July 2021

01211433 Homework 3

Requirement : Julia with ControlSystems, and Plot packages. To install, in the Julia REPL:

```
using Pkg
Pkg.add("ControlSystems")
Pkg.add("Plots")
```

Packages used.

```
using ControlSystems    , Plots
```

Problem: let us design a controller for our same old robot joint driven by DC motor developed since the first module

\begin{equation} P(s) = \frac{1}{7s^2 + 0.05s} \tag{1} \end{equation}

with the following design specs

- steady state error is eliminated
- low frequency disturbance is attenuated at least 0.05 below 0.5 rad/s
- high frequency measurement noise is attenuated 0.15 above 80 rad/s
- closed-loop stable, with phase margin at least 50 degrees

From the above discussion, this can be translated to stability and performance bounds

- –> $L(s)$ has an integrator. Note that $P(s)$ already has one
- –> $|S(j\omega)| \leq ?\ dB \rightarrow |L(j\omega)| \geq ?\ dB$ below 0.5 rad/s
- –> $|T(j\omega)| \leq ?\ dB \rightarrow |L(j\omega)| \leq ?\ dB$ above 80 rad/s
- –> $L(j\omega)$ has at least 50 degrees phase margin

To aid this design problem, we write a function lshape() in the cell below.

```
lshape (generic function with 1 method)
```

```julia
function lshape(C, P, lf, lfb, hf, hfb, pm)
    vecsize = 1000
    L = C*P
    #S = 1/(1+L)
    #T = L/(1+L)
    lf_log10 = log10(lf)
    hf_log10 = log10(hf)
    w_start = floor(lf_log10) - 1
    w_end = ceil(hf_log10) + 1
    w = exp10.(LinRange(w_start, w_end, vecsize))
    #bodeplot(L,w; title="Bode plot of L(s)",label="\$L(s)\$")

    # frequency response of L
    Lmag, Lph, Lom = bode(L,w)
    Lmag_db = 20*log10.(Lmag)

    # create bound vectors
    lf_bnd = ifelse.(w.<lf,lfb, 0)
    hf_bnd = ifelse.(w.<hf,0, hfb)

    # check whether violation occurs
    lf_idxv = findall(x->x>lf,w)
    lf_idx = lf_idxv[1] # index of low-freq region
    hf_idxv = findall(x->x<hf,w)
    hf_idx = hf_idxv[end] # index of high-freq region
    lfmag = Lmag_db[1:lf_idx]
    hfmag = Lmag_db[hf_idx:end]
    if minimum(lfmag)<lfb
        lf_legend = "LF bound (violated!)"
    else
        lf_legend = "LF bound"
    end
    if maximum(hfmag)>hfb
        hf_legend = "HF bound (violated!)"
    else
        hf_legend = "HF bound"
    end
    # desired phase margin line
    pmvec = (pm -180)*ones(vecsize)

    # compute gain/phase margin
    wgm, g_margin, wpm, ph_margin = margin(L)
    ph_at_crossover = (ph_margin.-180)

    # loophaping plot
    lmag_db = dropdims(Lmag_db,dims = (2,3))

    #return w, lmag_db
    gr()
    lmagplot = plot(w,lmag_db,xaxis=:log, label="|L(s)|",legend=:bottomleft)
    plot!(w,lf_bnd,xaxis=:log, label=lf_legend)
    plot!(w,hf_bnd,xaxis=:log, label=hf_legend,xlabel="frequency
(rad/s)",ylabel="magnitude (dB)",title="Bode plot of L(s) v.s. bounds")
    lph = dropdims(Lph, dims = (2,3))
    lphplot = plot(w,lph,xaxis=:log, label = "\$\\angle L(s)\$",legend=:bottomleft)
    plot!(w,pmvec,xaxis=:log, label = "lower bounds for PM")
    ph_margin_int = round(ph_margin[1])
    if ph_margin_int > pm
        ph_legend = "phase margin = $ph_margin_int deg."
    else
        ph_legend = "phase margin = $ph_margin_int deg. (violated!)"
    end
    plot!(wpm, ph_at_crossover, seriestype =:scatter, label=ph_legend)
```

```
    Lplot = plot(lmagplot,lphplot,layout=(2,1))


end
```

where the arguments are as follows:

- C : Controller
- P : Plant
- lf : Define low frequency region from 0 - lf (rad/s)
- lfb : Low frequency bound for $L(s)$
- hf : Define high frequency region from hf - ∞ (rad/s)
- hfb : High frequency bound for $L(s)$
- pm : Phase margin (degrees)

```
(LFbounds = 26.0206, HFbounds = -16.4782)
```

```
begin
    S_bnd_abs = 0.05
    T_bnd_abs = 0.15
    S_bnd = 20*log10(S_bnd_abs)
    L_bnd = -S_bnd
    T_bnd = 20*log10(T_bnd_abs)
    (LFbounds = L_bnd, HFbounds = T_bnd)
end
```

```
TransferFunction{Continuous, ControlSystems.SisoRational{Float64}}
     1.0
--------------
7.0s^2 + 0.05s

Continuous-time transfer function model
```

```
# create plant transfer function
begin
    s = tf("s")
    P = 1/(7*s^2+0.05*s)
end
```

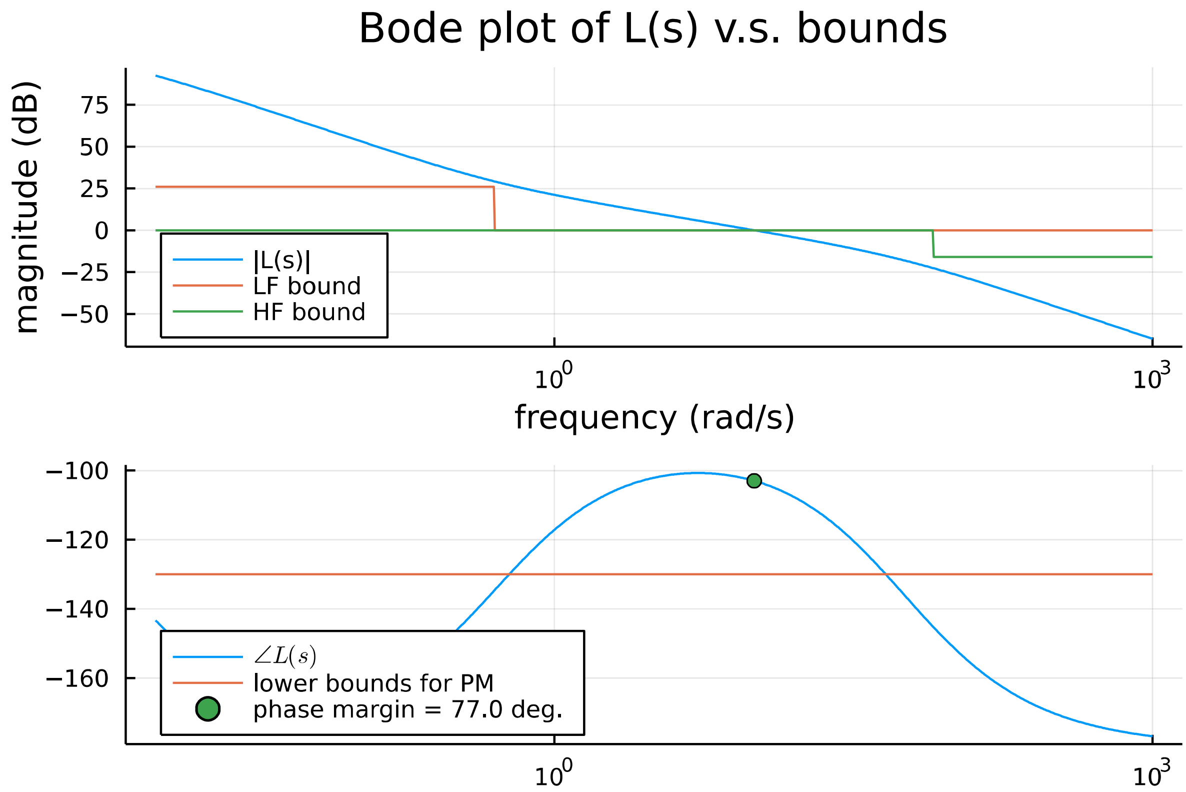
```
50
```

```
begin
    lf = 0.5# low frequency here
    lfb = 26# your answer
    hf = 80# high frequency here
    hfb = -16# your answer
    pm = 50 # phase margin
end
```

```
C = TransferFunction{Continuous, ControlSystems.SisoRational{Float64}}
    4000.0s + 2000.0
    ----------------
      1.0s + 56.0

    Continuous-time transfer function model
```

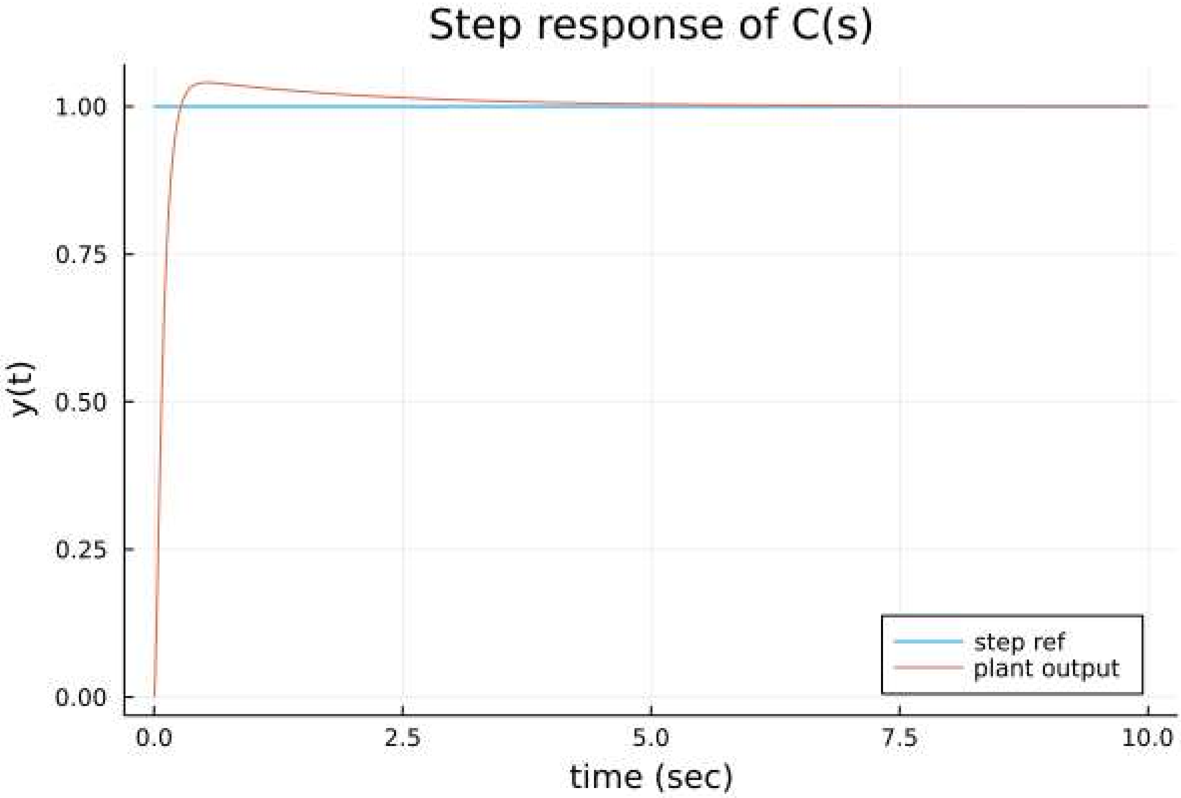
```
C = 4000*(s+0.5)/(s+56) # your controller
```




```
lshape(C, P, lf, lfb, hf, hfb, pm) # call loopshaping funciton
```

Iterate the above cell until you achieve a controller that meets the specs.

Then plot the closed-loop step response (adjust tvec if necessary)




```julia
begin
    L = C*P
    T = minreal(L/(1+L))
    tvec1 = collect(Float64,0:0.001:10)
    y1,t1,x1 = step(T,tvec1)
    r1 = ones(size(t1))
    plot(t1,r1, label="step ref")
    plot!(t1,y1, label="plant output",xlabel="time (sec)",ylabel="y(t)",title="Step response of C(s)",legend=:bottomright)
end
```

To make sure that this design meets all the specification, the disturbance and noise responses in the time-domain need to be evaluated. Create a function to plot output response of arbitrary function.

```
plot_response (generic function with 1 method)
```

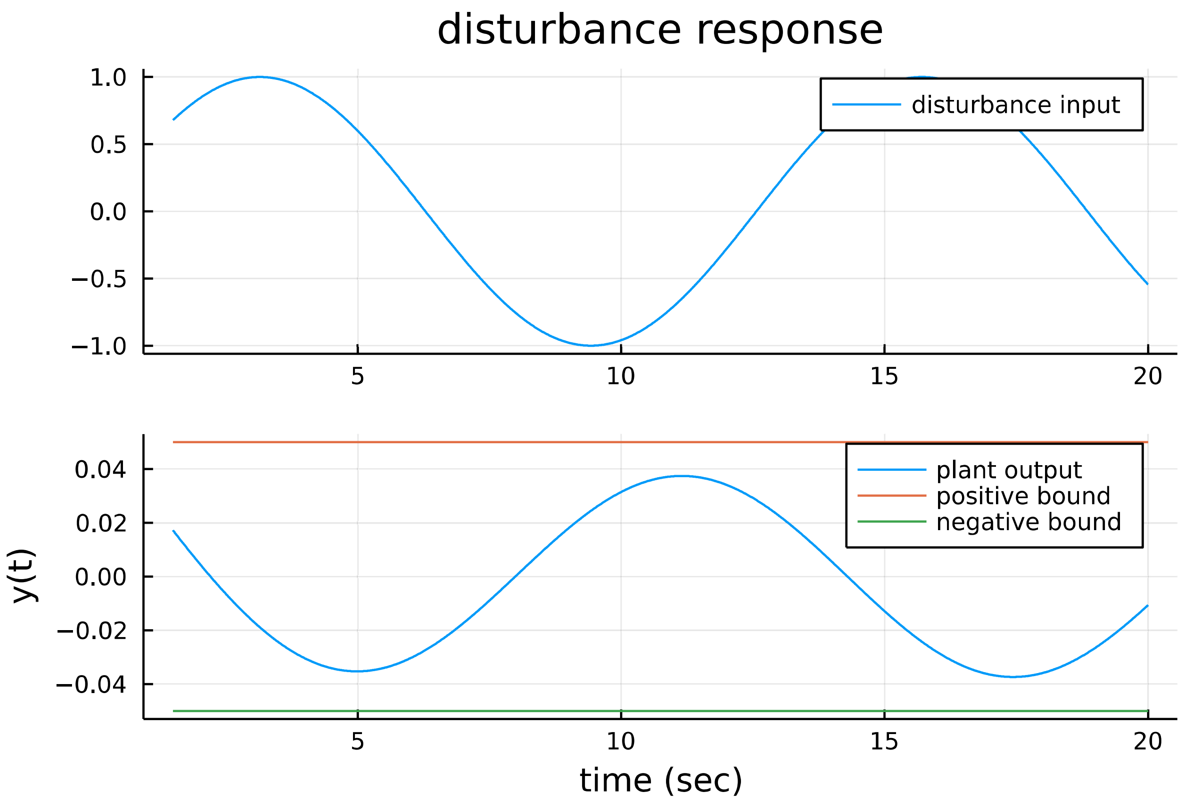
```julia
function plot_response(sys,u,t,ampbnd, plottitle)
    y, tout, x = lsim(sys, u, t,method=:zoh)
    t_idx = 150  # get rid of transient
    tout1 = tout[t_idx:end]
    y1 = y[t_idx:end]
    u1 = u[t_idx:end]

    pbnd = ampbnd*ones(size(tout1))
    nbnd = -ampbnd*ones(size(tout1))
    uplt = plot(tout1,u1, label = plottitle*" input",title=plottitle*" response")
    yplt=plot(tout1,y1, label = "plant output")
    plot!(tout1,pbnd, label = "positive bound")
    plot!(tout1, nbnd, label = "negative bound",xlabel="time (sec)",ylabel="y(t)")
    plot(uplt, yplt, layout=(2,1))

end
```

From the design specifications, the required disturbance attenuation is at least 0.05 for frequency below 0.5 rad/s. Since the attenuatiton is the least at $\omega = 0.5$ rad/s, we use this frequency as our test point.

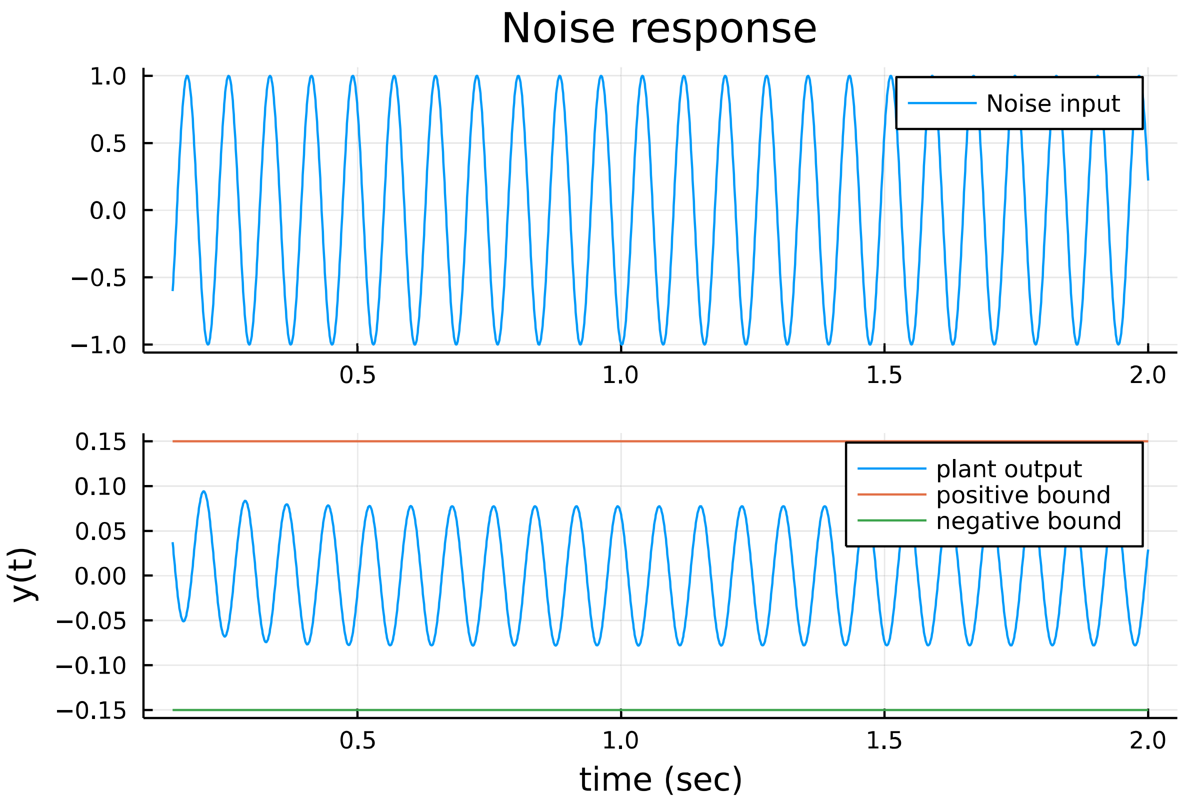
Recall that the closed-loop transfer function for the output disturbance response is $S(s)$. The plot from this cell must confirm that the controller meets this attenuation performance.




```
begin
    df = 0.5 # specify disturbance frequency in rad/s
    S = minreal(1/(1+L))
    tvec2 = collect(Float64,0:0.01:20)
    u2 = sin.(df*tvec2)
    abnd = 0.05
    plot_response(S,u2,tvec2,abnd,"disturbance")
end
```

Note : The output magnitude must swing within 0.05

Use the same plot function on the complementary sensitivity $T(s)$ to verify that, with a noise input $u(t) = \sin(\omega t)$ where $\omega = 80$ rad/s, the output should swing within ± 0.15 unit.




```
begin
    nf = 80 # specify noise frequency in rad/s
    tvec3 = collect(Float64,0:0.001:2)
    u3 = sin.(nf*tvec3)
    nbnd = 0.15
    plot_response(T,u3,tvec3,nbnd,"Noise")
end
```