

pelicano_serial

April 28, 2018

1 Parâmetros

Aqui faço a escolha de alguns parâmetros que serão utilizados bem como importo bibliotecas que serão utilizadas.

```
In [191]: include("../comum.jl")
          include("../modelos.jl")
          using Evolutionary, Plots;
          pyplot();

In [192]: Ts      = 0.05 # Intervalo entre leituras da saída
          tend    = 2.0  # tempo final para estabilização
          t0      = 0.0  # instante inicial
          r1      = 0.6  # referência junta 1
          r2      = 0.8  # referência junta 2
          xr = [r1, r2]
          popul   = 50   # população
          iterac  = 15;  # iterações
          = 10.      # parâmetro para o erro
          = 0.01     # parâmetro para o jerk
          = 0.1      # parâmetro para o torque
          per = 1/2    # início da leitura do vetor a parti de per do comprimento total
          kp_end = AbstractFloat[]
          kv_end = AbstractFloat[];
```

2 Otimização

Aqui a otimização será feita junta por junta, iniciando da junta mais externa e assim seguindo. Os ganhos das juntas não otimizadas ainda serão mantidos nulos.

2.1 Otimização junta 2

Aqui criei algumas funções para serem utilizadas na geração da população inicial. Como será visto posteriormente, dependendo da função geradora inicial temos diferentes resultados, isto para o cenário de 50 iterações do algoritmo genético (valor este utilizado para obter uma saída mais rápida).

```

In [193]: function gerador2(n)
    n = n/2
    kp = push!(zeros(n-1),rand()*rand([10.,100.,1000.,10000]))
    kv = push!(zeros(n-1),rand()*rand([10.,100.,1000.]))
    vcat(kp,kv)
end;

In [194]: function generateCusto(junta::Integer)
    out = function custo(gain::Vector{Float64})
        kp = SMatrix{2,2}(diagm([gain[1], gain[2]]))
        kv = SMatrix{2,2}(diagm([gain[3], gain[4]]))
        x, v, t, a, ta, j, tj, , t_tau = robot2dof(kp, kv, Ts, t0, tend, [r1, r2])

        sizeVector = length(x[1])

        erro_sum = 0.
        erro = -(x[junta]-xr[junta])
        erro_sum += sum(abs.(erro[floor(Integer,sizeVector*per):end]))

        jerk_sum = 0.
        jerk_sum += sum(abs.(j[junta]))

        torque_sum = 0.
        torque_sum += sum(abs.([junta]))

        erro_sum = erro_sum *
        jerk_sum = jerk_sum *
        torque_sum = torque_sum *
        #println(" $(erro_sum) / $(jerk_sum) / $(torque_sum)")
        out = erro_sum + jerk_sum + torque_sum
        out
    end
end;

In [195]: N = 4
    result, fitness, cnt = ga(generateCusto(2), N; initPopulation = gerador2, population

Progress:|| 0.0%

WARNING: Interrupted. Larger maxiters is needed.

Progress:|| 100.0%

Out[195]: ([0.00330958, 7385.74, 0.128808, 52.4041], 4.465087312978853, 15, 0.0, Dict{Symbol,A

In [196]: push!(kp_end, result[2])
    push!(kv_end, result[4])
    Markdown.parse("---|junta 2\n---|---\n**KP**|$(round(result[2],2))\n**KV**|$(round(r

```

Out[196]:

—	junta 2
KP	7385.74
KV	52.4

2.2 Otimização junta 1

```
In [197]: function gerador1(n)
    n = n/2
    kp = push!(zeros(n-2),rand()*rand([10.,100.,1000.,10000]))
    push!(kp,result[2])
    kv = push!(zeros(n-2),rand()*rand([10.,100.,1000.]))
    push!(kv,result[4])
    vcat(kp,kv)
end;

In [198]: N = 4
    result, fitness, cnt = ga(generateCusto(1), N; initPopulation = gerador1, population

Progress:|| 100.0%37m| 6.7%

Out[198]: ([3857.18, 7385.74, 1085.07, 52.4041], 105.9823753467286, 15, 0.0, Dict{Symbol,Any}())

In [199]: push!(kp_end, result[1])
    push!(kv_end, result[3])
    Markdown.parse("---|junta 1|n---|---\n**KP**|$(round(result[1],2))\n**KV**|$(round(r

Out[199]:
```

—	junta 1
KP	3857.18
KV	1085.07

2.3 Resultado da otimização

```
In [200]: kp = SMatrix{2,2}(diagm(flipdim(kp_end[1:2],1)))
    kv = SMatrix{2,2}(diagm(flipdim(kv_end[1:2],1)))
    x, v, t, a, ta, j, tj, , t_tau = robot2dof(kp, kv, Ts, t0, tend, [r1, r2])
    function plotx()
        p1 = plot(t,x[1], label = "PD ótimo - junta 1",xlabel ="tempo (s)", ylabel = "po
        p1= plot!([r1],seriestype=:hline, label = "referência");
        p2 = plot(t,x[2], label = "PD ótimo - junta 2",xlabel ="tempo (s)", ylabel = "po
        p2 = plot!([r2],seriestype=:hline, label = "referência");
        plot(p1,p2, title = "Posição")
    end

    function plotj()
        p1 = plot(tj,j[1], label = "PD ótimo - junta 1", xlabel ="tempo (s)", ylabel = "
```

```

p2 = plot(tj,j[2], label = "PD ótimo - junta 2",xlabel ="tempo (s)", ylabel = "J")
plot(p1,p2, title = "Jerk")
end;

function plotTau()
    p1 = plot(t_tau,[1], label = "PD ótimo - junta 1", xlabel ="tempo (s)", ylabel = "Torque")
    p2 = plot(t_tau,[2], label = "PD ótimo - junta 2",xlabel ="tempo (s)", ylabel = "Torque")
    plot(p1,p2, title = "Torque")
end;

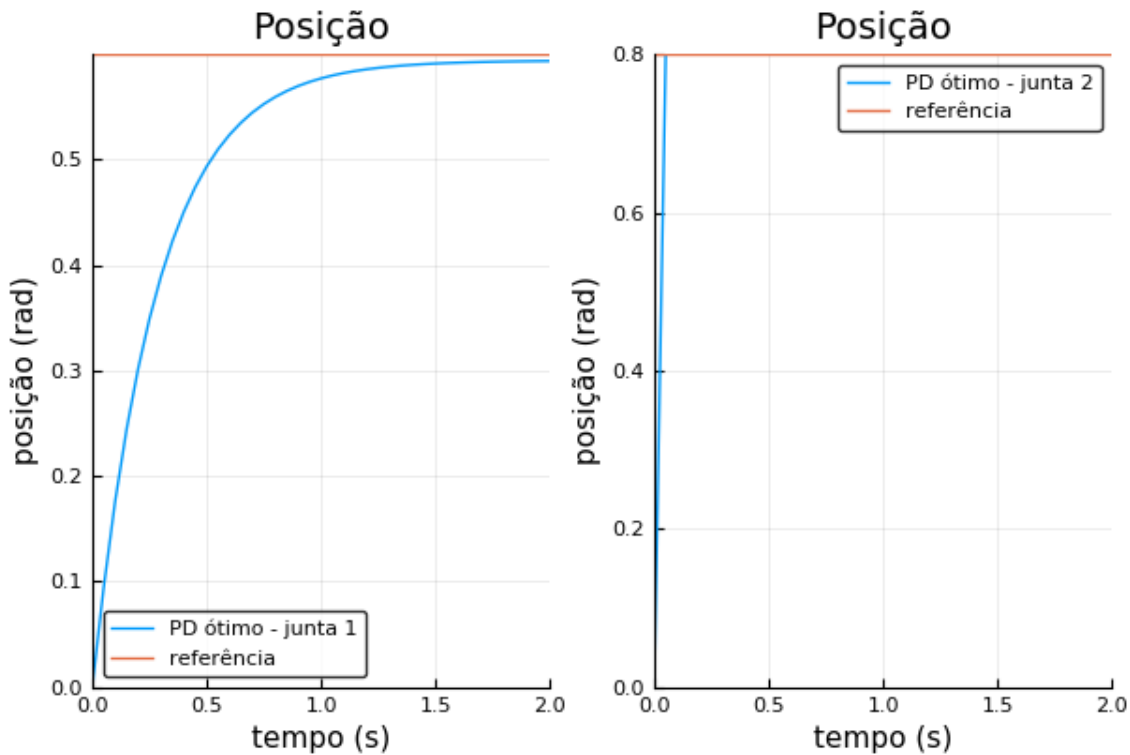
function plotv()
    p1 = plot(t,v[1], label = "PD ótimo - junta 1", xlabel ="tempo (s)", ylabel = "Velocidade")
    p2 = plot(t,v[2], label = "PD ótimo - junta 2",xlabel ="tempo (s)", ylabel = "Velocidade")
    plot(p1,p2, title = "Velocidade")
end;

function plota()
    p1 = plot(ta,a[1], label = "PD ótimo - junta 1", xlabel ="tempo (s)", ylabel = "Aceleração")
    p2 = plot(ta,a[2], label = "PD ótimo - junta 2",xlabel ="tempo (s)", ylabel = "Aceleração")
    plot(p1,p2, title = "Aceleração")
end;

```

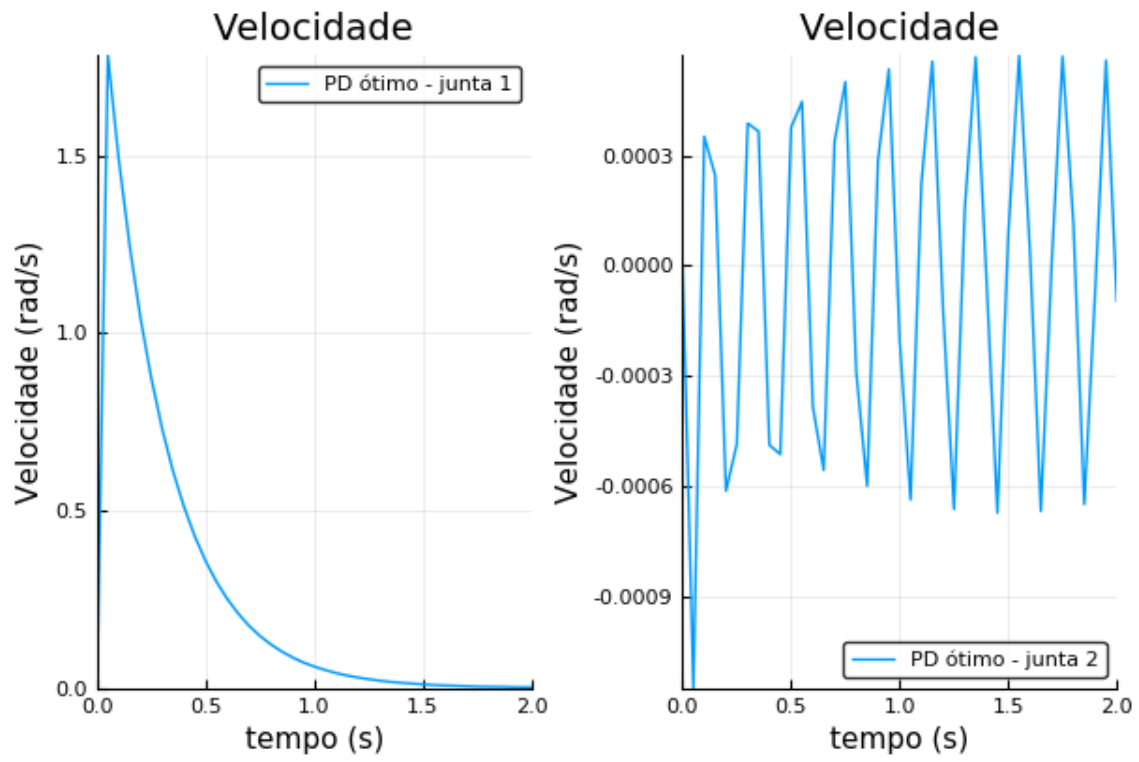
In [201]: plotx()

Out[201]:



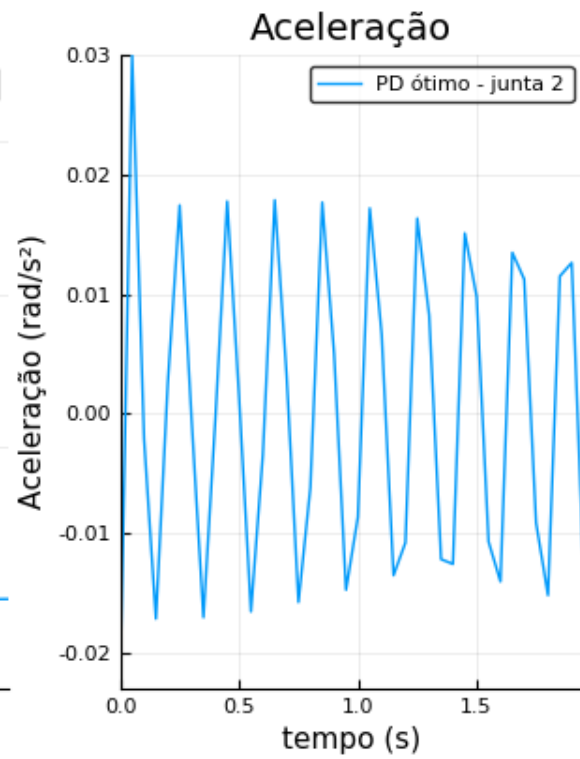
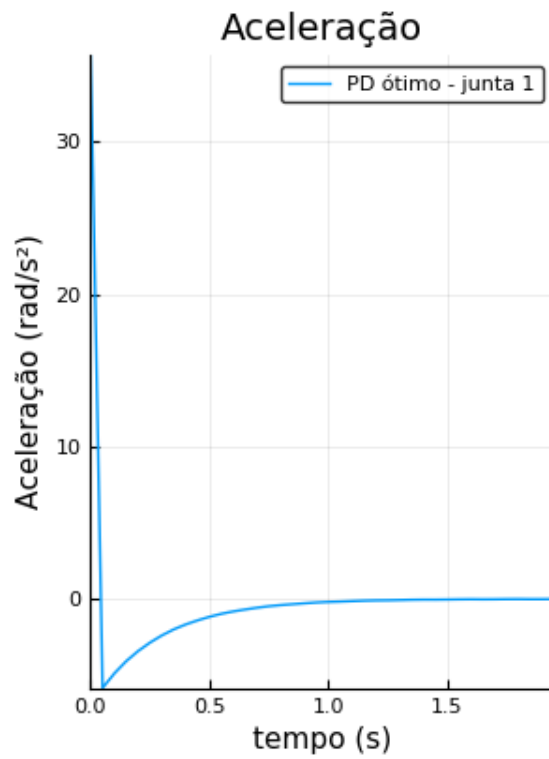
```
In [202]: plotv()
```

Out[202]:



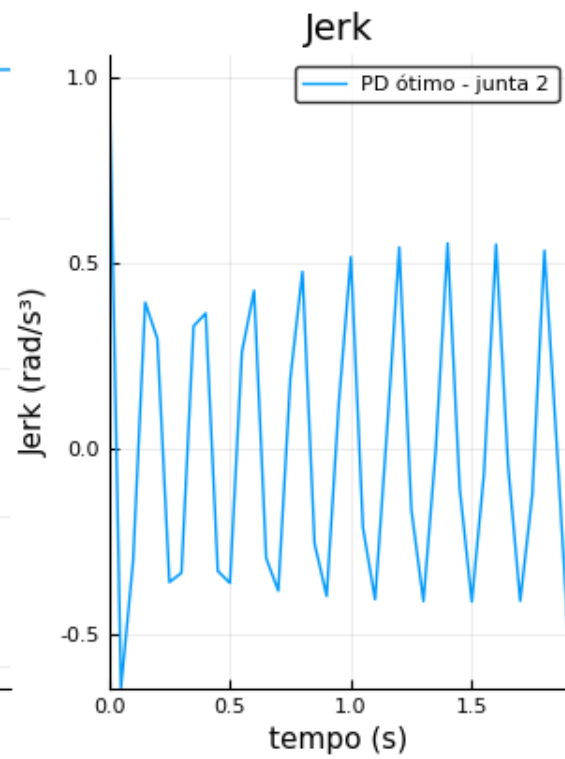
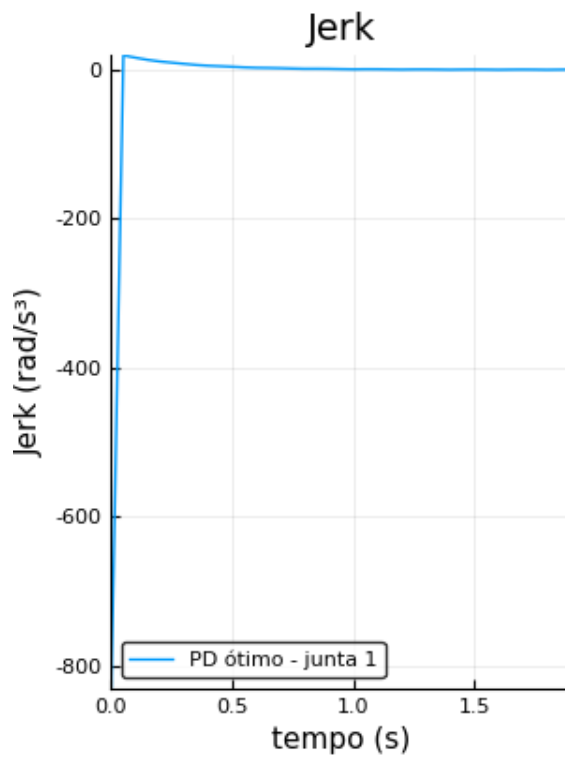
```
In [203]: plota()
```

Out[203]:



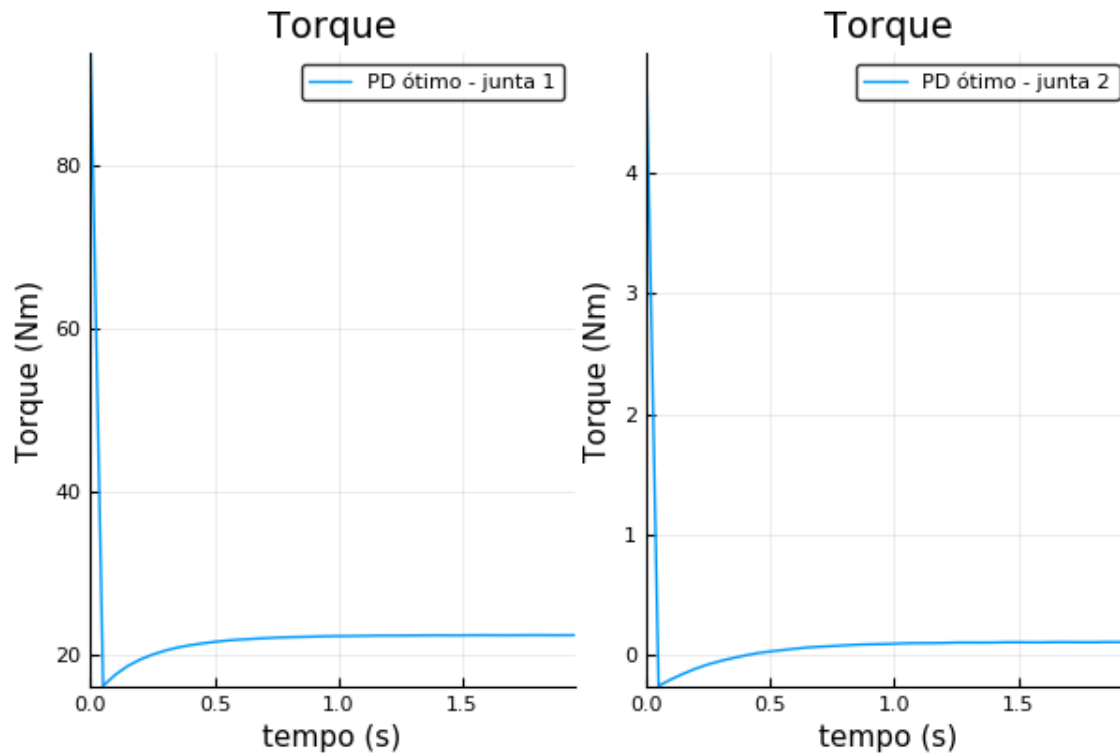
In [204]: `plotj()`

Out [204]:



```
In [205]: plotTau()
```

Out[205]:



A tabelas a seguir apresentam um resumo dos resultados para o PID otimizado.

```
In [206]: tabela(j, "Jerk")
```

Out[206]:

	— junta 1	junta 2
Jerk máximo	830.14	1.06
Jerk total	947.38	13.22

```
In [207]: tabela(, "Torque")
```

Out[207]:

	— junta 1	junta 2
Torque máximo	93.71	5.0
Torque total	940.11	8.61

3 PD clássico

3.1 Código

```
In [208]: kp_pid = SMatrix{2,2}(diagm([2800., 80.]))
kv_pid = SMatrix{2,2}(diagm([315., 15.]))
x_pid, v_pid, t_pid, a_pid, ta_pid, j_pid, tj_pid, _pid, t_tau_pid = robot2dof(kp_pid, kv_pid)
erro1 = -(x_pid[1] - r1)
erro2 = -(x_pid[2] - r2)
erro = [erro1, erro2]

function plotx_pid()
    p1 = plot(t_pid,x_pid[1], label = "PD - junta 1",xlabel ="tempo (s)", ylabel = "Posição (m)");
    p1= plot!([r1],seriestype=:hline, label = "referência");
    p2 = plot(t_pid,x_pid[2], label = "PD - junta 2",xlabel ="tempo (s)", ylabel = "Posição (m)");
    p2 = plot!([r2],seriestype=:hline, label = "referência");
    plot(p1,p2, title = "Posição")
end

function plotj_pid()
    p1 = plot(tj_pid,j_pid[1], label = "PD - junta 1", xlabel ="tempo (s)", ylabel = "Aceleração (m/s²)");
    p2 = plot(tj_pid,j_pid[2], label = "PD - junta 2",xlabel ="tempo (s)", ylabel = "Aceleração (m/s²)");
    plot(p1,p2, title = "Jerk")
end;

function plotTau_pid()
    p1 = plot(t_tau_pid,_pid[1], label = "PD - junta 1", xlabel ="tempo (s)", ylabel = "Torque (N)");
    p2 = plot(t_tau_pid,_pid[2], label = "PD - junta 2",xlabel ="tempo (s)", ylabel = "Torque (N)");
    plot(p1,p2, title = "Torque")
end;

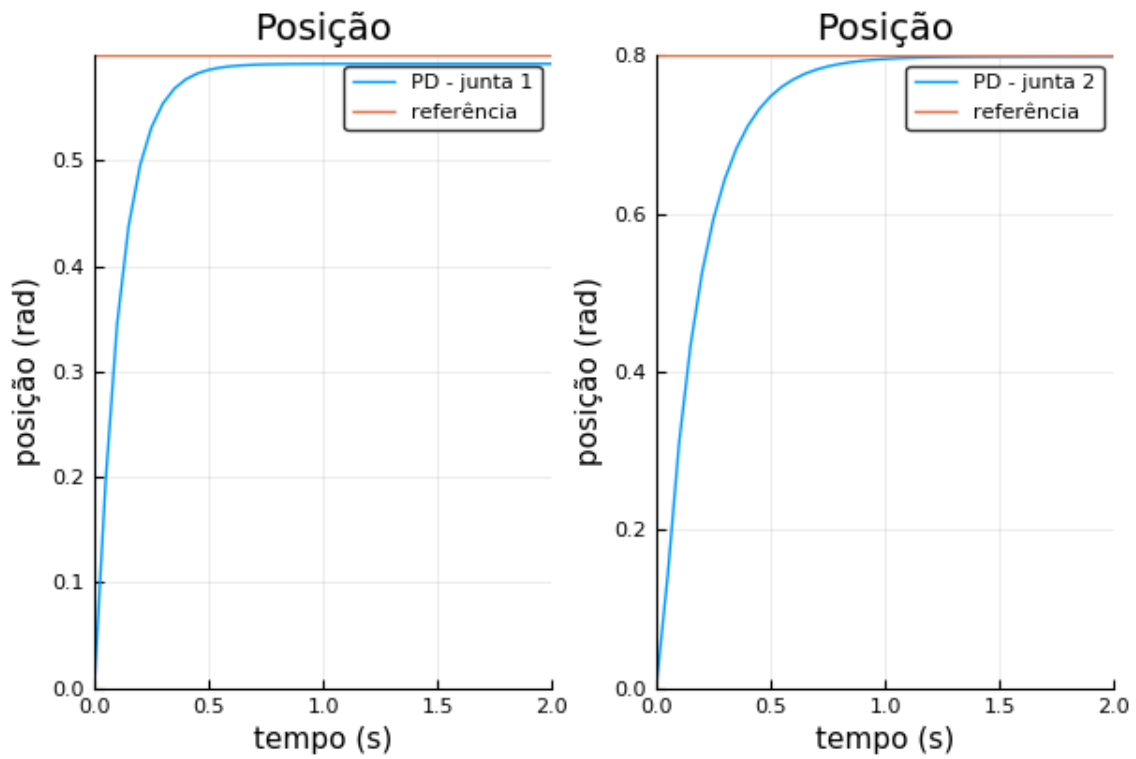
function plotv_pid()
    p1 = plot(t_pid,v_pid[1], label = "PD - junta 1", xlabel ="tempo (s)", ylabel = "Velocidade (m/s)");
    p2 = plot(t_pid,v_pid[2], label = "PD - junta 2",xlabel ="tempo (s)", ylabel = "Velocidade (m/s)");
    plot(p1,p2, title = "Velocidade")
end;

function plota_pid()
    p1 = plot(ta_pid,a_pid[1], label = "PD - junta 1", xlabel ="tempo (s)", ylabel = "Aceleração (m/s²)");
    p2 = plot(ta_pid,a_pid[2], label = "PD - junta 2",xlabel ="tempo (s)", ylabel = "Aceleração (m/s²)");
    plot(p1,p2, title = "Aceleração")
end;
```

3.2 Resultados PD

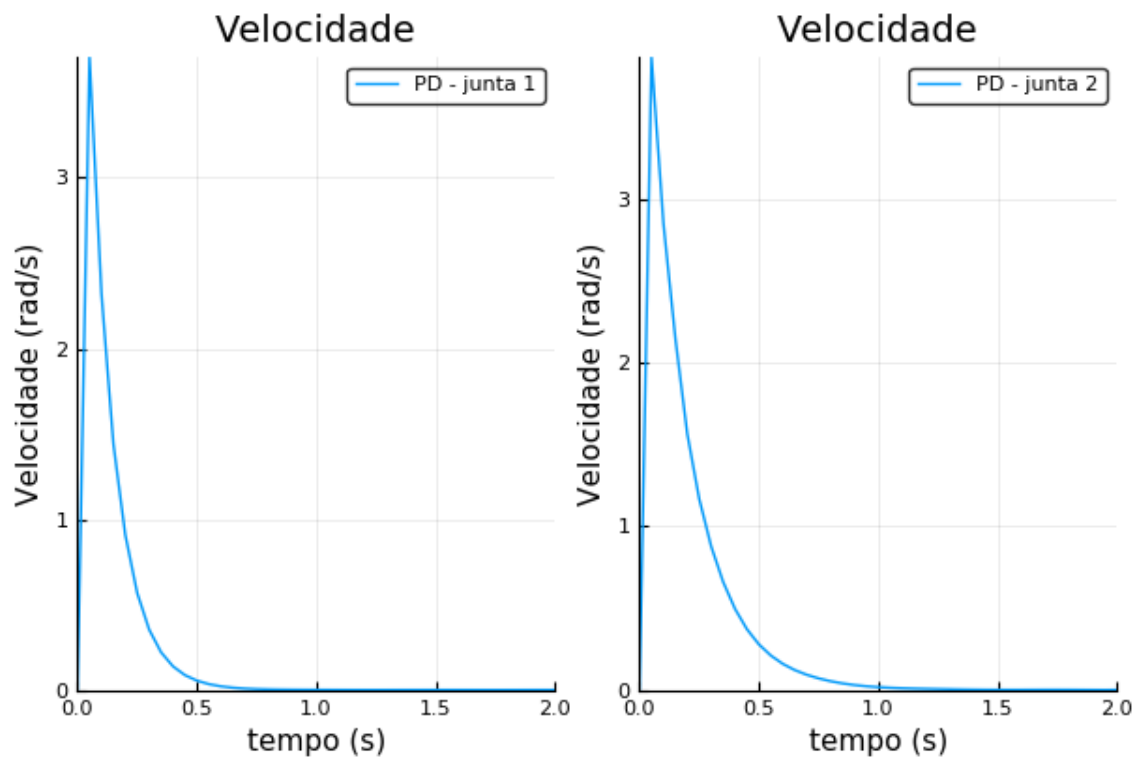
```
In [209]: plotx_pid()
```

```
Out[209]:
```



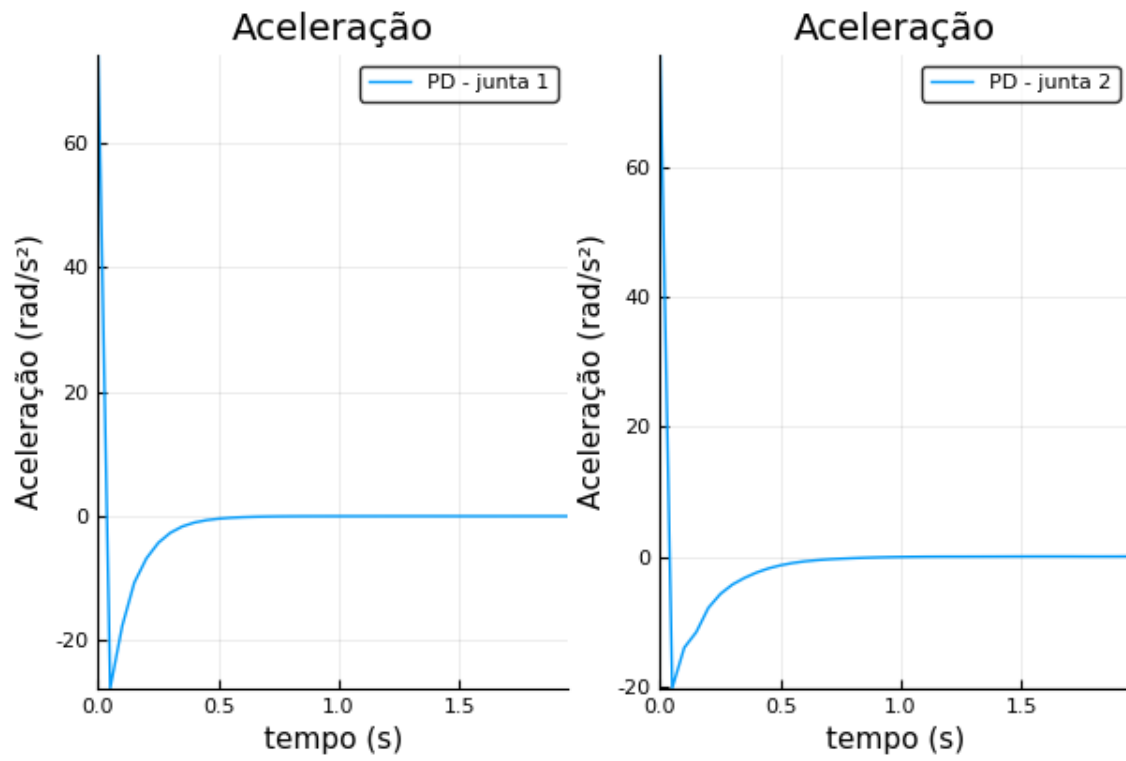
In [210]: `plotv_pid()`

Out [210]:



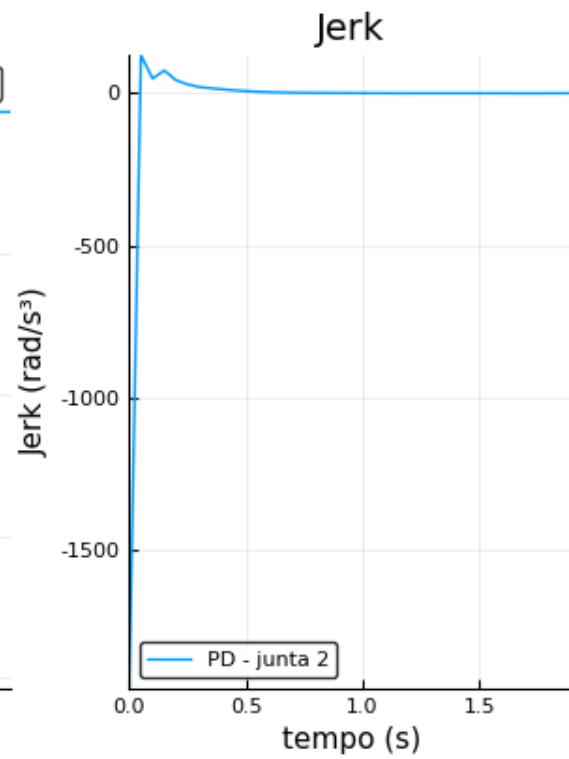
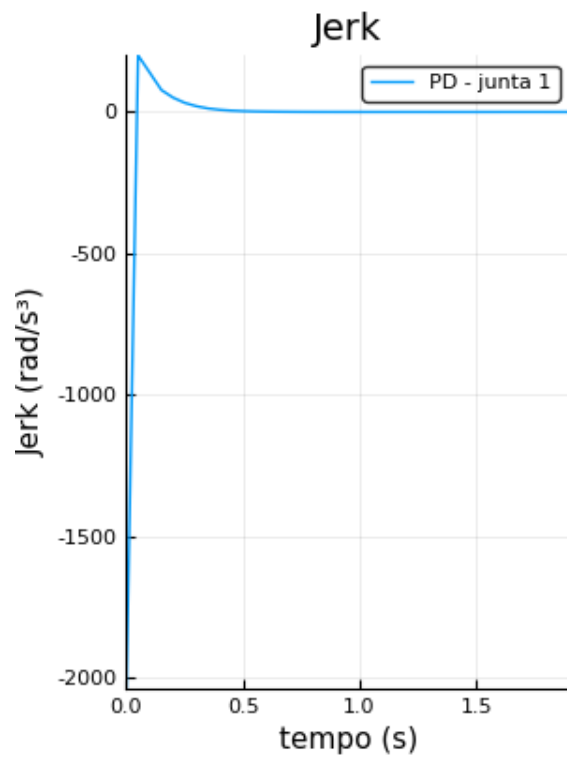
```
In [211]: plota_pid()
```

Out [211]:



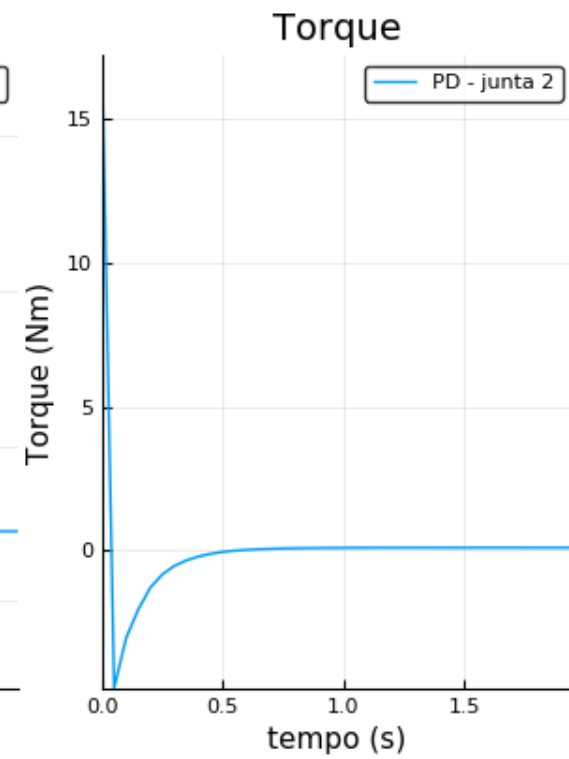
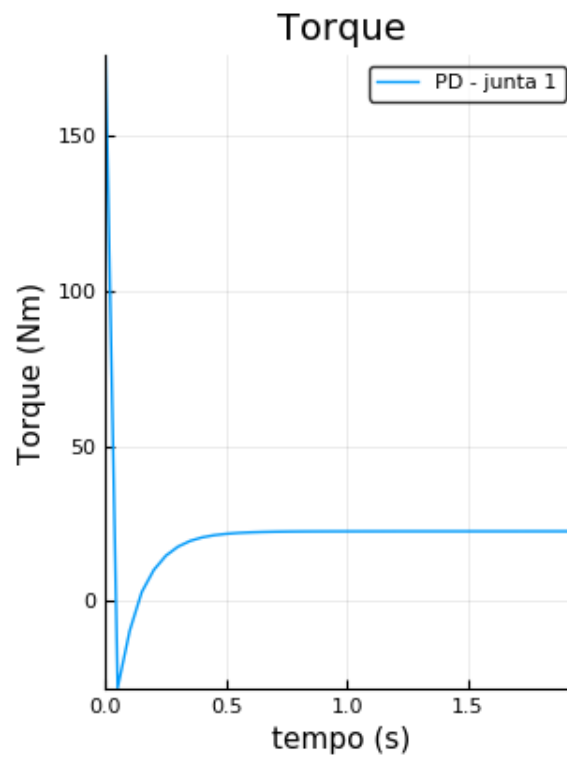
```
In [212]: plotj_pd()
```

Out [212]:



In [213]: `plotTau_pd()`

Out [213]:



A tabelas a seguir apresentam um resumo dos resultados para o PID clássico.

```
In [214]: tabela(j_pid,"Jerk (PD)")
```

Out[214]:

	— junta 1	junta 2
Jerk (PD) máximo	2034.53	1954.23
Jerk (PD) total	2587.32	2361.25

```
In [215]: tabela(_pid,"Torque (PD)")
```

Out[215]:

	— junta 1	junta 2
Torque (PD) máximo	175.93	17.22
Torque (PD) total	993.2	33.11

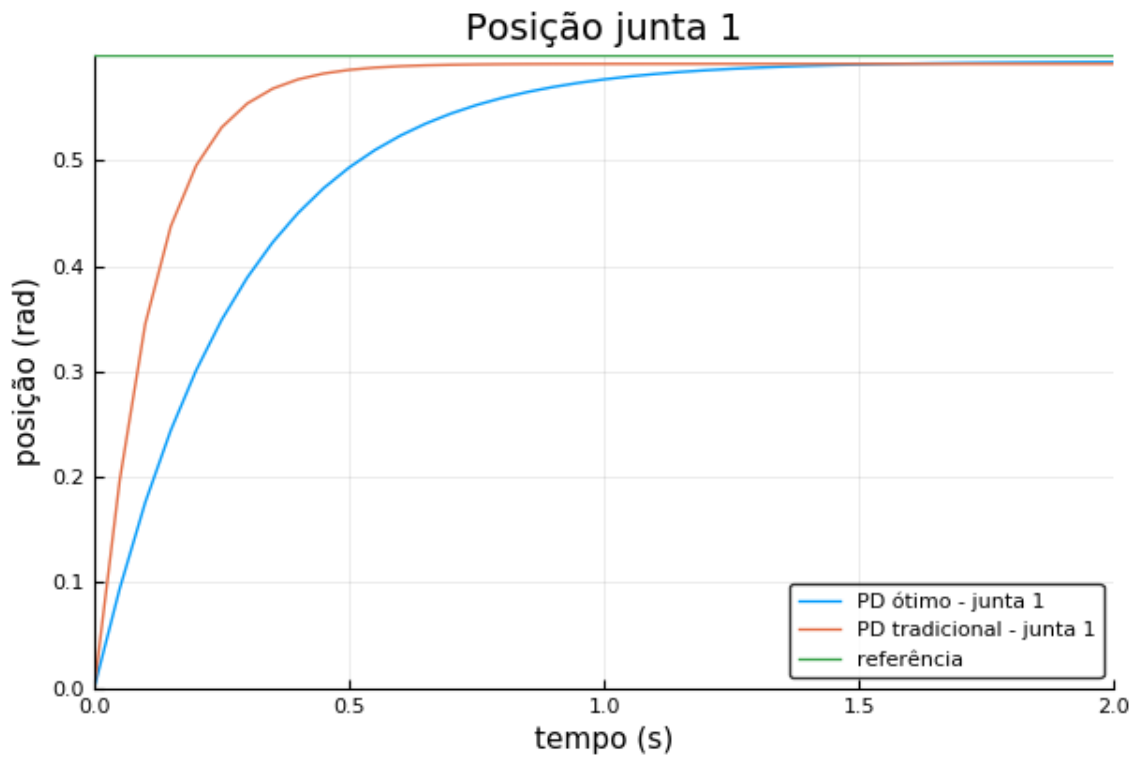
4 Gráficos comparativos

4.1 Posição

4.1.1 Posição junta 1

```
In [216]: plot(t,x[1],label = "PD ótimo - junta 1")
          plot!(t,x_pid[1],label = "PD tradicional - junta 1")
          plot!([r1],seriestype=:hline, label = "referência", title = "Posição junta 1", xlabel = "t (s)", ylabel = "x (m)")
```

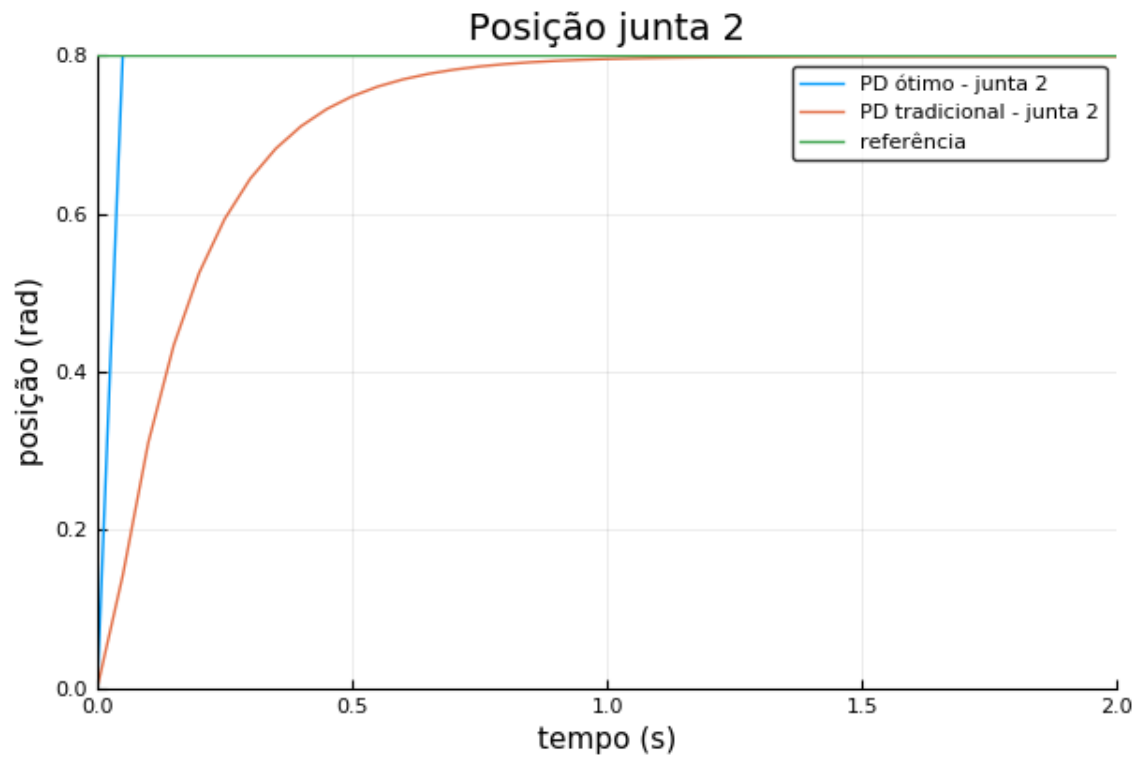
Out[216]:



4.1.2 Posição junta 2

```
In [217]: plot(t,x[2],label = "PD ótimo - junta 2")  
          plot!(t,x_pid[2],label = "PD tradicional - junta 2")  
          plot!([r2],seriestype=:hline, label = "referência", title = "Posição junta 2", xlabel = "tempo (s)")
```

Out[217]:

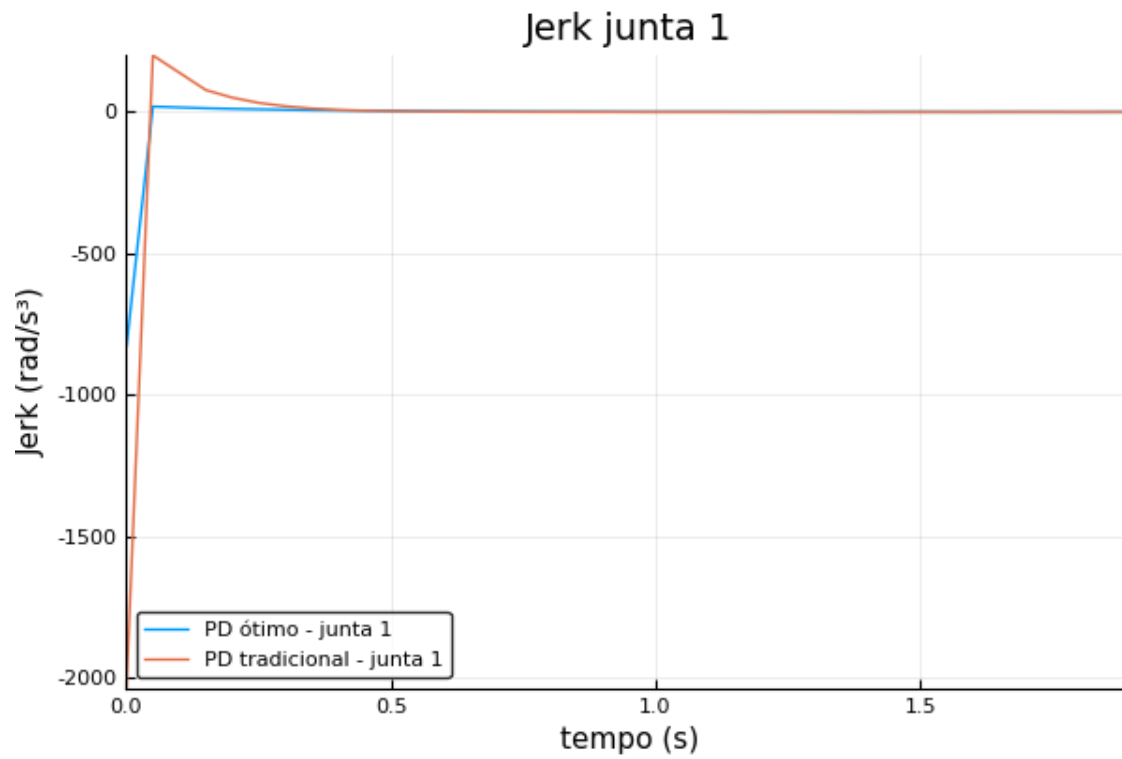


4.2 Jerk

4.2.1 Jerk junta 1

```
In [218]: plot(tj,j[1],label = "PD ótimo - junta 1")  
          plot!(tj,j_pid[1],label = "PD tradicional - junta 1", title = "Jerk junta 1", xlabel
```

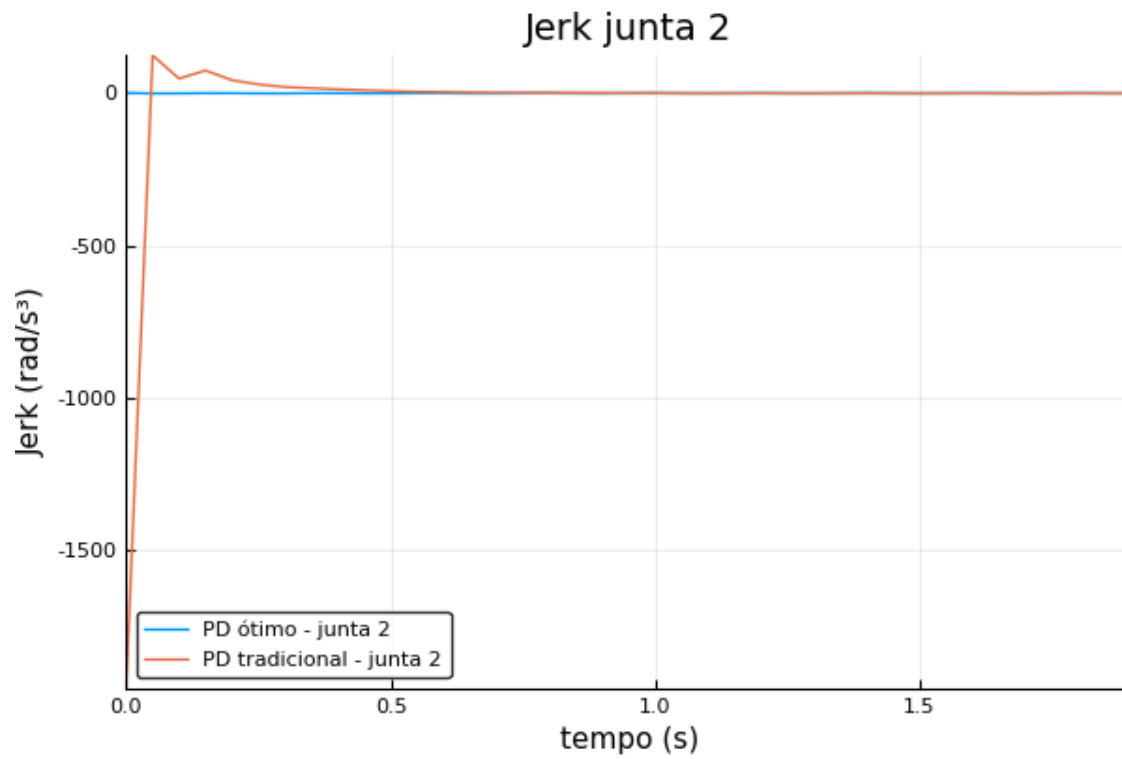
Out[218]:



4.2.2 Jerk junta 2

```
In [219]: plot(tj,j[2],label = "PD ótimo - junta 2")  
          plot!(tj,j_pid[2],label = "PD tradicional - junta 2", title = "Jerk junta 2", xlabel = "tempo (s)", ylabel = "Jerk (rad/s³)")
```

Out[219]:

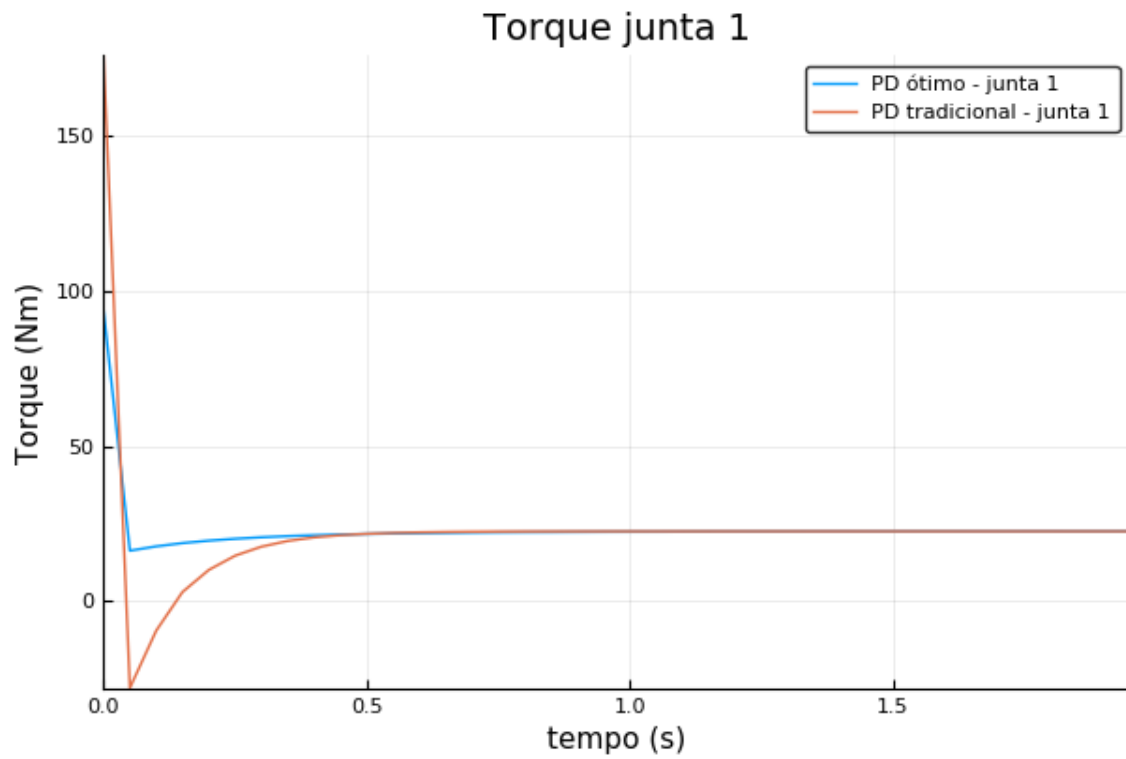


4.3 Torque

4.3.1 Torque junta 1

```
In [220]: plot(t_tau_pid,[1],label = "PD ótimo - junta 1")  
          plot!(t_tau_pid,_pid[1],label = "PD tradicional - junta 1", title = "Torque junta 1")
```

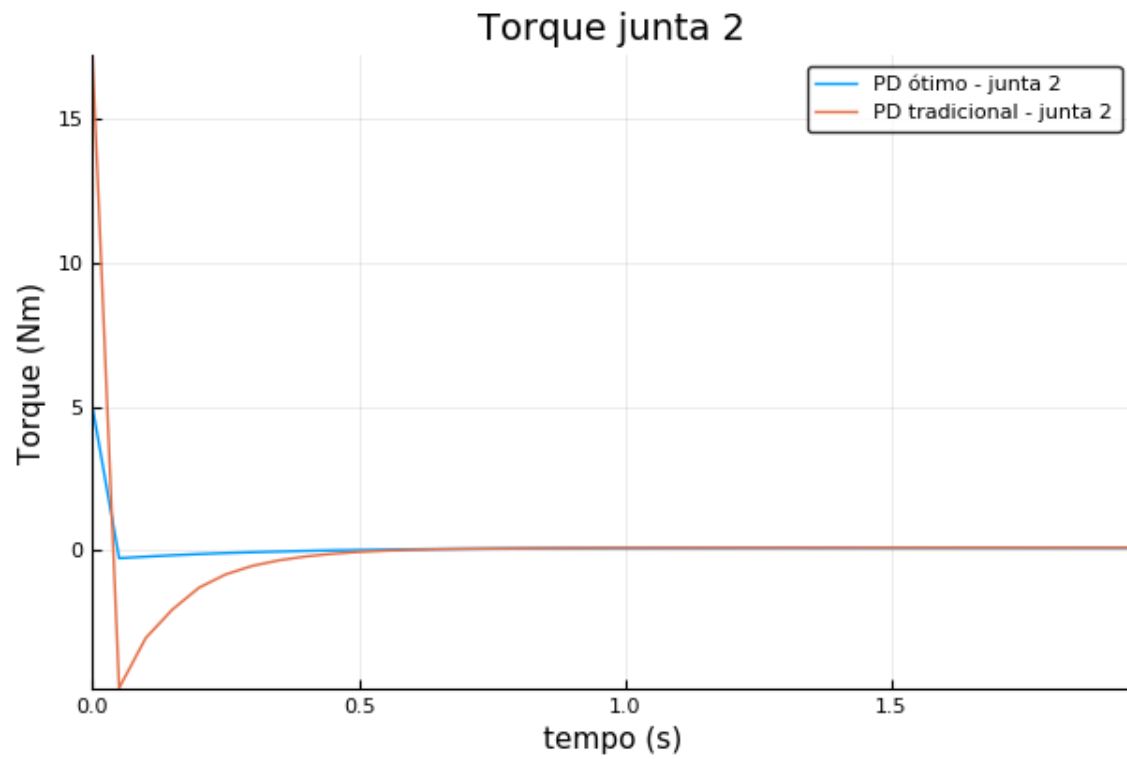
Out[220]:



4.3.2 Torque junta 2

```
In [221]: plot(t_tau_pid,[2],label = "PD ótimo - junta 2")  
          plot!(t_tau_pid,_pid[2],label = "PD tradicional - junta 2", title = "Torque junta 2")
```

Out[221]:

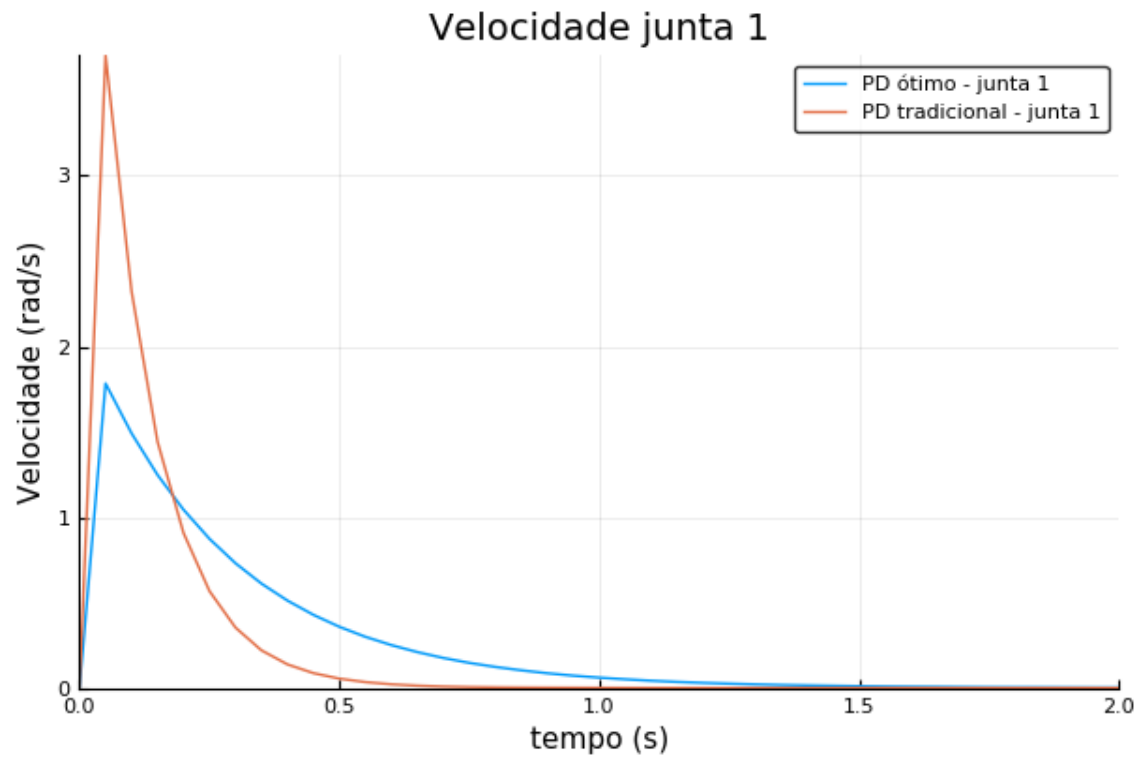


4.4 Velocidade

4.4.1 Velocidade junta 1

```
In [222]: plot(t,v[1],label = "PD ótimo - junta 1")
          plot!(t,v_pid[1],label = "PD tradicional - junta 1", title = "Velocidade junta 1", x=
```

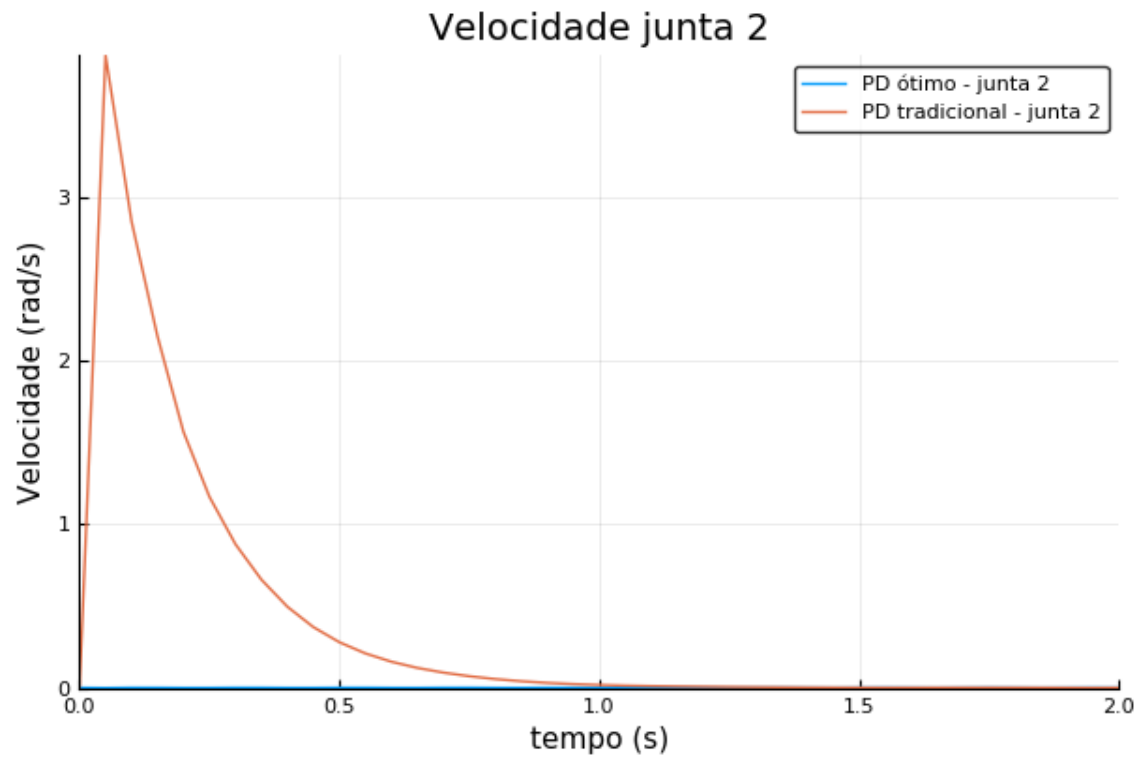
Out[222]:



4.4.2 Velocidade junta 2

```
In [223]: plot(t,v[2],label = "PD ótimo - junta 2")  
          plot!(t,v_pid[2],label = "PD tradicional - junta 2", title = "Velocidade junta 2", x
```

Out[223]:

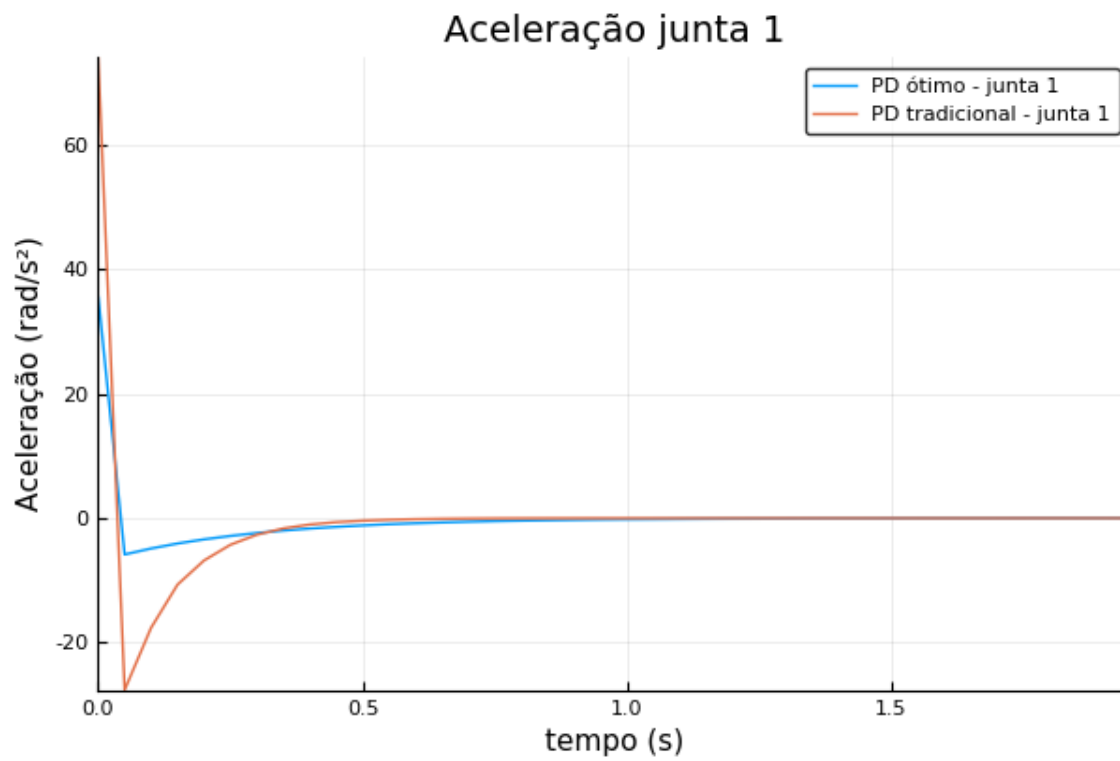


4.5 Aceleração

4.5.1 Aceleração junta 1

```
In [224]: plot(ta,a[1],label = "PD ótimo - junta 1")  
          plot!(ta,a_pid[1],label = "PD tradicional - junta 1", title = "Aceleração junta 1", :)
```

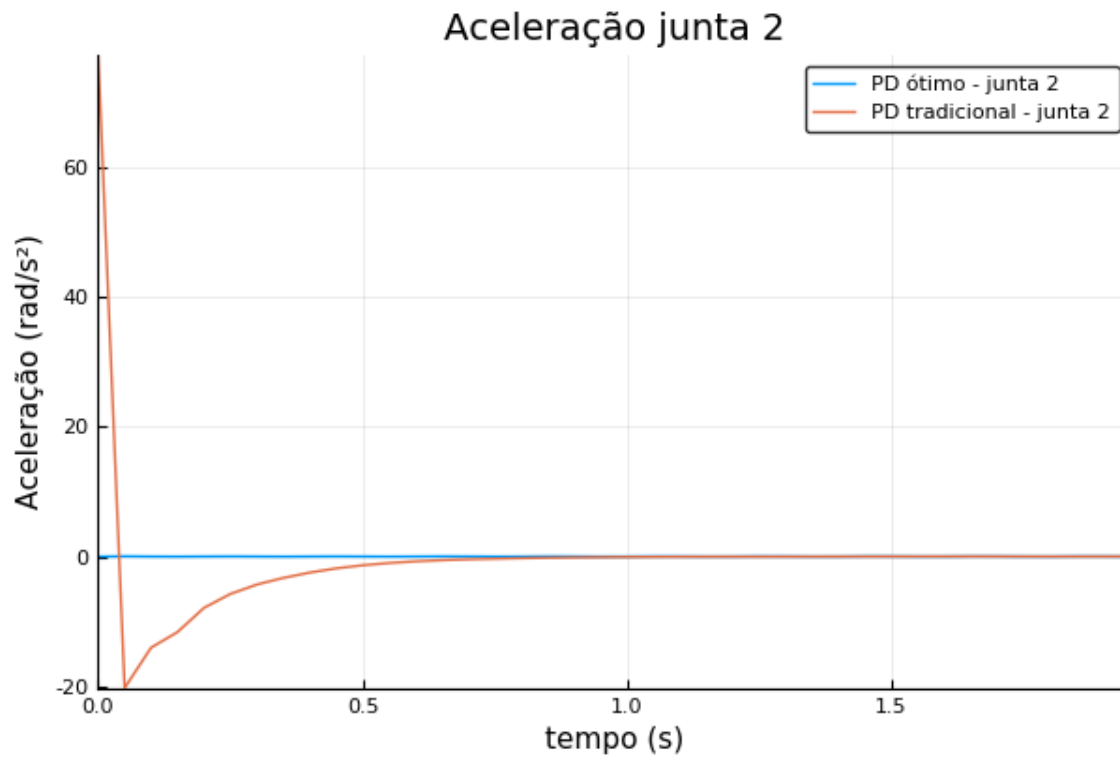
Out[224]:



4.5.2 Aceleração junta 2

```
In [225]: plot(ta,a[2],label = "PD ótimo - junta 2")  
          plot!(ta,a_pid[2],label = "PD tradicional - junta 2", title = "Aceleração junta 2",
```

Out[225]:



5 Discussão

5.1 Sobre o torque

In [226]: `tabela(, "Torque (ótimo)")`

Out[226]:

	— junta 1	junta 2
Torque (ótimo) máximo	93.71	5.0
Torque (ótimo) total	940.11	8.61

In [227]: `tabela(_pid, "Torque (clássico)")`

Out[227]:

	— junta 1	junta 2
Torque (clássico) máximo	175.93	17.22
Torque (clássico) total	993.2	33.11