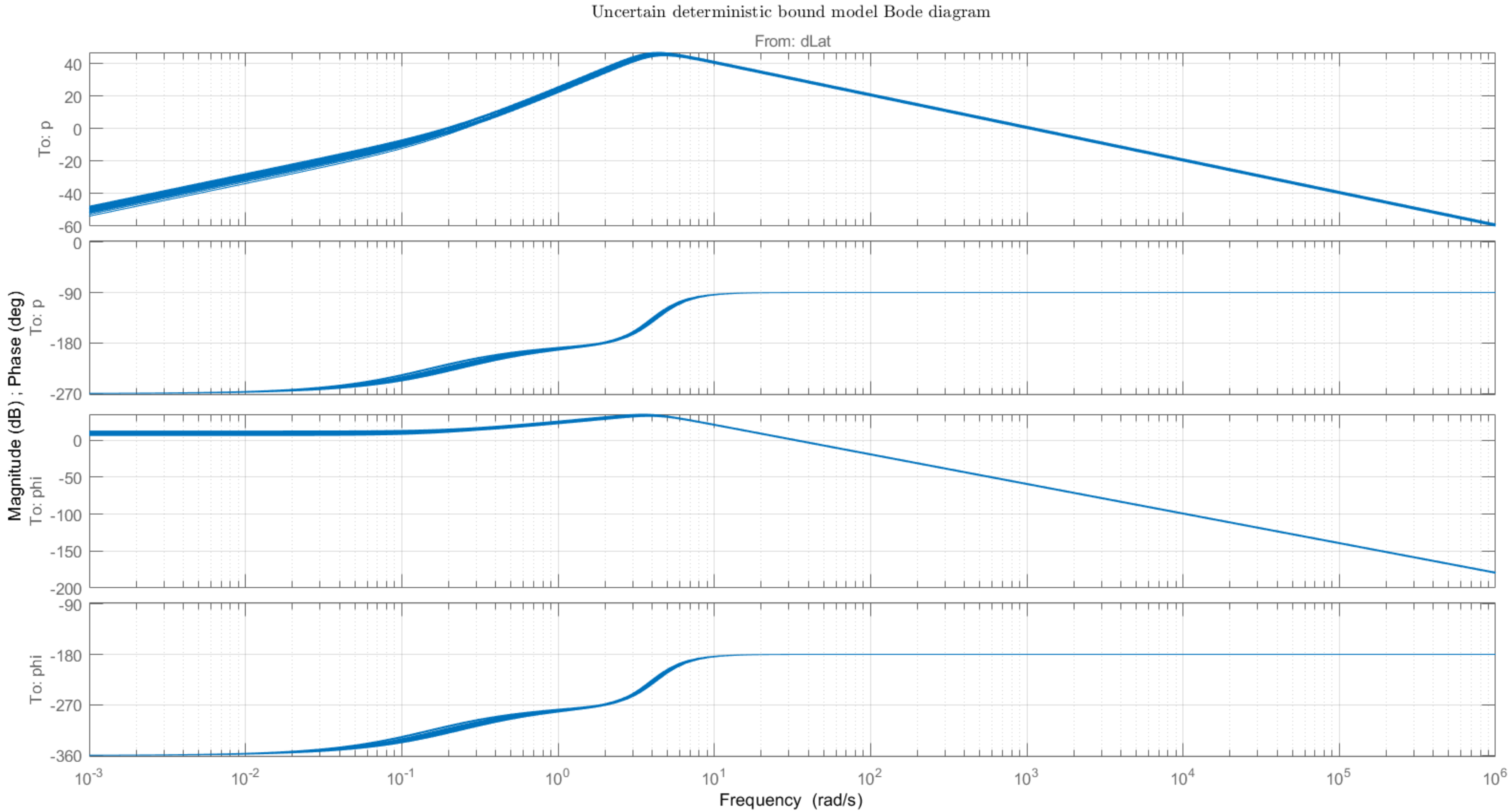
Follows uncertain parameters with associated nominal values and percentage standard deviation:

Stability Derivatives:

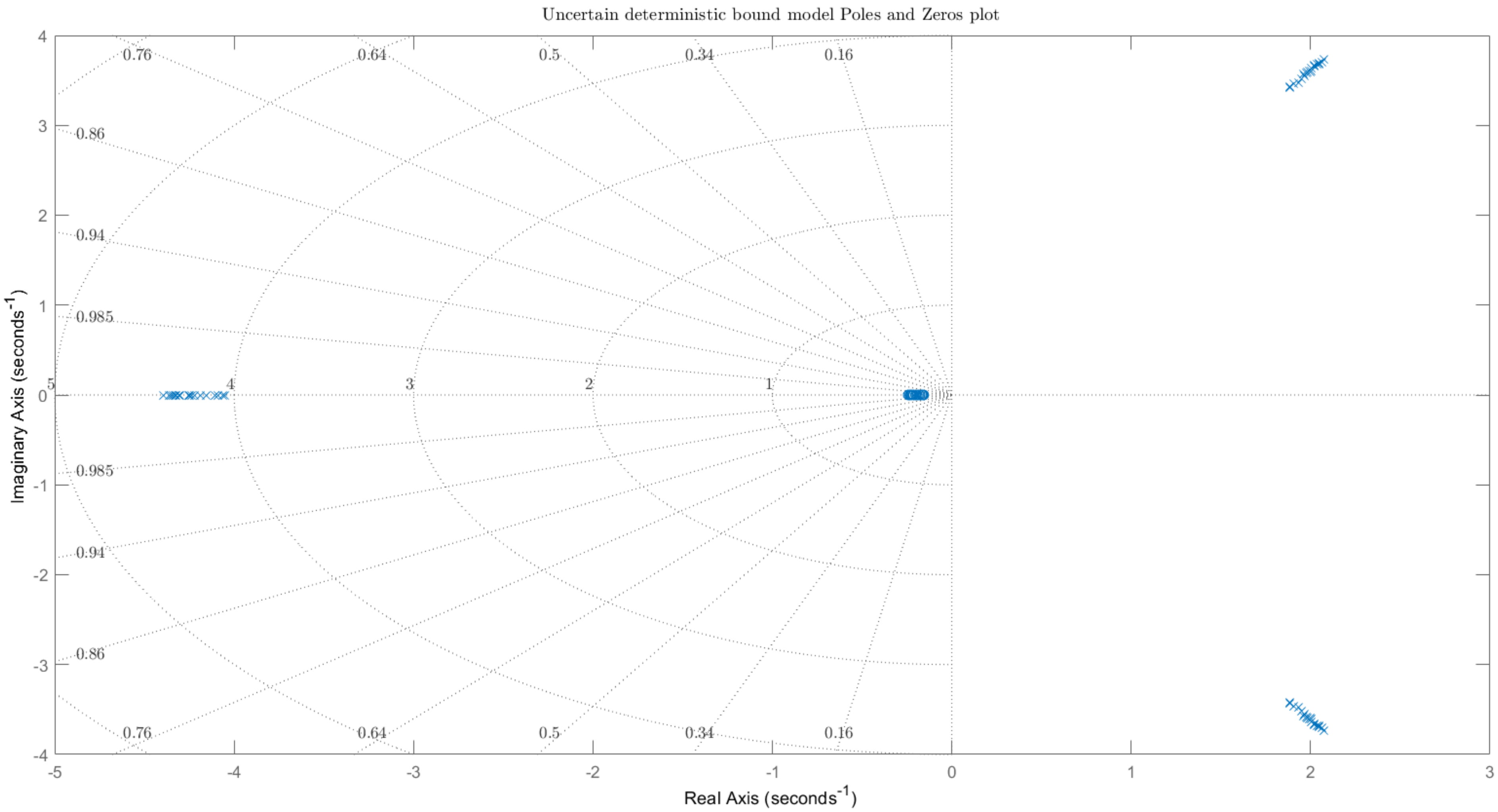
| | Nominal Value | Standard deviation % |
|---|---|---|
| Yv | -0.264 1/s | 4.837% |
| Lv | -7.349 rad s/m | 4.927% |

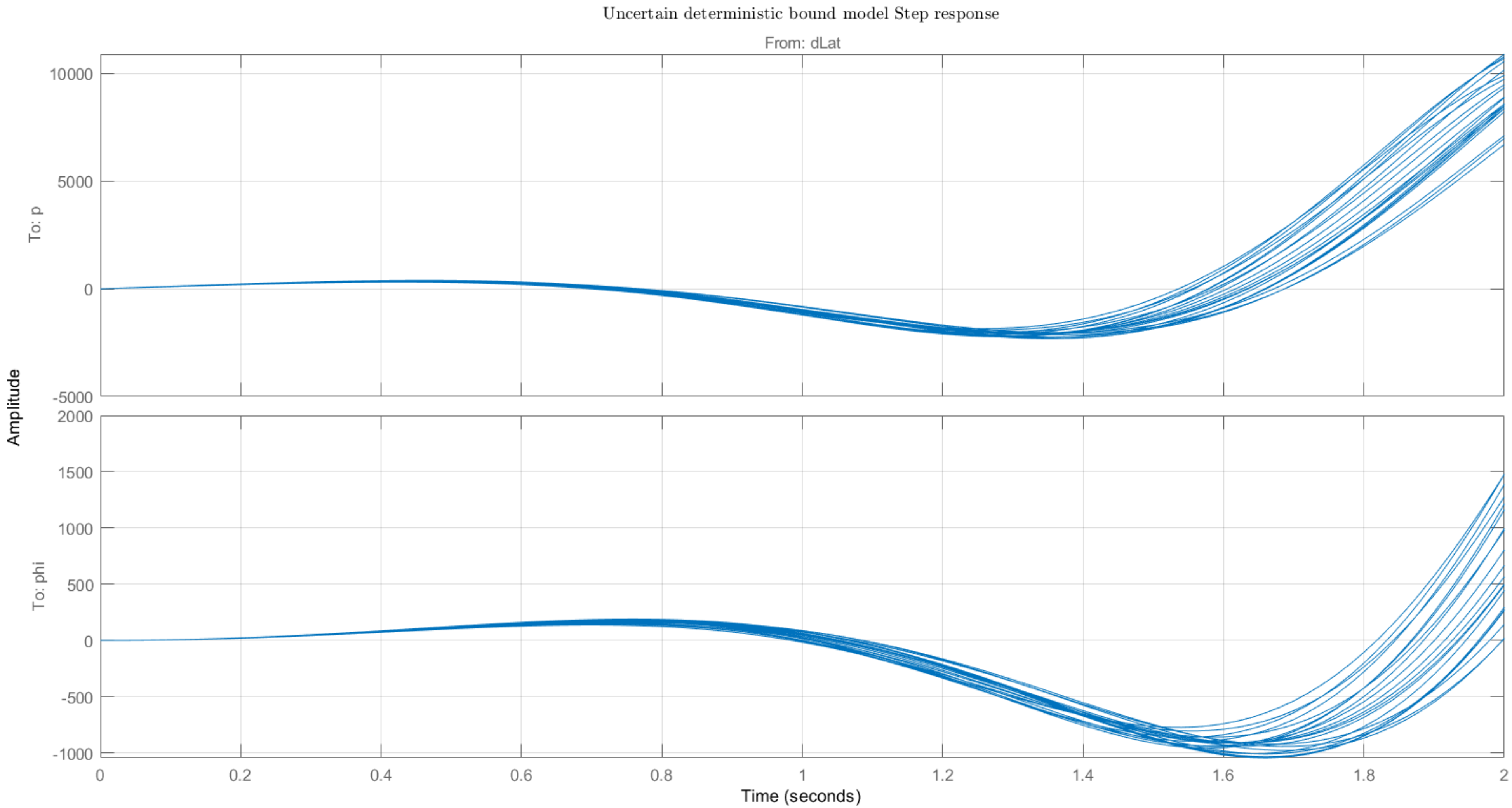
Control Derivatives:

| | Nominal Value | Standard deviation % |
|---|---|---|
| Yd | 9.568 m/s2 | 4.647% |
| Ld | 1079.339 rad/s2 | 2.762% |



Uncertain deterministic bound model Bode diagram
From: dLat
To: p
To: phi
Magnitude (dB) ; Phase (deg)
40
20
0
-20
-40
-60
-90
-180
-270
-50
-100
-150
-200
-360
$10^{-3}$
$10^{-2}$
$10^{-1}$
$10^{0}$
$10^{1}$
$10^{2}$
$10^{3}$
$10^{4}$
$10^{5}$
$10^{6}$
Frequency (rad/s)


Uncertain deterministic bound model Poles and Zeros plot
Imaginary Axis (seconds$^{-1}$)
Real Axis (seconds$^{-1}$)
0.76
0.64
0.5
0.34
0.16
0.86
0.94
0.985
4
3
2
1


Uncertain deterministic bound model Step response
From: dLat
10000
5000
0
-5000
To: p
Amplitude
2000
1500
1000
500
0
-500
-1000
To: phi
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Time (seconds)

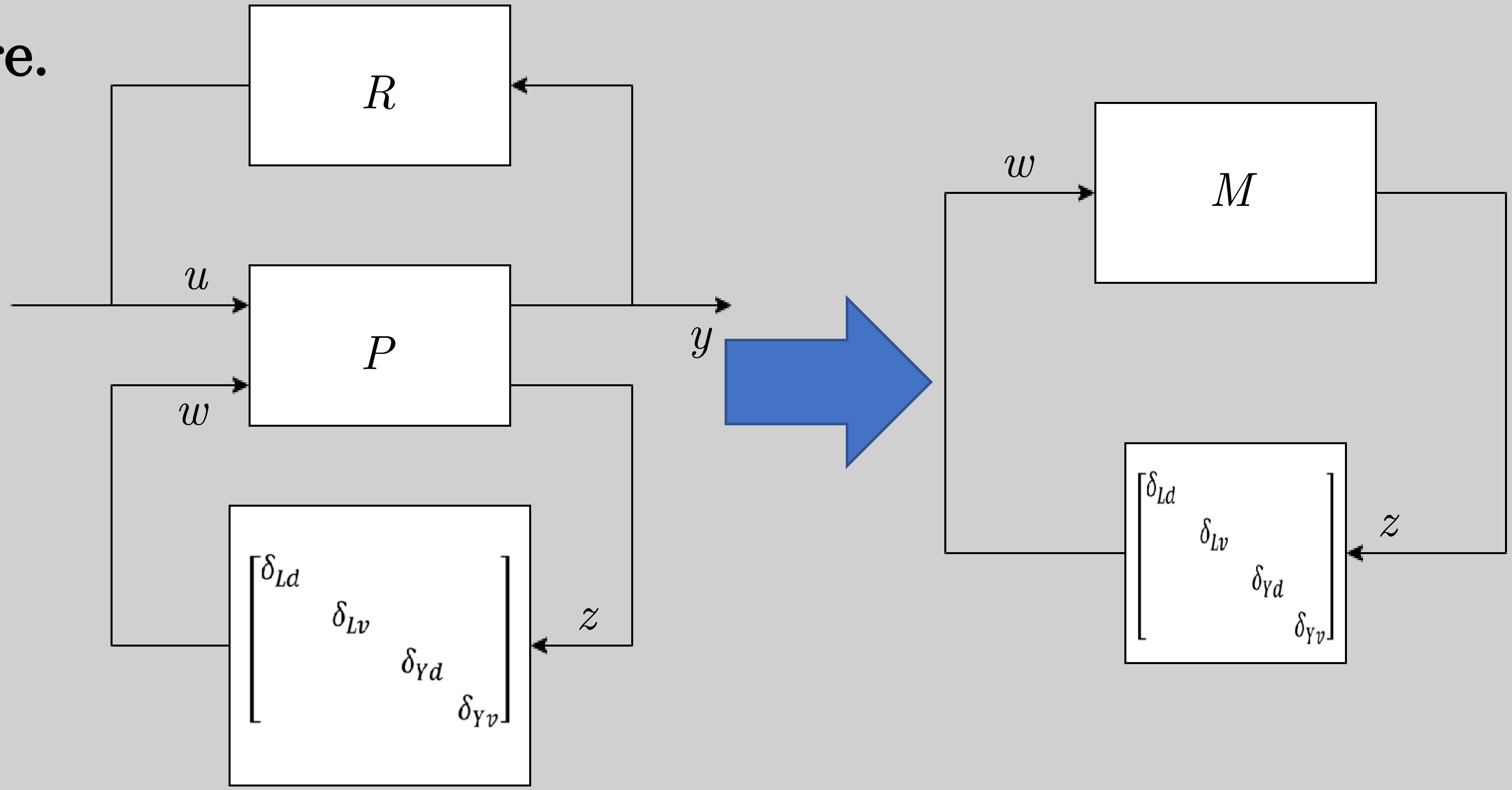
```
%Stability Derivatives
Yv = ureal('Yv',-0.264,'Percentage',3*4.837); %1/s
Yp = 0; %m/(s*rad)
Lv = ureal('Lv',-7.349,'Percentage',3*4.927); %rad*s/m
Lp = 0; %1/s
%Control Derivatives
Yd = ureal('Yd',9.568,'Percentage',3*4.647); %m/s^2
Ld = ureal('Ld',1079.339,'Percentage',3*2.762); %rad/s^2
```

# 2.1.1 Robust Stability, summary of Different Approaches.

To deal with uncertainty, the idea is to separate uncertainties from the model. This is possible with two different approaches:

## 1)"Pulling out the Deltas" procedure.

The system is divided in two parts: M, which contains the nominal part and the square matrix Delta. The dimension of Delta is equal to the number of uncertainties.




*This Model:*

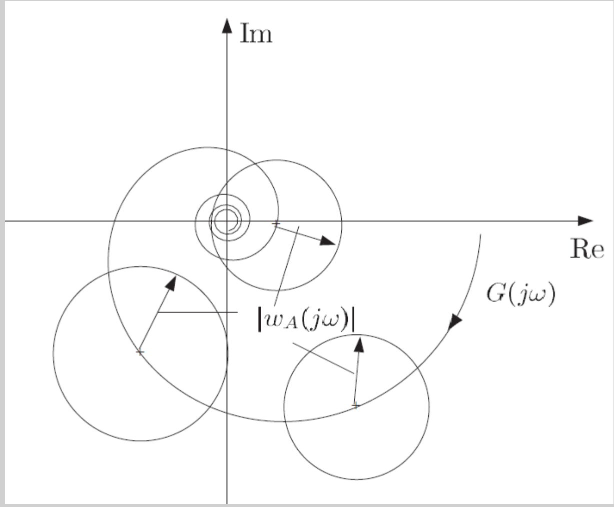
- *It's an exact representation, because considers uncertainties one by one.*
- *It's quite difficult to deal with because it has a multi-channel feedback.*

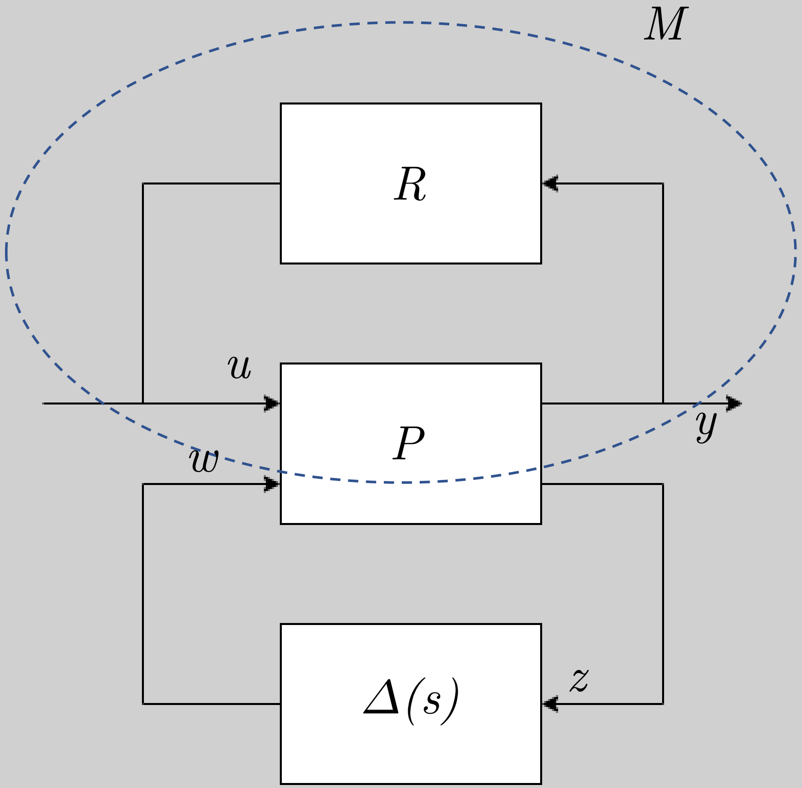
# 2)"Try every combination and Overbound", the relative error covering procedure.

This procedure divides always the system in a pure nominal part and an uncertain part, but now uncertainties are "put together" in a single W(jω), Δ(jω), by the error covering procedure.

$$Gp(j\omega) = \overline{G(j\omega)} \quad + \quad G(j\omega)W(j\omega)\Delta(j\omega)$$

Defined the radius: $w_A = G(j\omega)W(j\omega)$



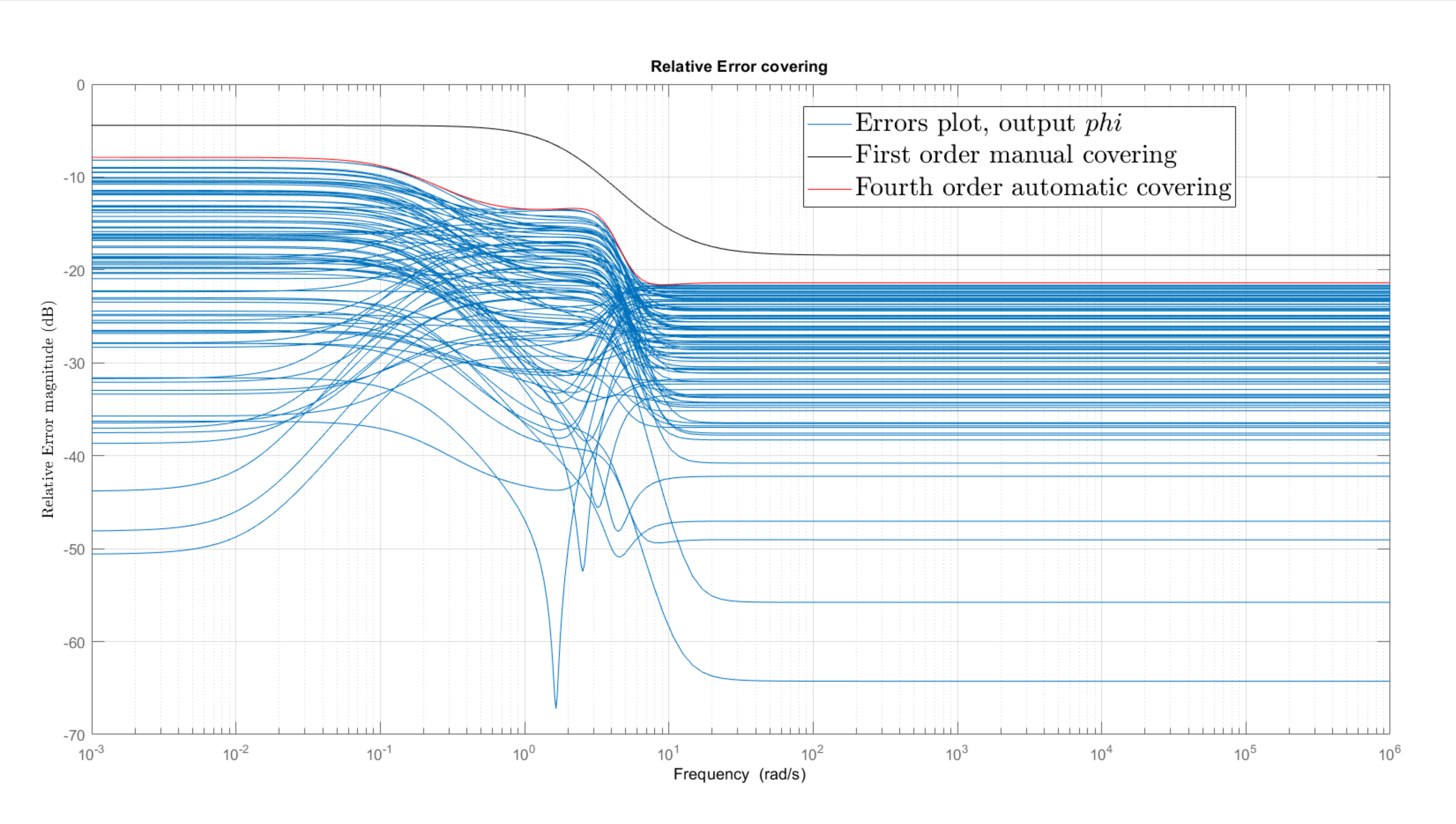



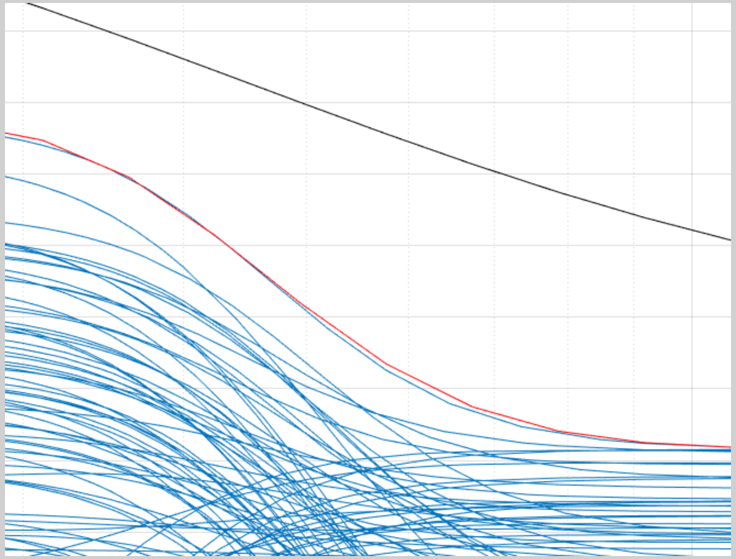

$\Delta(j\omega)$ becomes a scaling factor, with:
$0° < \varphi(\Delta(j\omega)) < 360°$ *and* $|\Delta(j\omega)| < 1$

*This Model:*

- *It's conservative if Delta is not complex.*
- *It's easier to deal with because it has always a singular channel feedback.*

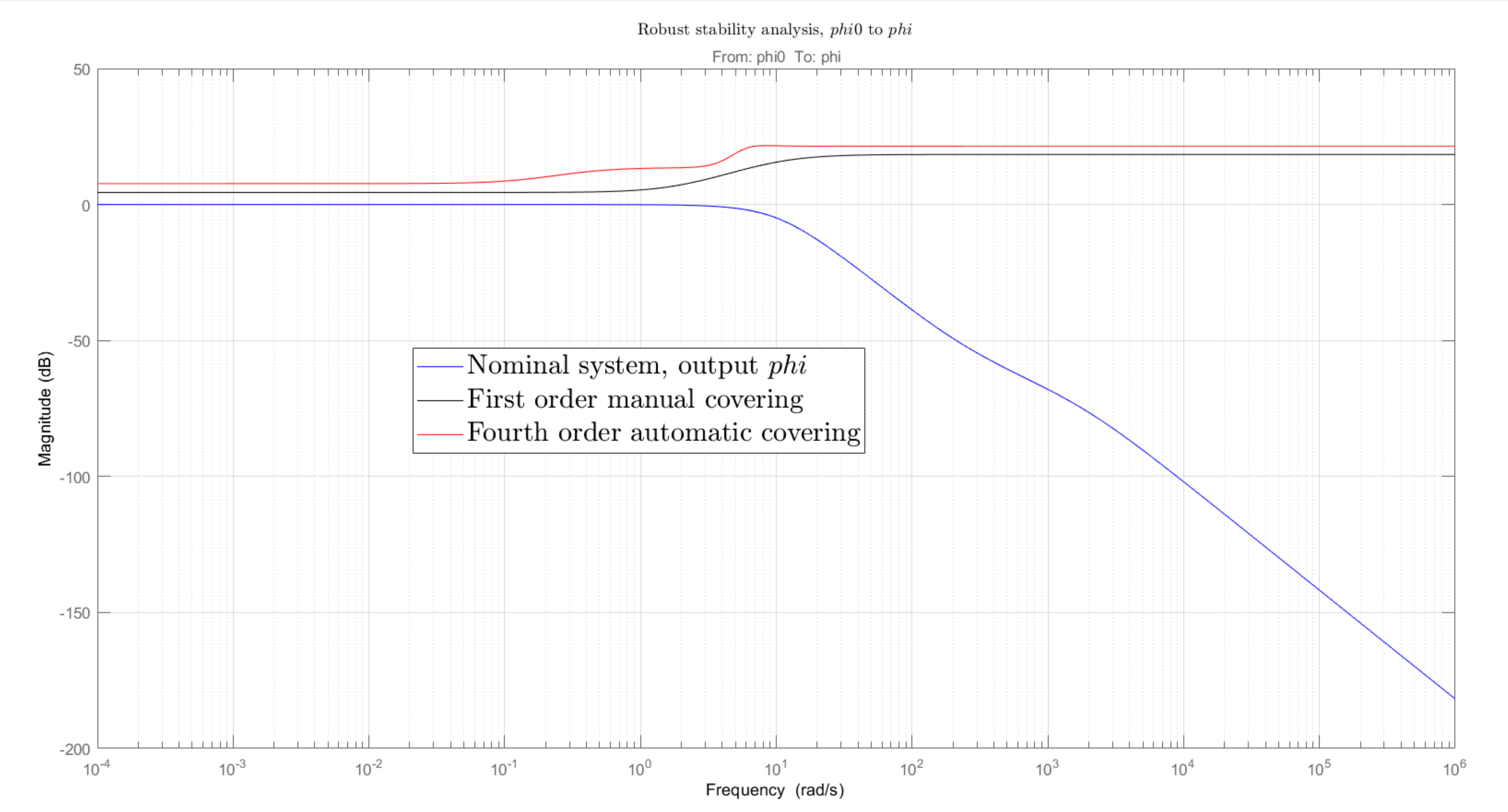
# 2.1.2 Robust Stability, Relative Error Covering Procedure



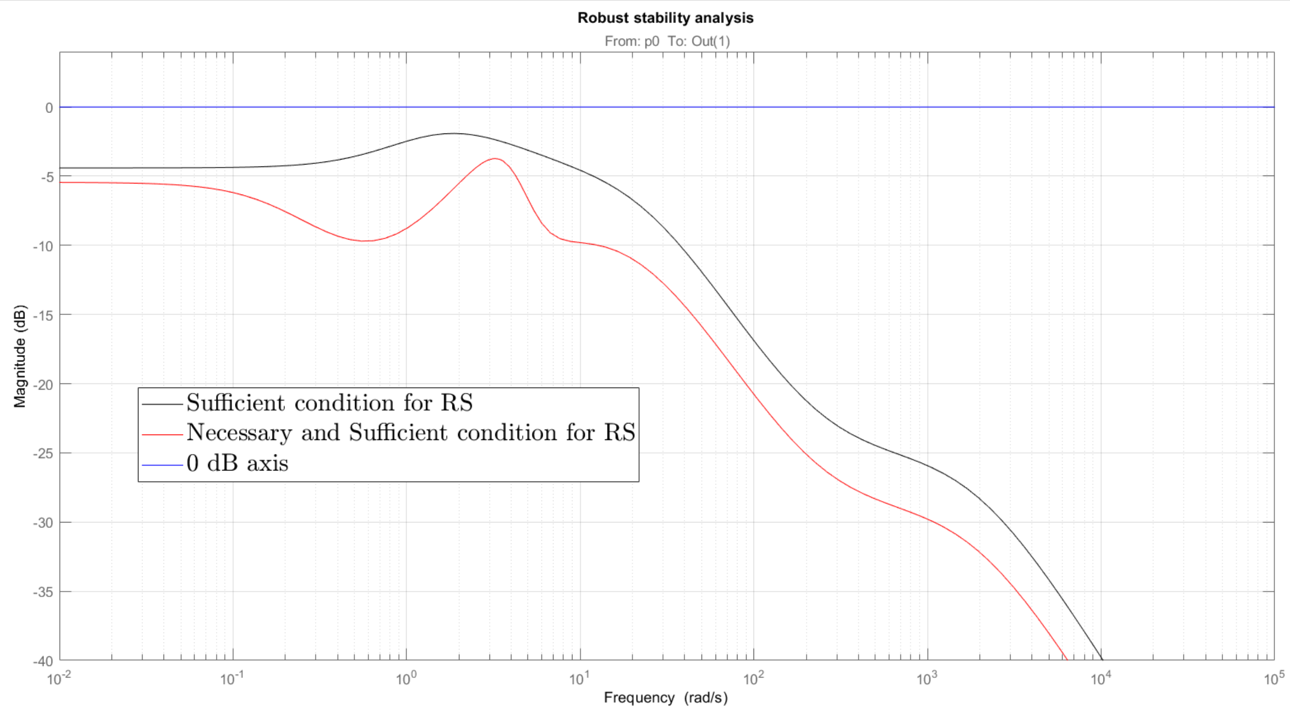



Relative errors magnitude has been estimated by 150 random initialization.

$$\frac{|Gp(j\omega) - \overline{G(j\omega)}|}{|\overline{G(j\omega)}|} = |W(j\omega||\Delta(j\omega| \leq |W(j\omega)|$$




R.S. $\leftrightarrow$ $\left|\overline{F(j\omega)}\right| < \dfrac{1}{|W(j\omega)|}\,, \quad \forall\omega$




$$M(s) = -W(s)\bar{F}(s)$$

$$M(s) = -\,W2(s)\overline{F2}(s)$$

$$\mathrm{R.S} \quad \leftrightarrow \quad |M(j\omega)| < 1\,, \qquad \forall\omega$$

## And what about the output $p$?

The third row of the lateral dynamic space state system shows the relation between $\varphi$ *and* $p$*. Precisely:*

$$\dot{\varphi} = p$$

If $\varphi$ *is constant,* $p = 0$

For this reason, the RS analysis has been executed only on output $\varphi$.

```
G1array = usample(G1db,100); %numero di volte che prova
Gn1 = G1db.NominalValue;



NN1 = 0.6*[0.15 1];
DD1 = [0.7 1];
Wt1 = tf(NN1,DD1,Ts);
figure, bodemag((Gn1-G1array)/Gn1,{0.001,1000000},Wt1,'r'),grid,ylabel('Error magnitude'),title('Manual covering at first order’);


[G1db,Info1] = ucover(G1array,Gn1,4); %copertura al 4 ordine
figure,bodemag((Gn1-G1array)/Gn1,Info1.W1,{0.001,1000000},'k'),grid, ylabel('Error magnitude'),title('Automatic covering at fourth order');
```

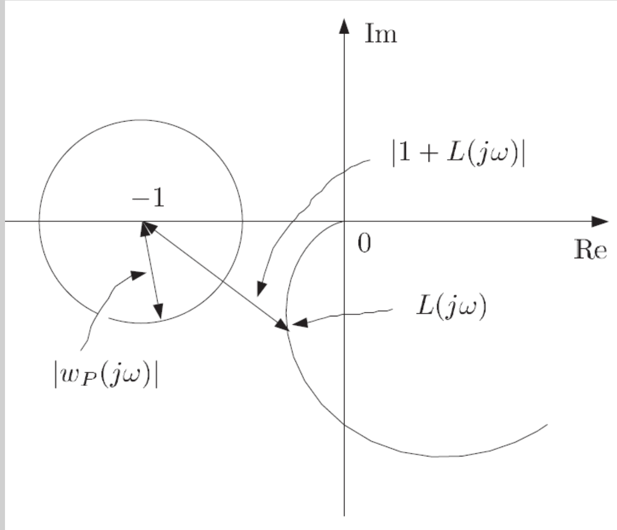
# 2.2 Robust Performance analysis

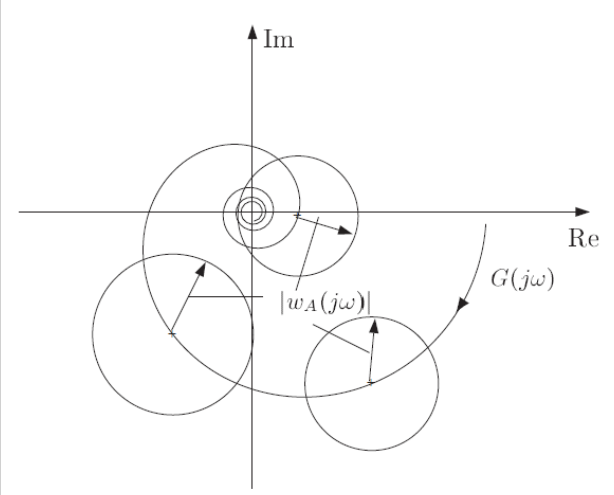
- Nominal performance equation

$$|\overline{S(j\omega)}| < \frac{1}{|W_p(j\omega)|}, \forall\omega$$

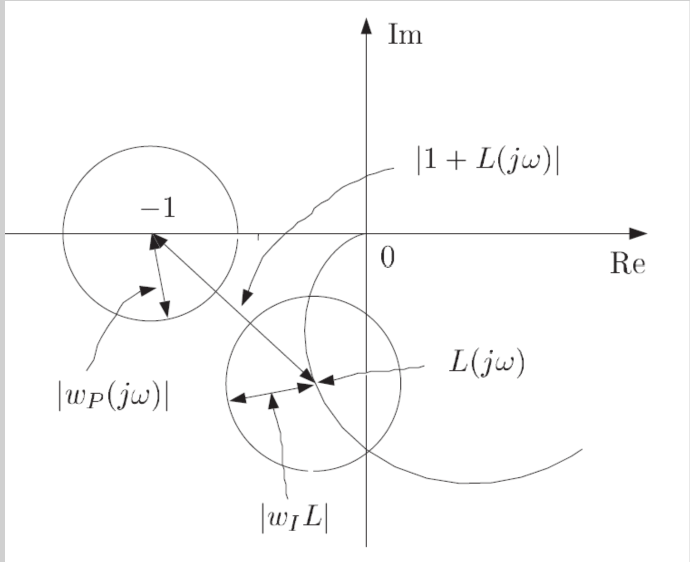
- Robust performance equation

$$|\overline{F(j\omega)}| < \frac{1}{|W\ (j\omega)|}, \forall\omega$$



- The Robust performance equation come from the imposition that the two circles does not intersect

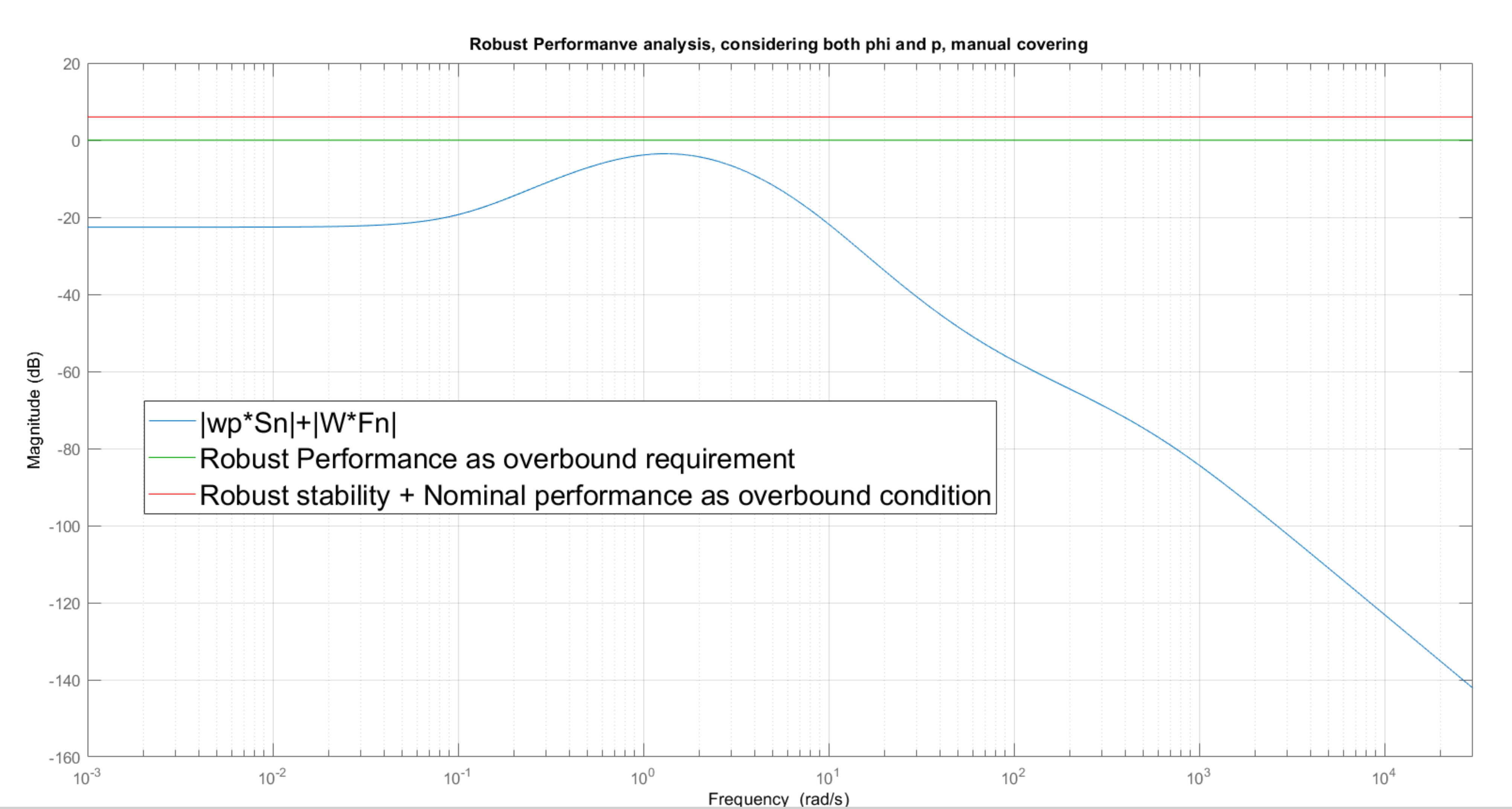


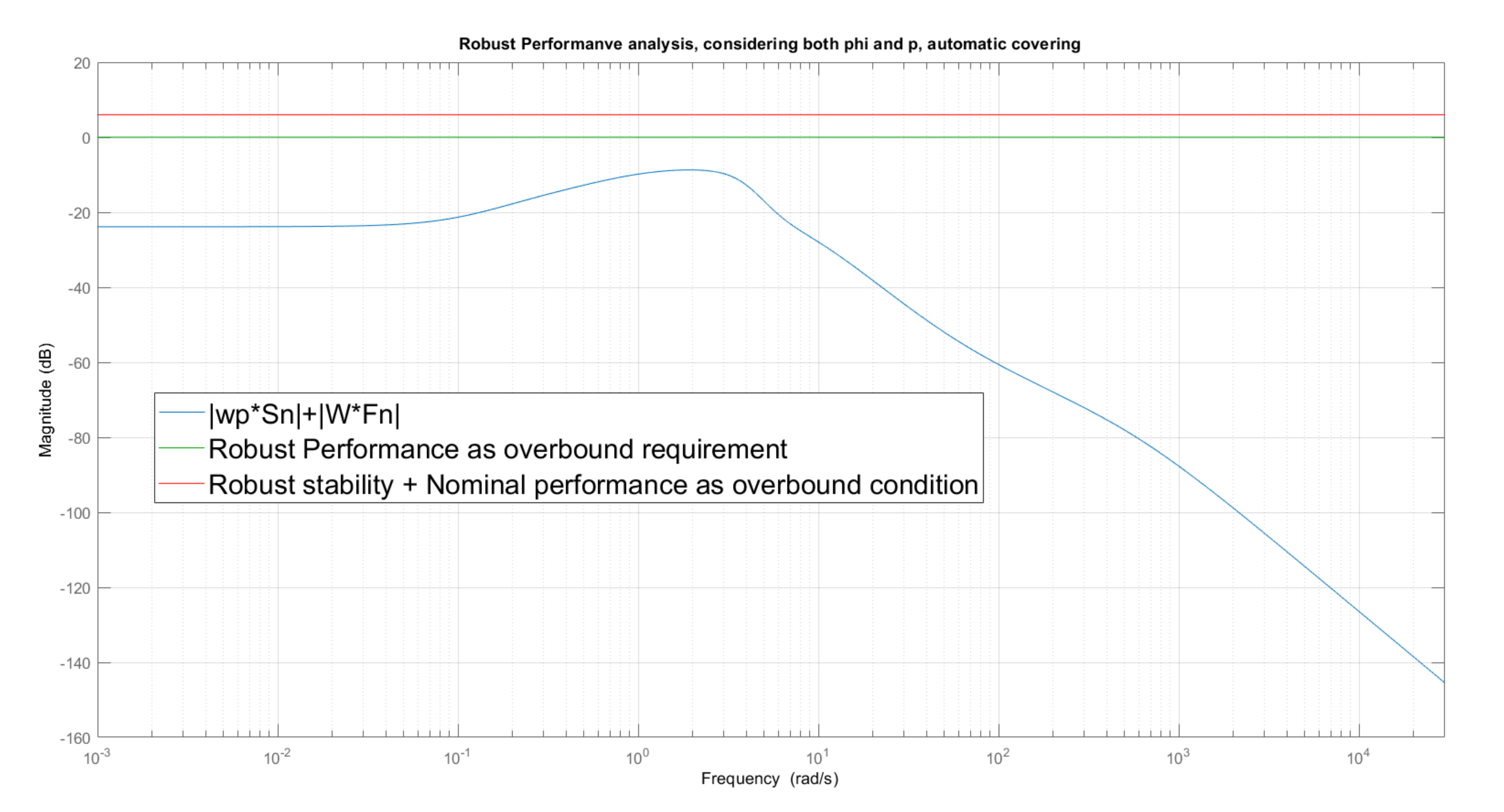

$$|W_p(j\omega)||S(j\omega)| + |W_p(j\omega)||S(j\omega)| < 1 \ \forall\omega$$

- Note that if we sum RS and NP, we obtain:

$$|W_p(j\omega)||S(j\omega)| + |W_p(j\omega)||S(j\omega)| < 2 \ \forall\omega$$

This means that the RP is not guarantee by the RS and NP together. However they are really close, so very often RS and NP guarantee the RP.


Robust Performanve analysis, considering both phi and p, manual covering
Magnitude (dB)
Frequency (rad/s)
|wp*Sn|+|W*Fn|
Robust Performance as overbound requirement
Robust stability + Nominal performance as overbound condition
20
0
-20
-40
-60
-80
-100
-120
-140
-160
$10^{-3}$
$10^{-2}$
$10^{-1}$
$10^{0}$
$10^{1}$
$10^{2}$
$10^{3}$
$10^{4}$




Once again, the RP condition has been imposed only on output $\varphi$

```matlab
%% Robust Performance Analysis
RobustStabiltyErrCover;
close all; clc; clear W1 W2 W_np W_rs ww magNP magRS NN np rs n Rp;

O = tf(1); %to underline 0 dB axis
I = tf(2); %to underline 20log(2) line
ErrorCovering = 'automatic';        % -manual: fatta manuale al primo ordine
                                   % -automatic: fatta automaticamente con
ucover al quarto ordine
switch ErrorCovering
    case 'manual'
    W1 = Wt1;
    W2 = Wt2;
    case 'automatic'
    W1 = Info1.W1;
    W2 = Info2.W1;
end

W_np = Wp*S;
%W_rs = W1*tf(SystemTuned(1,:))+W2*tf(SystemTuned(2,:));% considering both
outputs
W_rs = W2*tf(SystemTuned(2,:)); %considering only phi0 phi

NN = 30000;
ww = 0:0.001:NN;
[magNP,~] = bode(W_np,ww);
[magRS,~] = bode(W_rs,ww);
```

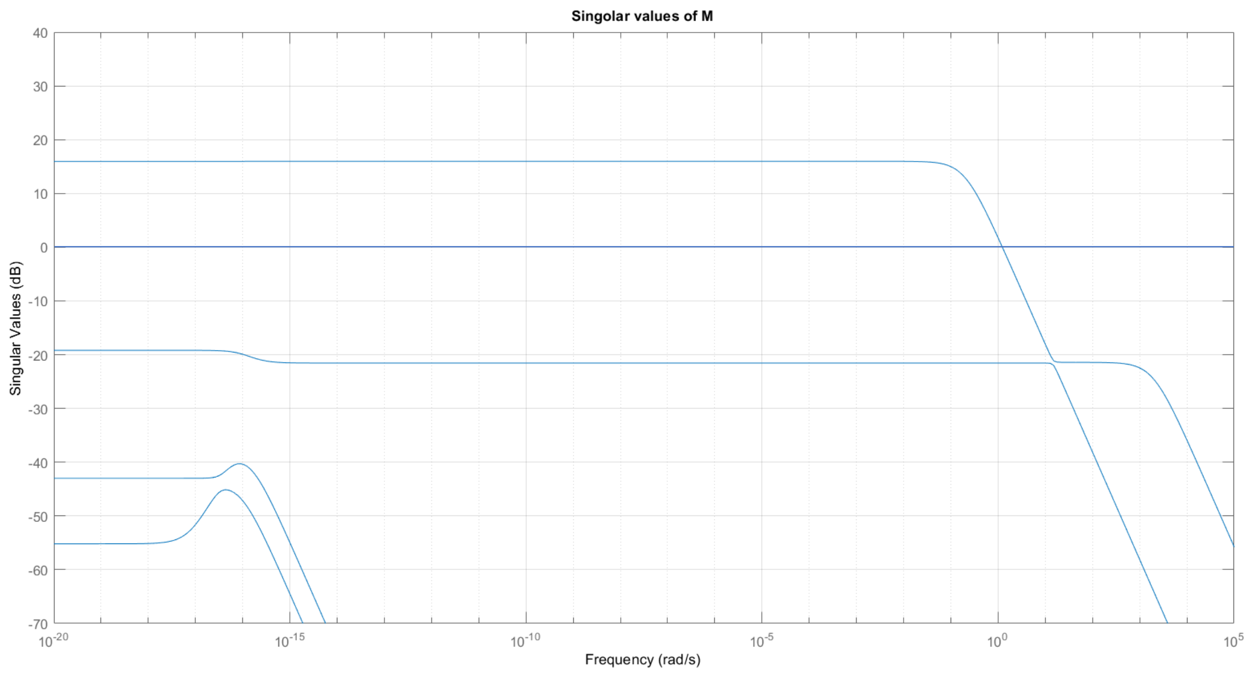
```matlab
np = zeros(NN*1000,1);
rs = zeros(NN*1000,1);
for n=1:(NN*1000+1)
    np(n) = magNP(1,1,n);
    rs(n) = magRS(1,1,n);
end

npdb = 20*log10(np);
rsdb = 20*log10(rs);
Rp = npdb+rsdb;

figure
semilogx(ww,Rp);
hold
bodemag(O,'g',I,'r',{0.001,30000}),grid, title('Robust
Performanve analysis, considering both phi and p, manual
covering'),legend('|wp*Sn|+|W*Fn|',...
    'Robust Performance as overbound requirement','Robust
stability + Nominal performance as overbound condition');
```

# 2.3 Singular Values and Structured Singular Value (μ Analysis)

To evaluate the robust stability of the system, it is also possible to use methods regarding the use of **singular values**: $\bar{\sigma} = \big(M(i\omega)\big) < 1\ \forall\omega \leftrightarrow R.S.$ (assuming $\Delta$ to be full). This condition is very uncommon in reality. The computation of the singular values of M is shown in figure. The condition is not respected, but it's important to notice that this condition is conservative.

Better approaches for evaluating robust stability may be performed using **scaling**s. Artificially computed and imposed to have a less strict condition.

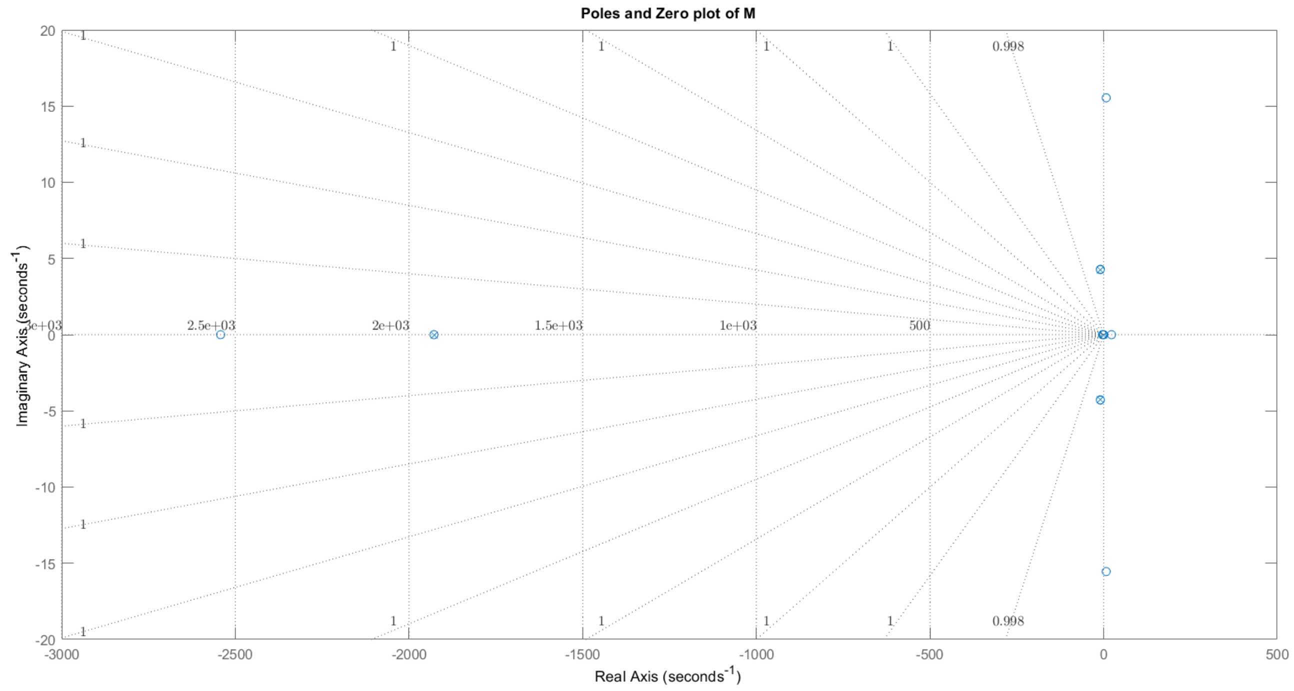
Another possibility is, instead, the use of **structured singular value**.



The structure singular values makes it possible to have an even less strict condition to be respected for R.S.

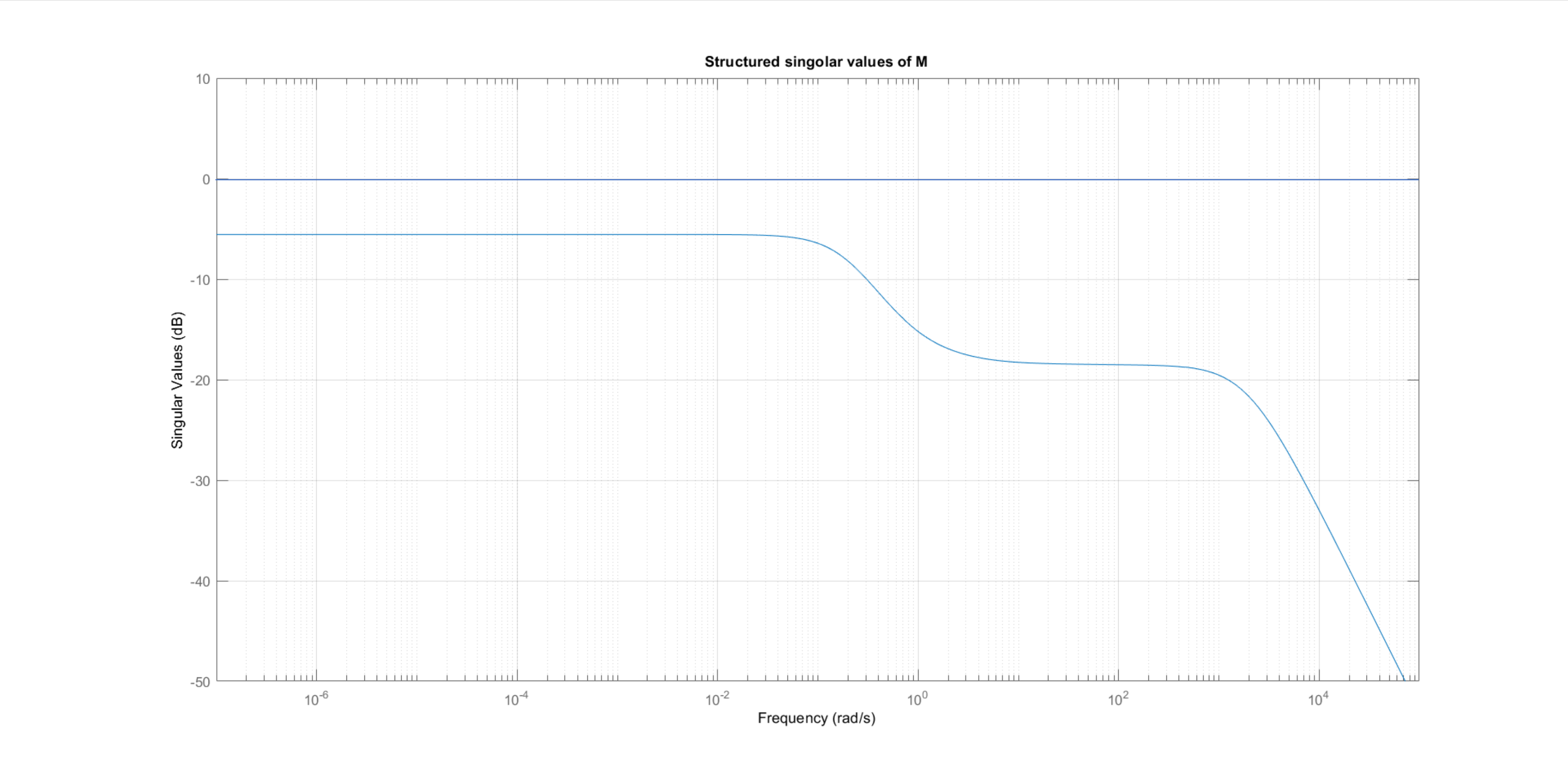
The structure of Δ is now imposed, with a singular value that is less than 1. Hence, it comes the definition of μ. Comparing it with the max gain margin of the loop transfer function that leaves the system to be stable, it is possible to reach the conclusion that:

- Large μ, small gain. With a small perturbation the system loses R.S.
- Small μ, large gain. With a small perturbation the system doesn't lose R.S.




Therefore, it is possible to use a theorem to evaluate the R.S of a system: $\mu\big(M(i\omega)\big) < 1 \;\forall\omega$

In the graph it is possible to see that the condition is now achieved, therefore R.S is guaranteed.



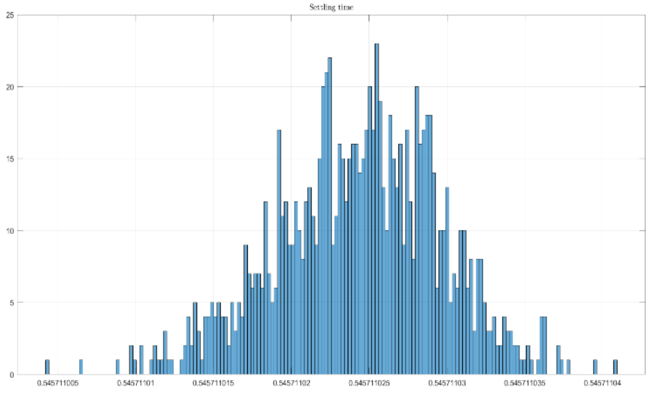

```
%Robust stability using mu
M = ss(M);
omega = logspace(-7,5,2000);
bounds = mussv(frd(M,omega),[1 0;1 0;1 0;1 0]);
```

```
function MDeltaAnalysis(M)

figure
pzplot(M),grid,title('Poles and Zero plot of M') %deve essere AS
per procedere!
isstable(M)
disp('eseguito controllo stabilita'' di M');

%Robust stability analisys using M
figure
sigma(M),grid,title('Singolar values of M'); %MaxSingolarValue
deve stare sotto asse 0 decibel
%Molto conservativo perche' delta non e' full and complex ma
diagonale %condizione necessaria

%Robust stability using mu
M = ss(M);
omega = logspace(-7,5,2000);
bounds = mussv(frd(M,omega),[1 0;1 0;1 0;1 0]);

figure
sigma(bounds),grid,ylim([-50 10]),title('Structured singolar
values of M') %mu deve stare sotto asse 0 decibel %Condizione
necessaria e sufficiente

end
```

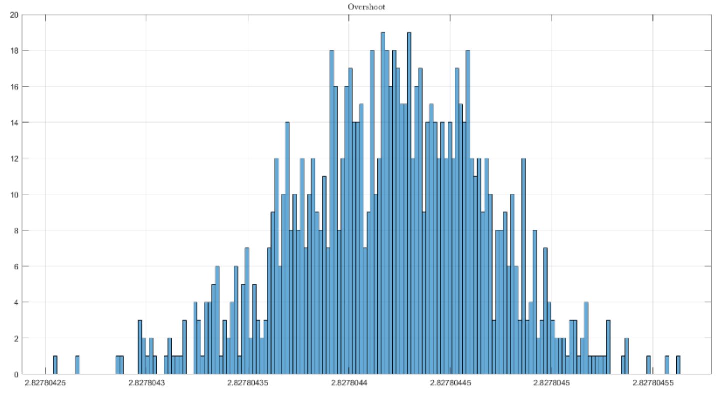
# 3. Uncertain System Verification

## 3.1 Monte Carlo Analysis

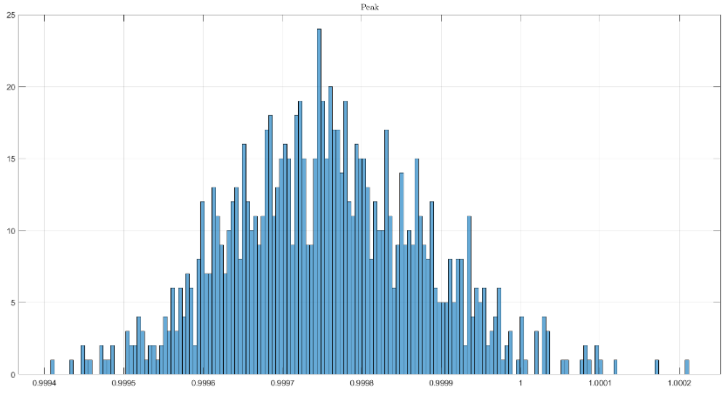
This analysis is performed because of the fact that the system has passed a process of simplification in order for it to be suitable for mathematical computation. The real **detailed model** might introduce losses that may lead to instability, even tough for the simplified model all was stable. To do so, the **Monte Carlo method** comes in hand. Because this is a stochastic verification, it takes into account a Gaussian distribution probability. This is a complete overview of probability, with respect to the deterministic method, which takes into account "only" the 99% of cases. Graphs regarding overshoot, settling time and peak.



Settling time



Overshoot



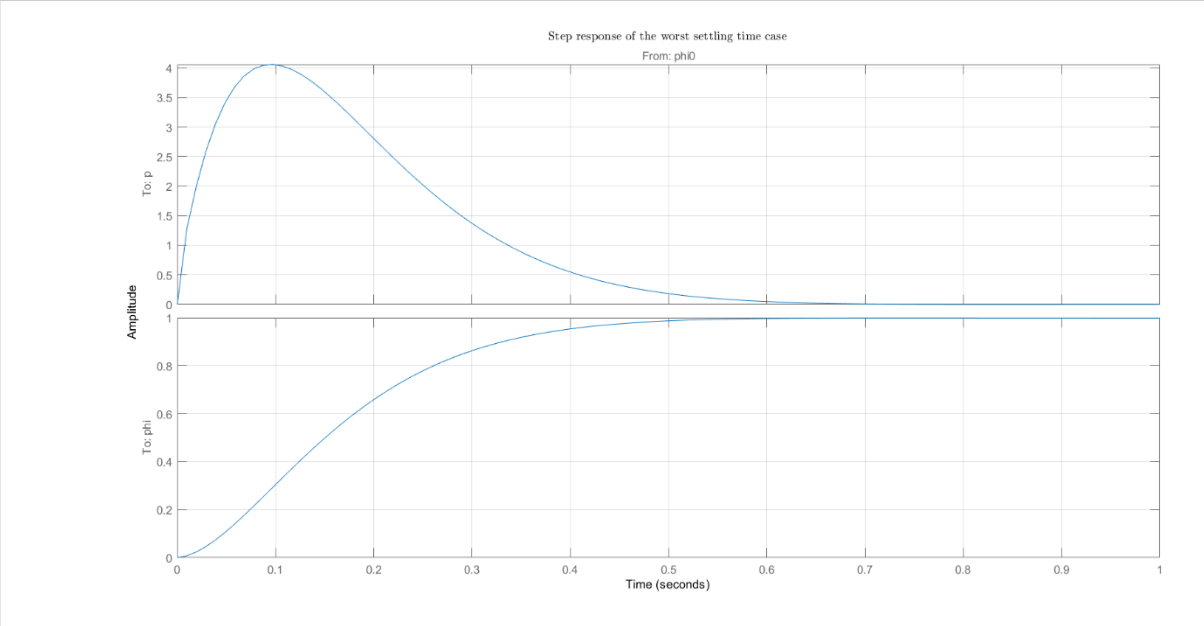
Peak

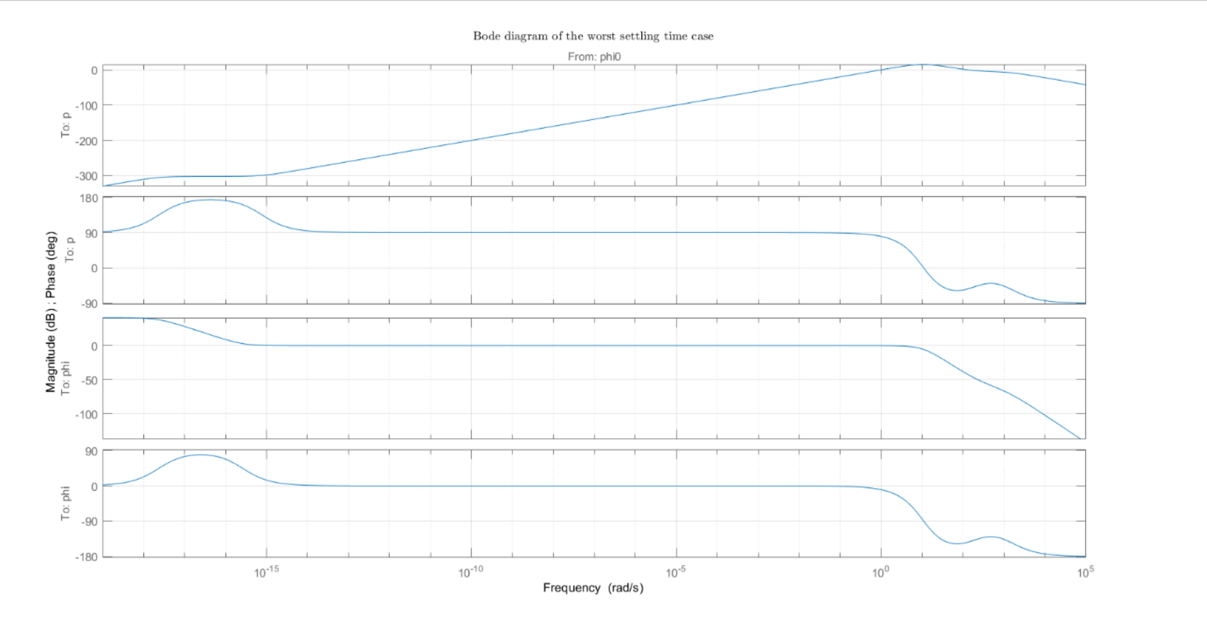
```matlab
%% Monte Carlo Simulation
N=1000; %Number of samples in MC simulation

%Preallocation
YvMC = zeros(N,1);
LvMC = zeros(N,1);
YdMC = zeros(N,1);
LdMC = zeros(N,1);
LpMC = 0; %m/(s*rad)
YpMC = 0; %m/(s*rad)
Yp = 0;
Lp = 0;


Sett2 = zeros(N,1);
Over2 = zeros(N,1);
Peak2 = zeros(N,1);

for n=1:N
    %Resampling of uncertain parameters
    YvMC(n,1) = -0.264 + (-0.264/100)*4.837*randn(1); %1/s
    LvMC(n,1) = -7.349 + (-7.349/100)*4.927*randn(1); %rad*s/m
    YdMC(n,1) = 9.568 + (9.568/100)*4.647*randn(1); %m/s^2
    LdMC(n,1) = 1079.339 + (1079.339/100)*2.762*randn(1); %rad/s^2

    %Construction of resampled uncertain plant
    A_MC = [YvMC(n,1) YpMC g; LvMC(n,1) LpMC 0; 0 1 0];
    B_MC = [YdMC(n,1); LdMC(n,1); 0];
    C_MC = [0 1 0;0 0 1];
    D_MC = [0; 0];

    [Numm, Denn] = ss2tf(A_MC,B_MC,C_MC,D_MC);
    G1MC = tf(Numm(1,:), Denn);
    G2MC = tf(Numm(2,:), Denn);
    GMC = [G1MC; G2MC];
%%Plant Assembling
    GMC.InputName = 'dLat';
    GMC.OutputName{1} = 'p';
    GMC.OutputName{2} = 'phi';

    GMC = c2d(GMC,Ts);
    Sum = sumblk('e = phi0 - phi');

    SystemMC = connect(GMC,Rphi,Rp,Sum,'phi0',{'p','phi'});

    %PERFORMANCES
    % Computation of gain and phase margins
    %[Gm(n),Pm(n)] = margin(P_mc(n)*Cn);

    % Computation of step response
    t=(0:0.004:10);
    opt = stepDataOptions('StepAmplitude',1);
    y1 = step((SystemMC(1,:)),t,opt);
    y2 = step((SystemMC(2,:)),t,opt);

    % Computation of step response characteristics
    S1 = stepinfo(y1,t,1);
    S2 = stepinfo(y2,t,1);

    Sett2(n,1)= S2.SettlingTime;
    Over2(n,1)= S2.Overshoot;
    Peak2(n,1) = S2.Peak;
end
```

```matlab
%% Analysis of the results
% % Plot the histogram of phase margin
% % hist(Pm,100), grid, title('Phase margin')
% % Plot the histogram of gain margin
% % hist(Gm,100), grid, title('Gain margin')

figure
histogram(Sett2,170), grid, title('Settling
time') %Plot the histogram of settling time
figure
histogram(Over2,170), grid, title('Overshoot')
%Plot the histogram of overshoot
figure
histogram(Peak2,170), grid, title('Peak') %Plot
the histogram of Peak
```

A useful result of the Monte Carlo analysis is the possibility to study the so-called worst-case **scenarios**. To extract it (in the case of the worst settling time):

```
% find MC case corresponding to worst settling time
max_Sett2=max(Sett2);          %max settling time
worst=find(Sett2==max_Sett2);  %associated number
```



Thank you for your attention and happy controlling! ☺