Self Directed Project

Harsh Meel, hm642
MAE 5780 - Feedback Control Systems
Cornell University
Prof. MacMartin
18 December 2025

Table of Contents

Dynamics of the System

Dynamic equations of the physical system and parameters:

For the self directed course problem, I decided to solve a simplified version of the Climate engineering feedback control problem. I have used the Box-diffusion model defined in the paper titled Dynamics of the coupled human–climate system resulting from closed-loop control of solar geoengineering (MacMartin,2013).

Here are the dynamic equations of the system:

$$C\,\frac{dT}{dt} = F - \lambda T + \beta \frac{\delta T_d}{\delta z}\Big|_{z=0} \qquad (1)$$

$$\frac{\delta T_d}{\delta t} = \kappa \frac{\delta^2 T_d}{\delta z^2} \qquad (2)$$

Here the variables are ,
Radiative Forcing F(t), Surface temperature T(t), Deep ocean temperature $T_d(z,t)$ with the boundary condition T_d (0,t) = T(0,t) (taking the top of the deep ocean as z=0).

The parameters are:
λ describes the natural climate feedback (the change in radiation due to a change in surface temperature), C = $c\rho H$ is the surface layer heat capacity per unit area, κ the thermal diffusivity, and $\beta = c\rho\kappa$ for density ρ and specific heat capacity c.

Feedback Loop of the system:

For the sake of our project, following is the block diagram used:

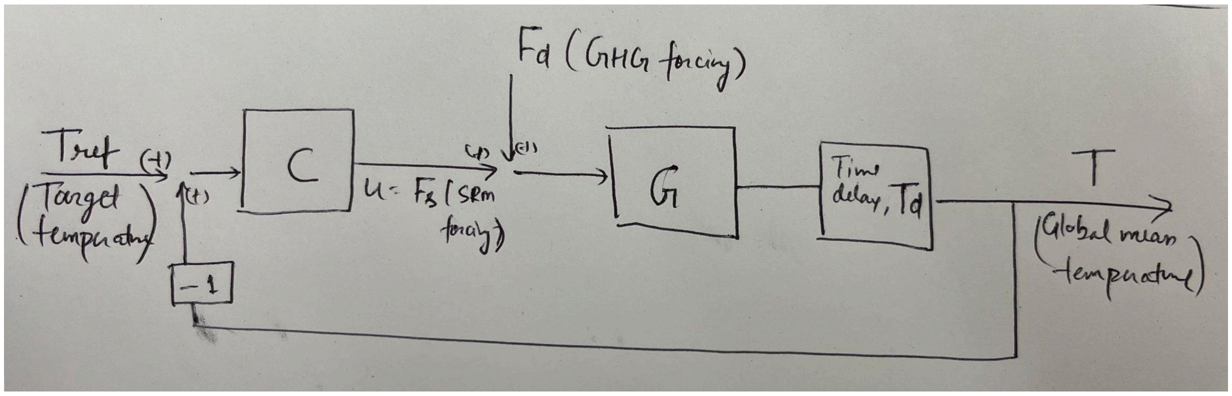


Figure 1 : Feedback Control Block Diagram in frequency domain

Here,

$T = G(F_d + F_s)$
$F_s = -1 * C * (T-T_{ref}) * T_d$

F_d = Green house gas forcing (This is expected to be a ramped input signal for this project. More on this later)
F_s = SRM forcing from the feedback loop
T_d = Time delay transfer function

Transfer functions:

For the plant transfer function G(s), we take the calculated value in frequency domain from the paper.

$$G(s) = \frac{T(s)}{F(s)} = \frac{1}{\lambda + \beta(s/\kappa)^{1/2} + Cs} \qquad (3)$$

For the parameter values, the paper calculates them by comparing equation 3 with the calculated frequency response from HadCM3L (a climate model) .
This yield the the best fit to the calculated frequency response yields λ **= 1.2 W m^{-2} K^{-1},** τ **=** $\beta^2/\lambda^2\kappa$ **= 13 years, and C = 3.2 x 10^6 J m^{-2} K^{-1}**

For the time delay, it has to be understood what causes it. For our problem, we calculate the reference temperature T_{ref} to be tracked by taking the mean of global temperature over the last N years. If SAI is deployed today with the average information of last N years, the system is always reacting to the value of state approximately N/2 years ago. Therefore, there is a time delay of N/2 years. **We will take N =1 year for our calculations.**

$T_d(s) = e^{-N/2s}$ (transfer function of time delay in frequency domain) (4)

With these values, we are set to stabilize the Tref , hence intending to stabilize the global mean temperature against greenhouse gas forcing or any other disturbance. We will be using MATLAB for our control design, using the SISO frequency domain control design methods.

Code:
```
%% frd method
clear;clc;
%year to seconds
% Convert time from years to seconds
yts = 365 * 24 * 3600;  % Time from years to seconds
% Parameters (SI units where applicable)
lambda = 1.2;        % W/(m^2·K)           — natural climate feedback
C      = 3.2e6;      % J/(m^2·K)           — heat capacity
betabysqrtk = sqrt(13*yts)*lambda;  % W/(m^2·K)
N =1; %years of data taken into account
Td = N/2 *yts; %time delay in seconds
```

Note:

Next, we need to define the transfer function of the plant and controller etc. Here, given the presence of $s^{0.5}$ in the plant's transfer function, we cannot use the tf('s') command in the matlab. This issue can be addressed by using the frd model with a numerical array of frequency in the concerned range.

Following is the range we used with the model:

Code:

```
% Choose a frequency grid (rad/s). Adjust to the band of interest.
w = logspace(-9, -6, 600);       % rad/s

% Complex frequency response of G(jw) with principal branch of sqrt
Hjw = 1 ./ ( lambda + betabysqrtk .* sqrt(1j*w) + C .* (1j*w) ); %rad per second

% Build the FRD model (continuous time, frequencies in rad/s)
G_frd = frd(Hjw, w);

% Frequency-domain analysis (FRD supports these)
figure(1);
h = bodeplot(G_frd);
% Set the FrequencyUnit property of the plot handle
h.FrequencyUnit = 'cycles/year'; % cycles/year
```

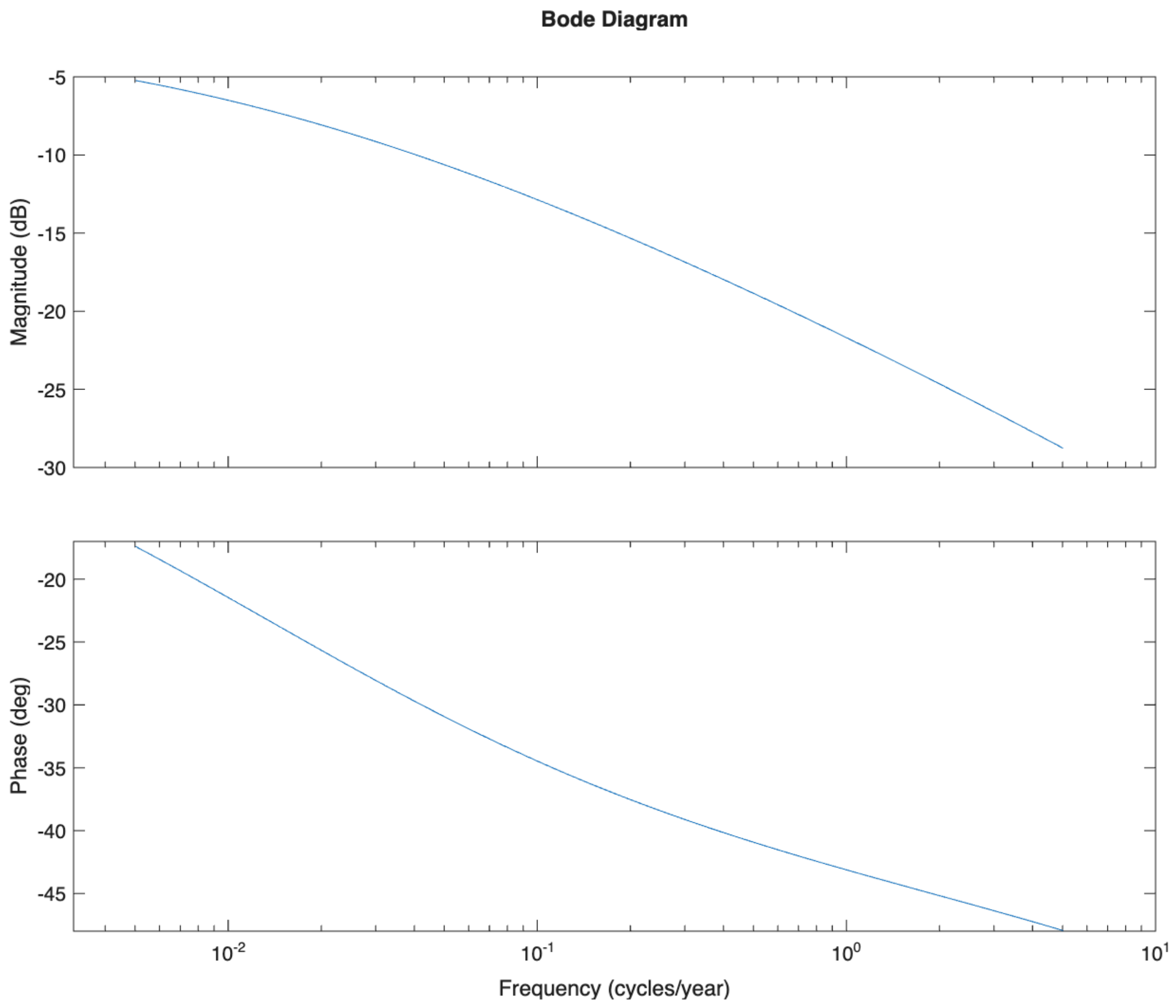


Figure 2: Bode plot of the Plant G(s)

Note : The display unit in time is chosen to be cycles/year assuming SAI deployment happens on the scale of a few times in many years.

Next we decide the performance requirements.

Performance Requirements

For this problem the chosen performance requirements are:

1. Steady State error in keeping the T_{ref} should be less than 1%

   Here the error has to be stabilized against an ramped disturbance $F_{d.}$ For the case of simplicity, we defined F_d as:

   $F_d(t) = t$

2. For frequency till 1/10 years^{-1} , the tracking error should be less than 10%.

   This was decided based on the expected time it takes for SAI deployments to take place and effects to happen. A good value for this could be once in every ten years.
   Also, the usual increase of global mean temperature which can cause issues is about 2^0 C from pre industrial. Thus, a 0.1^0 error by 10% is quite good tracking.

3. Phase Margin $>30^0$

   This is mostly to have a lesser overshoot and system response . A high overshoot can lead to unnecessary changes in global mean temperature.

4. Gain Margin > 2

   This is to meet conventional gain margin requirements.

Controller design

First a proportional controller was applied to make the system stable against time delay.

Code:
```
%% now adding time delay
Tdjw = exp(-1*1j*w*Td);
Td_frd = frd(Tdjw,w);
figure(2);
h2 = bodeplot(G_frd*Td_frd);
h2.FrequencyUnit = 'cycles/year';
%% adding a proportional control
Kp = 10;
figure(3);
margin(G_frd*Td_frd*Kp);
%h3.FrequencyUnit = 'cycles/year';
title('Proportional control bode')
```

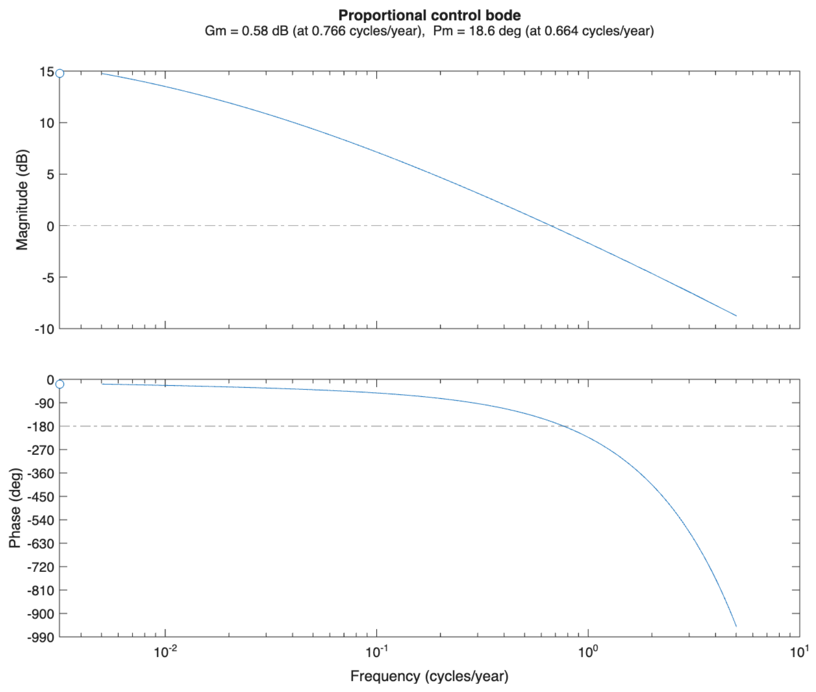
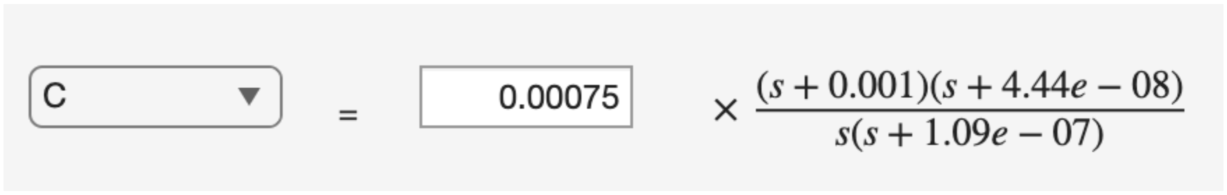


Figure 3: Open Loop Bode of the stable system

Next, to meet the requirements 1 and 2. We added an integrator and then to keep the phase and gain margin, we used compensators.

This design was done using the SISO tool in Matlab.
Here is the Controller 1.



$$C = 0.00075 \times \frac{(s+0.001)(s+4.44e-08)}{s(s+1.09e-07)}$$

Figure 4 : Controller 1, a lead and a lag

Here is the bode plot of the open loop transfer function for Controller 1:

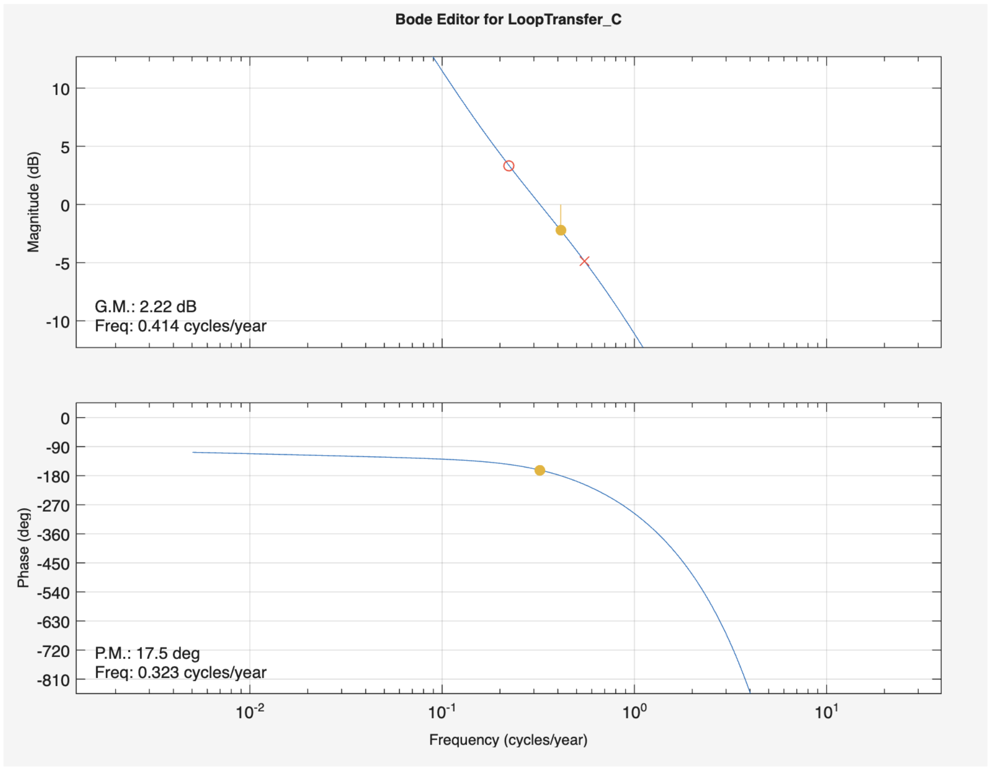


Figure 5 : Open loop transfer function with Controller 1 with Gain and Phase Margin

And here is the step response from reference to output:

Code:

```
%% getting transfer function of reference to output signal
r2y = ssest(G_frd*Td_frd*C_siso_1/(1+G_frd*Td_frd*C_siso_1));
```

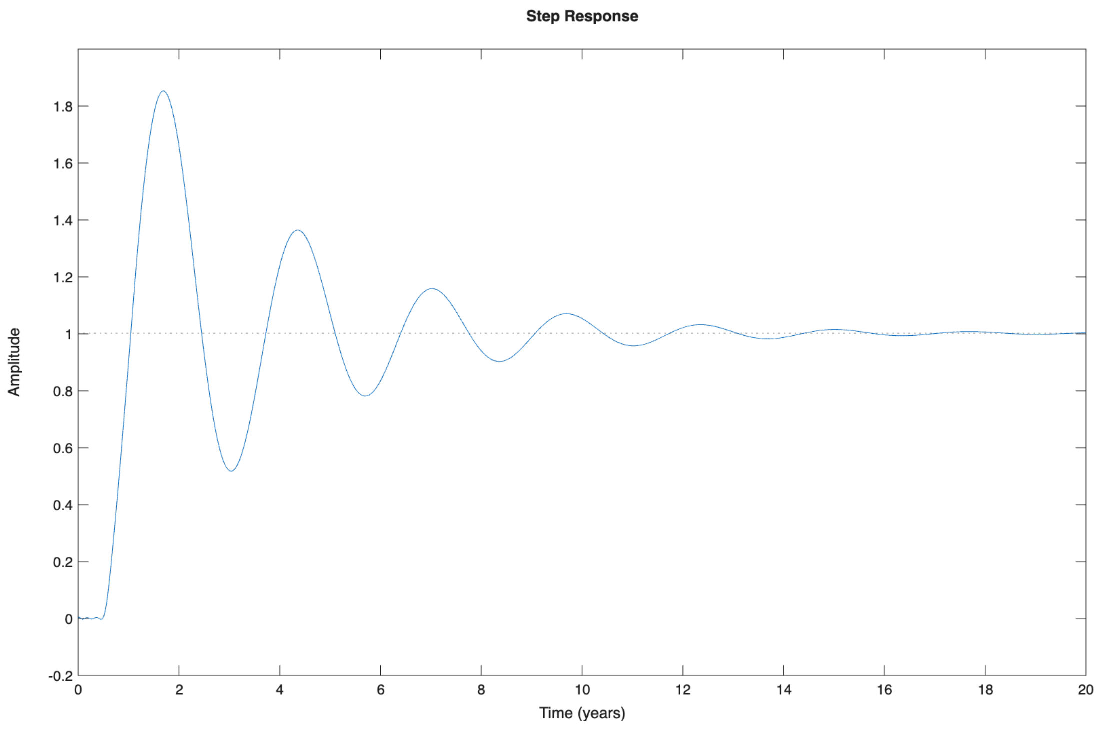


Figure 6 : Step Response from Reference to Output for Controller 1

The controller exactly tracks the reference signal.

Next the response to the ramped disturbance F_d to output is checked for Controller 1:

```
%% getting transfer fun from disturbance to output signal
d2y = ssest(G_frd*Td_frd/(1+G_frd*Td_frd*C_siso_1));
```

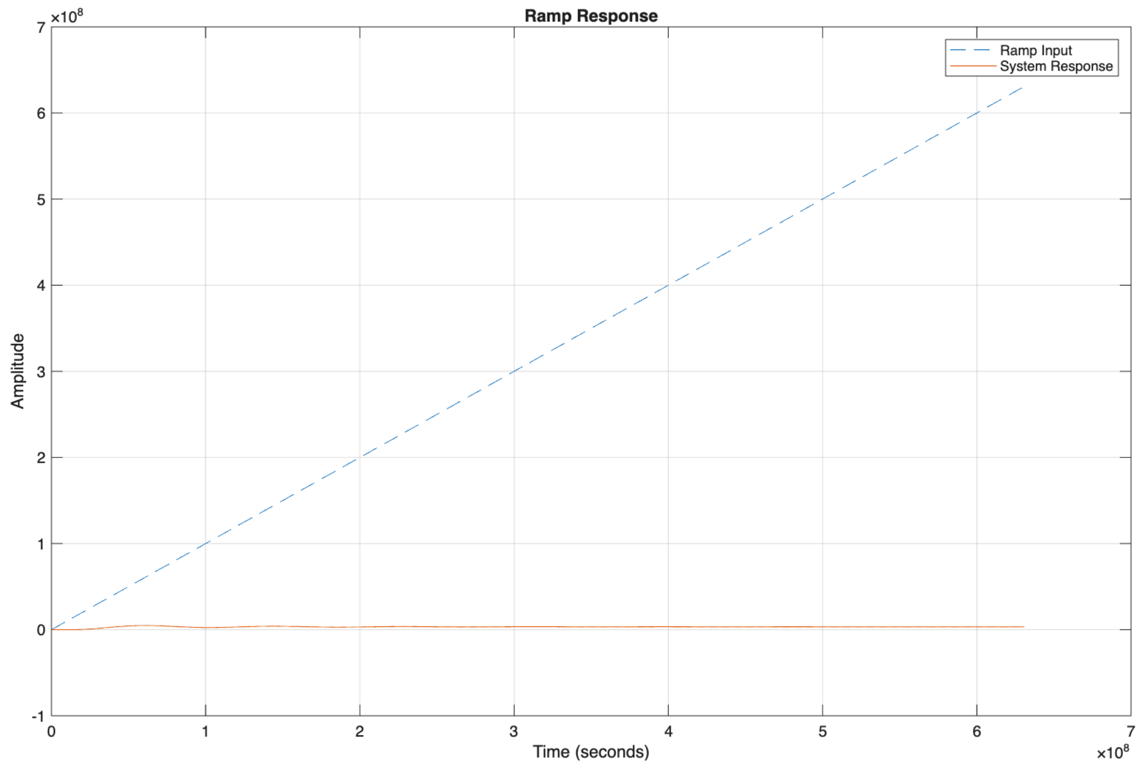


Figure 7: Ramp response from disturbance to output for Controller 1 for 20 years

Code:

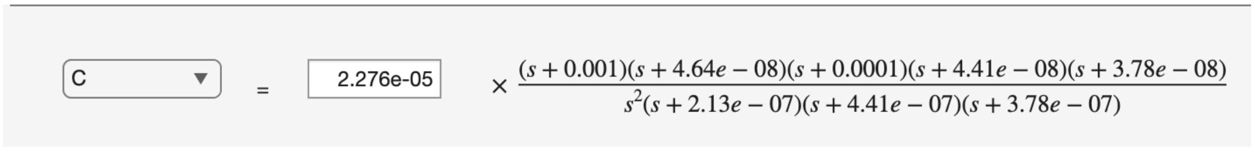
```
%% for ramp input
% Define the time vector and the ramp input signal (e.g., slope of 1)
t = 0:0.01*yts:20*yts; % Time from 0 to 20 year with 0.01 year intervals
ramp_input = t; % For a unit ramp (slope = 1)

% Simulate the system response
[y, t, x] = lsim(d2y, ramp_input, t);
% Plot the response
figure(7);
plot(t, ramp_input, '--', t, y, '-');
title('Ramp Response');
xlabel('Time (seconds)');
ylabel('Amplitude');
legend('Ramp Input', 'System Response');
grid on;
```

The controller keeps the output temperature at reference as the disturbance increases in magnitude.

However, the Controller 1 doesn't lead to zero steady state error for ramped disturbance rejection (look closely at the graph) and also fails to satisfy the phase margin requirement.

So another integrator with a zero is added to satisfy the steady state error requirements with the addition of three lead compensators to satisfy the phase margin requirements, especially after the addition of the integrator.

Controller 2:



C = 2.276e-05 × $\frac{(s+0.001)(s+4.64e-08)(s+0.0001)(s+4.41e-08)(s+3.78e-08)}{s^2(s+2.13e-07)(s+4.41e-07)(s+3.78e-07)}$

Bode of the open loop transfer function:

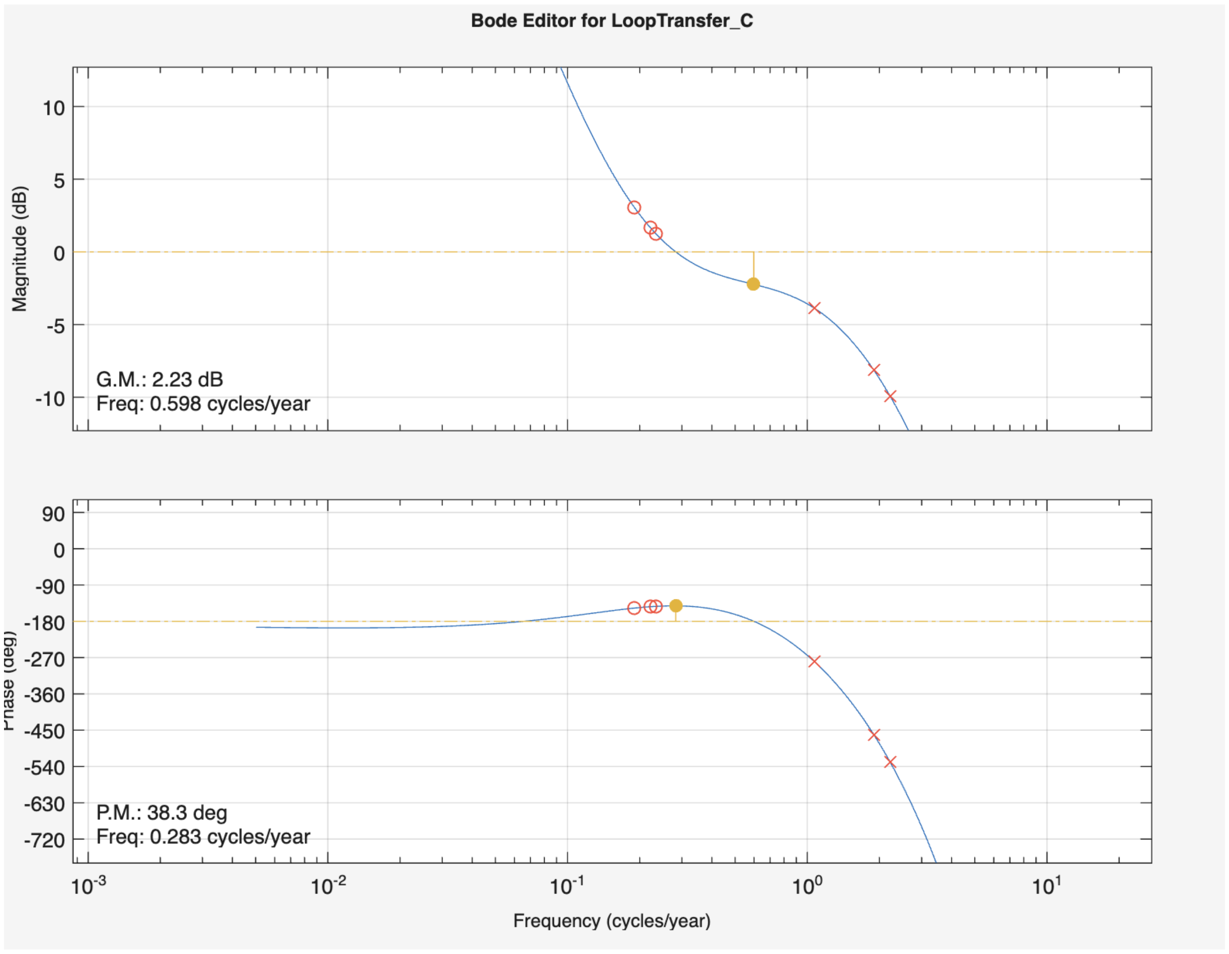


Figure 8 : Open Loop transfer function with Controller 2 with Gain and phase margin

As one can see, the requirements 1. bandwidth , 3. phase margin and 4. gain margin is met.

Step response of reference to output for controller 2:

Code:
```
%% getting transfer function of reference to output signal
r2y = ssest(G_frd*Td_frd*C_SISO_ramp/(1+G_frd*Td_frd*C_SISO_ramp));
step(r2y)
```

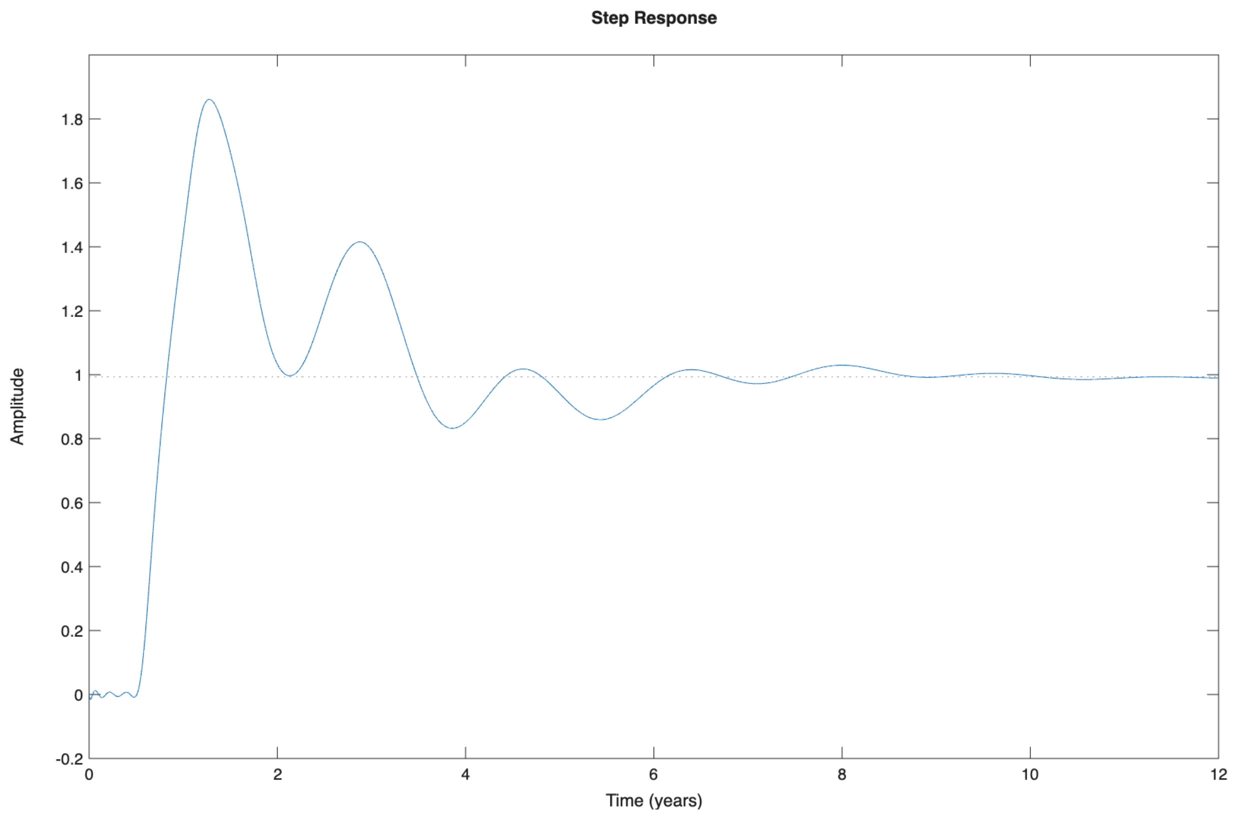


Figure 9 : Step Response of Controller 2 for unit change in reference

As seen here, **the settling time (2%) has improved from ~13 years to 7 years.**

Looking at the response to ramped disturbance:

Controller 2 satisfies requirement 2. Steady state error <1% for ramped disturbance by making it zero (look closely at the system response in Figure 10).

Code:
```
%% getting transfer fun from disturbance to output signal
d2y = ssest(G_frd*Td_frd/(1+G_frd*Td_frd*C_SISO_ramp));
```

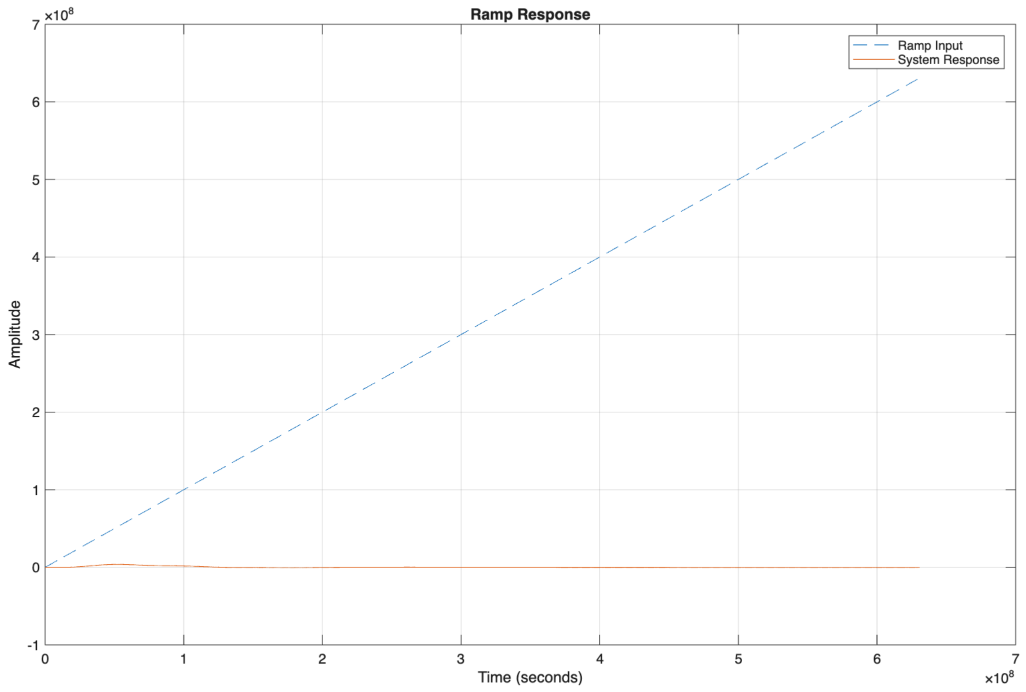


Figure 10: Output Response to ramped GHG disturbance for Controller 2 for 20 years

Let's have a look at the Control signal against the ramped response:

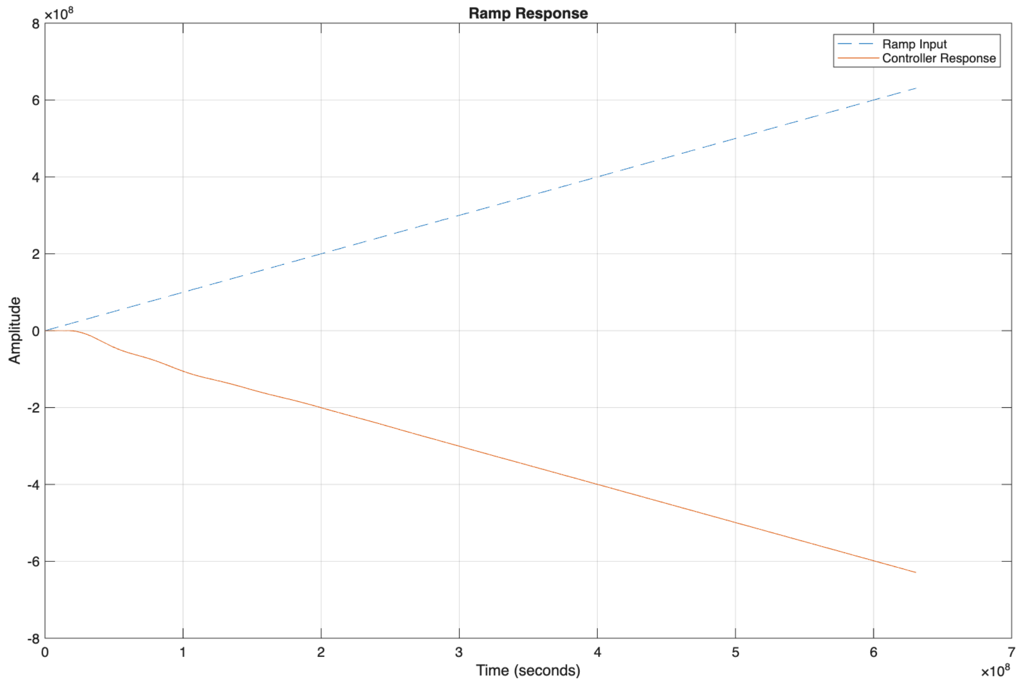


Figure 11: Controller 2's response to ramped GHG disturbance for 20 years

Code:

```
%% getting transfer function of disturbance to control signal
d2c = ssest(-G_frd*Td_frd*C_SISO_ramp/(1+G_frd*Td_frd*C_SISO_ramp));

%% ramp response from distrubance to control signal
% Define the time vector and the ramp input signal (e.g., slope of 1)
t = 0:0.01*yts:20*yts; % Time from 0 to 20 year with 0.01 year intervals
ramp_input = t; % For a unit ramp (slope = 1)
[y, t, x] = lsim(d2c, ramp_input, t);
% Plot the response
figure(8);
plot(t, ramp_input, '--', t, y, '-');
title('Ramp Response');
xlabel('Time (seconds)');
ylabel('Amplitude');
legend('Ramp Input', 'Controller Response');
grid on;
```

Saturation:

From Figure 11, one notes that **there is a ramping in the controller signal to compensate for the ramping of greenhouse gases.** This ramping in control signal won't be possible in the real world deployment.
Like any controller, **SAI cannot provide control input indefinitely and after a point the controller will saturate**. For SAI the saturation happens when the aerosol density is so high that the aerosols undergo coagulation and become heavier. These aerosols then drop down the atmospheric layer and SAI forcing stops. Hence there is a limit to the extent of greenhouse gas forcing SAI can compensate for.

References:

MacMartin, Douglas G. “Dynamics of the coupled human–climate system resulting from closed-loop control of solar geoengineering.” vol. 45, 2013, pp. 243-258, https://link.springer.com/article/10.1007/s00382-013-1822-9.