pelicano_serial

April 29, 2018

1 Parâmetros

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

```
In [1]: include("../comum.jl")
        include("../modelos.jl")
        using Evolutionary, Plots;
        pyplot();
```

```
INFO: Recompiling stale cache file /home/phelipe/.julia/lib/v0.6/QuadGK.ji for module QuadGK.I
```

```
In [2]: 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     #inicio da leitura do vetor a parti de per do comprimento total
        kp_end = AbstractFloat[]
        kv_end = AbstractFloat[];
```

2 Otimização

Aqui a otmização 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 geredora inicial temos diferentes resultados, isto para o cenário de 50 iterações do algoritmo genético (valor este utilizado para obter uma saída mais rápida).

In [3]:
```
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 [4]:
```
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 [5]:
```
N = 4
 result, fitness, cnt = ga(generateCusto(2), N; initPopulation = gerador2, populationSi
```

```
 Progress:|| 100.0%
```

Out[5]:
```
([-0.0542029, 7933.52, 0.319774, 53.6495], 4.280257368935602, 15, 0.0822803255100455, I
```

In [6]:
```
push!(kp_end, result[2])
push!(kv_end, result[4])
Markdown.parse("---|junta 2\n---|---\n**KP**|$(round(result[2],2))\n**KV**|$(round(resu
```

Out[6]:

—	junta 2
KP	7933.52
KV	53.65

2.2 Otimização junta 1

```
In [7]: 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 [8]: N = 4
         result, fitness, cnt = ga(generateCusto(1), N; initPopulation = gerador1, populationSi

 Progress:|| 100.0%37m| 6.7%

Out[8]: ([3083.5, 7933.52, 794.635, 53.6462], 106.57655134800977, 15, 0.0, Dict{Symbol,Any}())

In [9]: push!(kp_end, result[1])
        push!(kv_end, result[3])
        Markdown.parse("---|junta 1\n---|---\n**KP**|$(round(result[1],2))\n**KV**|$(round(resu
```

Out[9]:

—	junta 1
KP	3083.5
KV	794.64

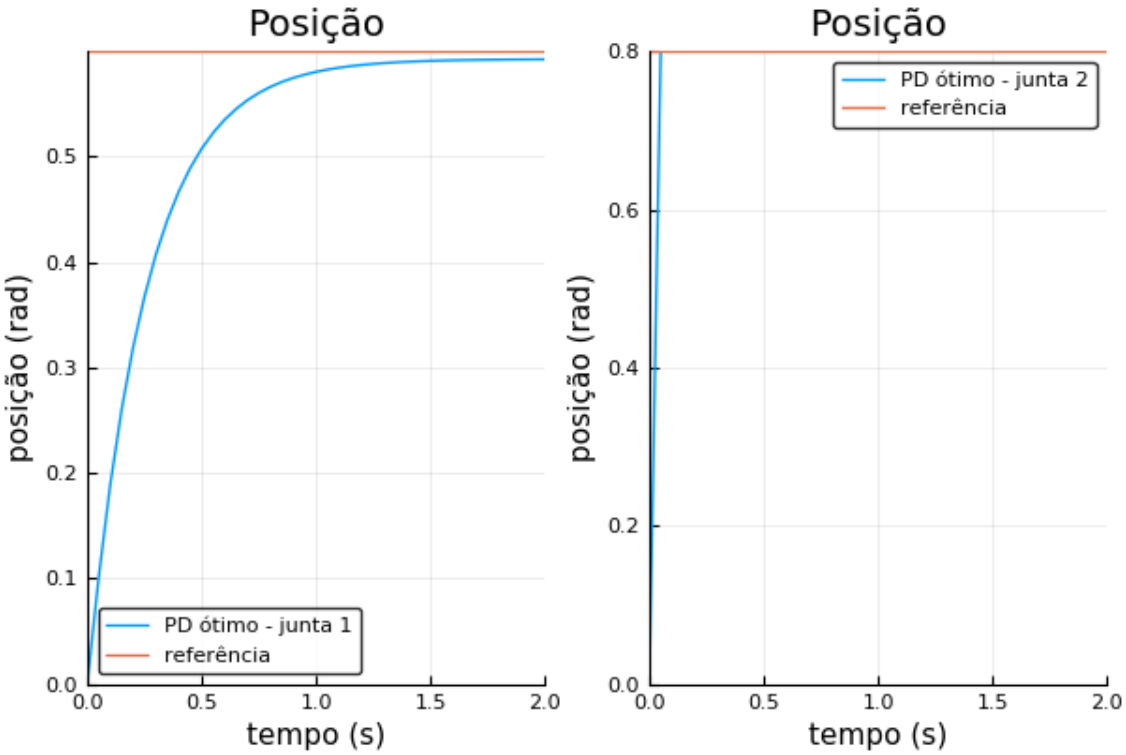
2.3 Resultado da otimização

```
In [10]: 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 = "posi
             p1= plot!([r1],seriestype= :hline, label = "referência");
             p2 = plot(t,x[2], label = "PD ótimo - junta 2",xlabel ="tempo (s)", ylabel = "posi
             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 = "Ac
```

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

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

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

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

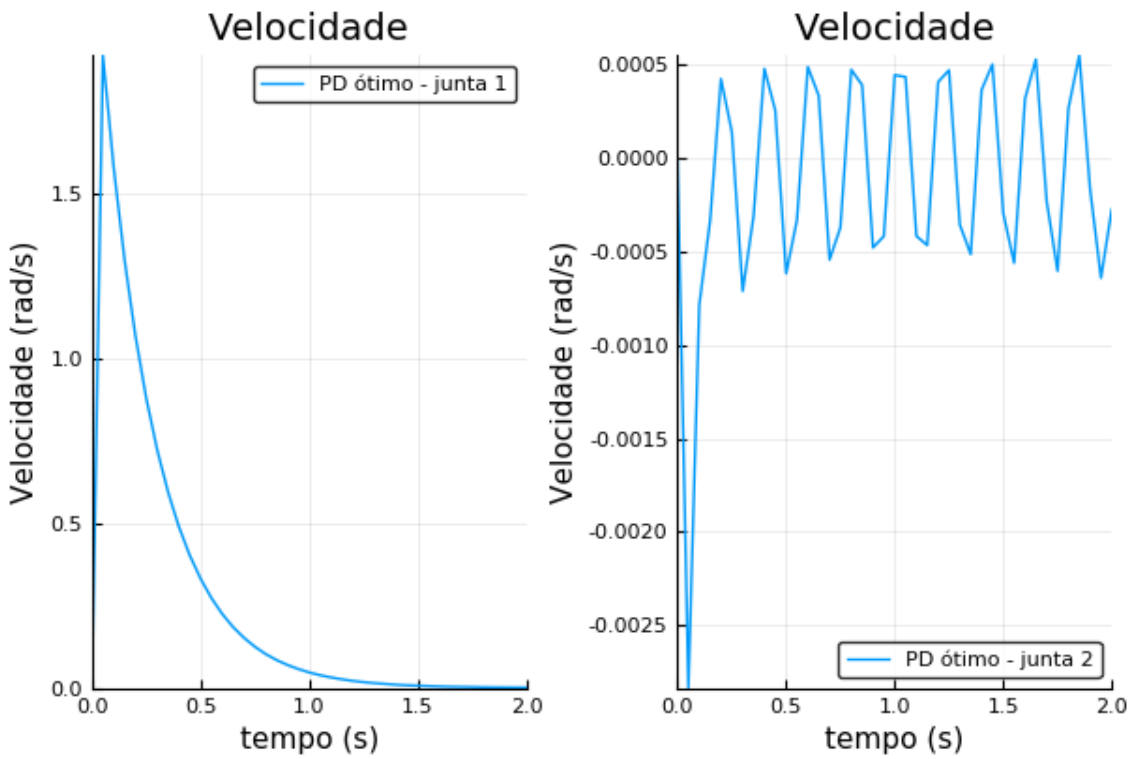
In [11]:
```
plotx()
```

Out[11]:



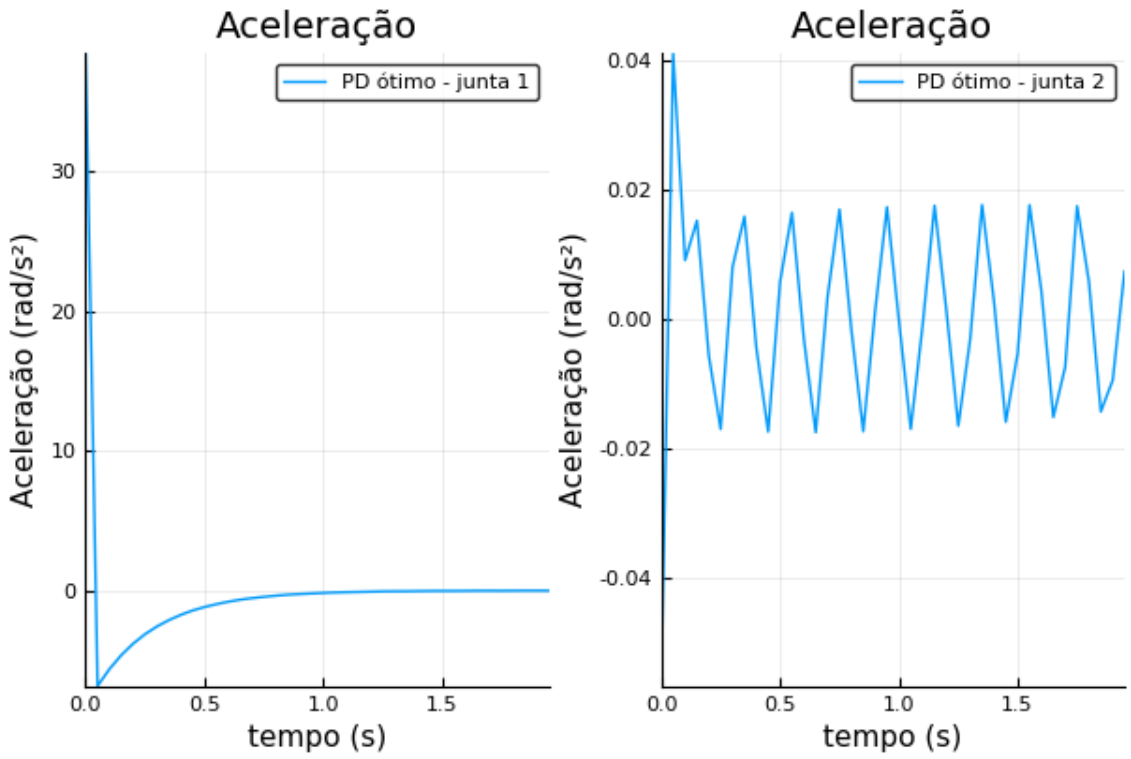
In [12]: plotv()

Out[12]:



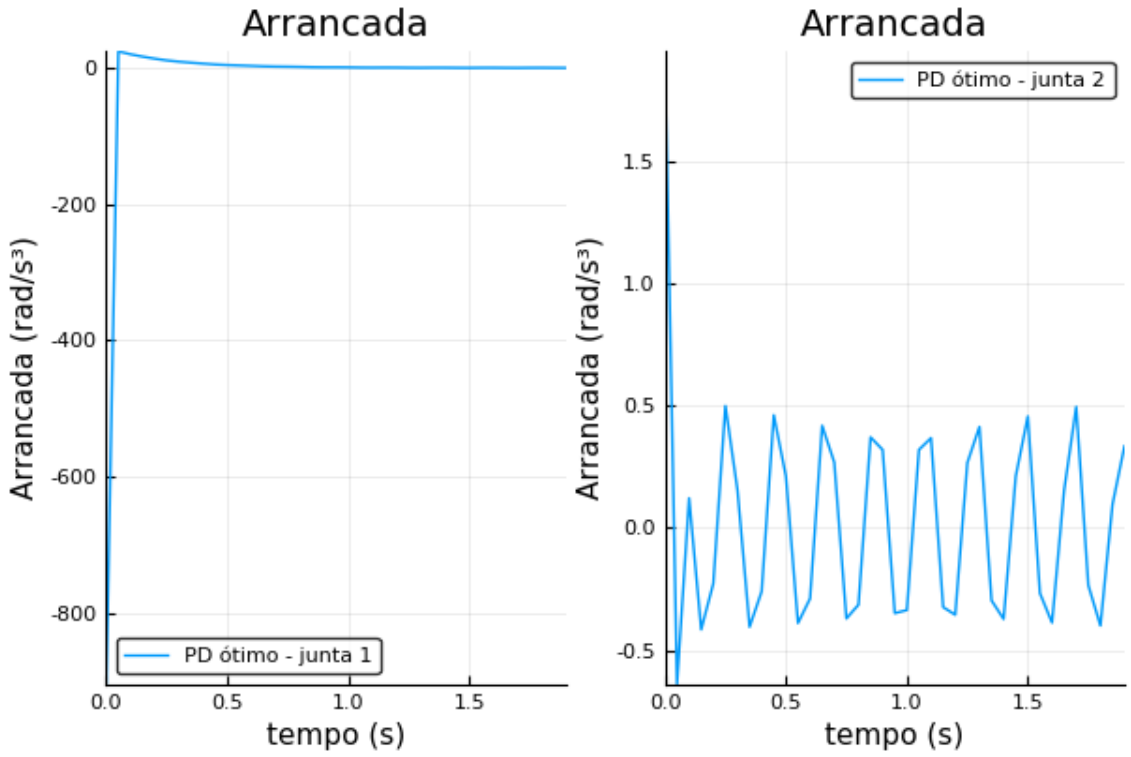
In [13]: plota()

Out[13]:



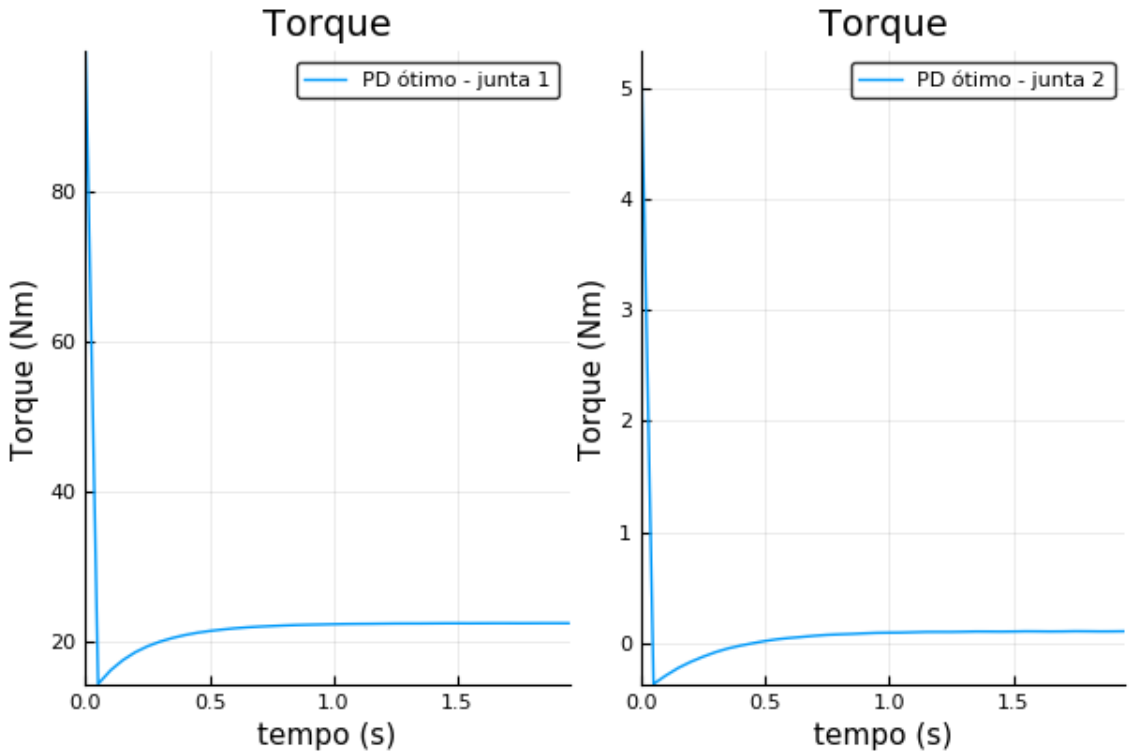
In [14]: plotj()

Out[14]:



In [15]: plotTau()

Out[15]:



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

In [16]: tabela(j, "Jerk")

Out[16]:

—	junta 1	junta 2
Jerk máximo	904.1	1.95
Jerk mínimo	0.0	0.1
Jerk total	1040.6	14.55

In [17]: tabela(, "Torque")

Out[17]:

—	junta 1	junta 2
Torque máximo	98.75	5.33

—	junta 1	junta 2
Torque mínimo	14.41	0.0
Torque total	937.44	9.37

3 PD clássico

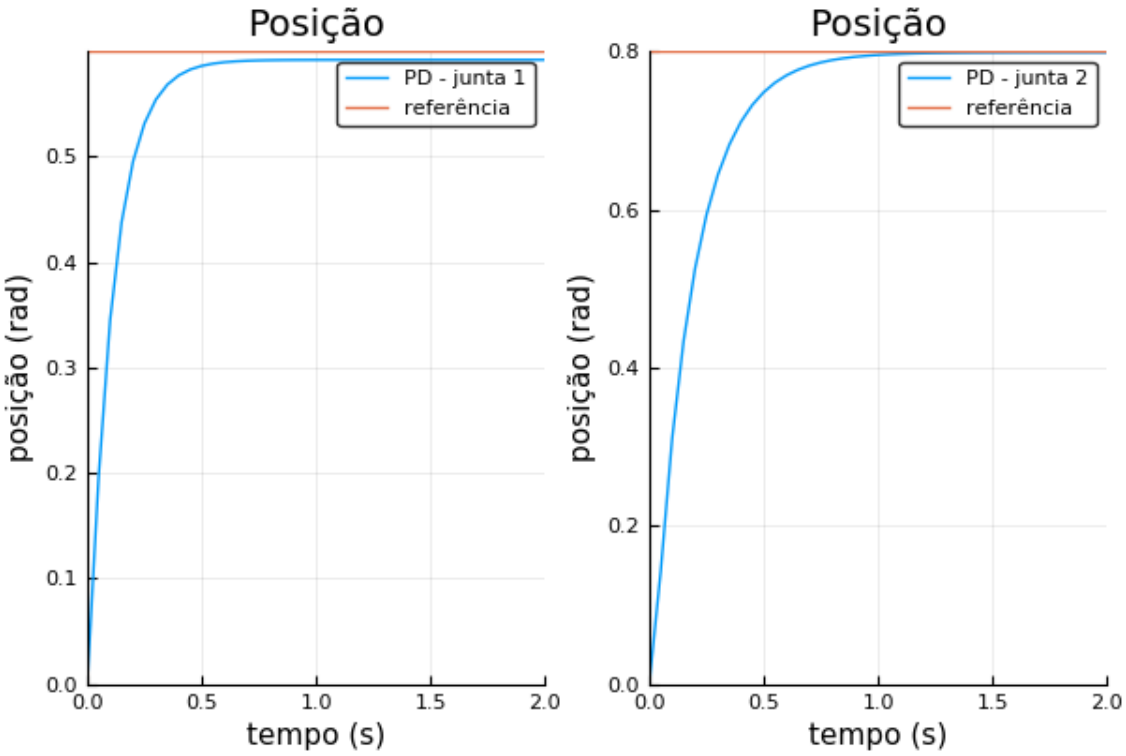
3.1 Código

```
In [18]: 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
         erro1 = -(x_pid[1] - r1)
         erro2 = -(x_pid[2] - r2)
         erro = [erro1, erro2]

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

         function plotj_pd()
             p1 = plot(tj_pid,j_pid[1], label = "PD - junta 1", xlabel ="tempo (s)", ylabel = "
             p2 = plot(tj_pid,j_pid[2], label = "PD - junta 2",xlabel ="tempo (s)", ylabel = "A
             plot(p1,p2, title = "Arrancada")
         end;

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

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

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

3.2 Resultados PD

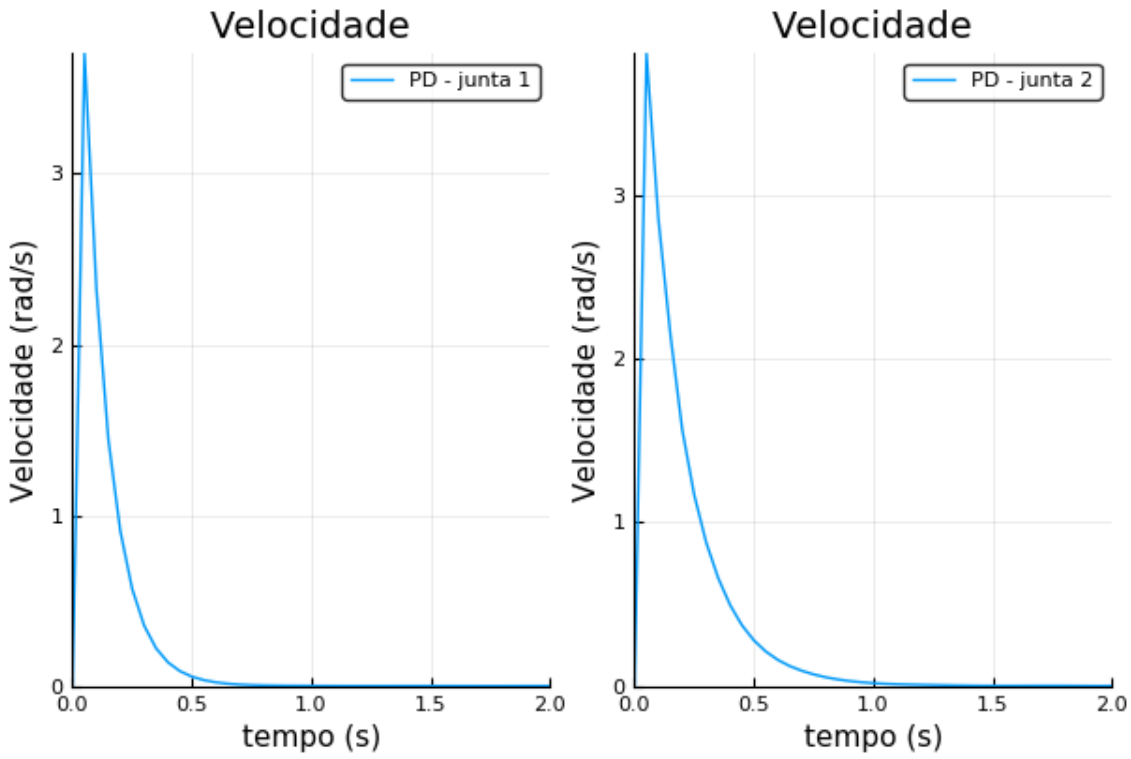
```
In [19]: plotx_pd()
```

Out[19]:



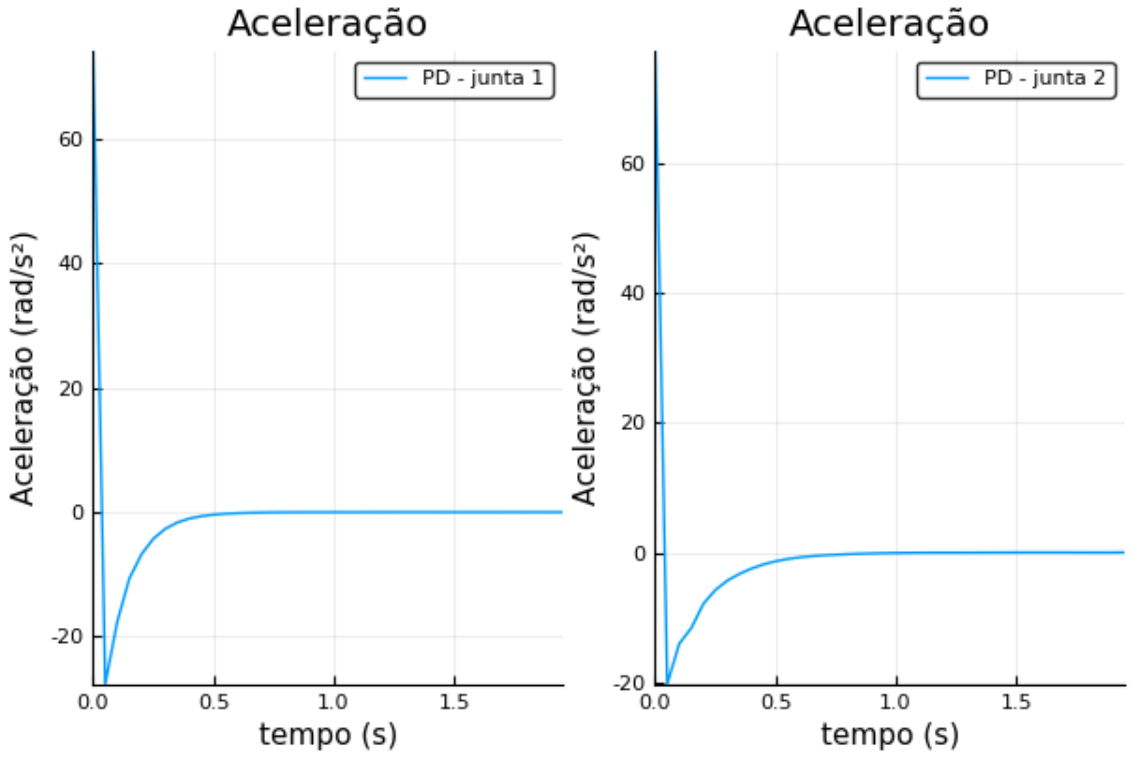

```
In [20]: plotv_pid()
```

Out[20]:



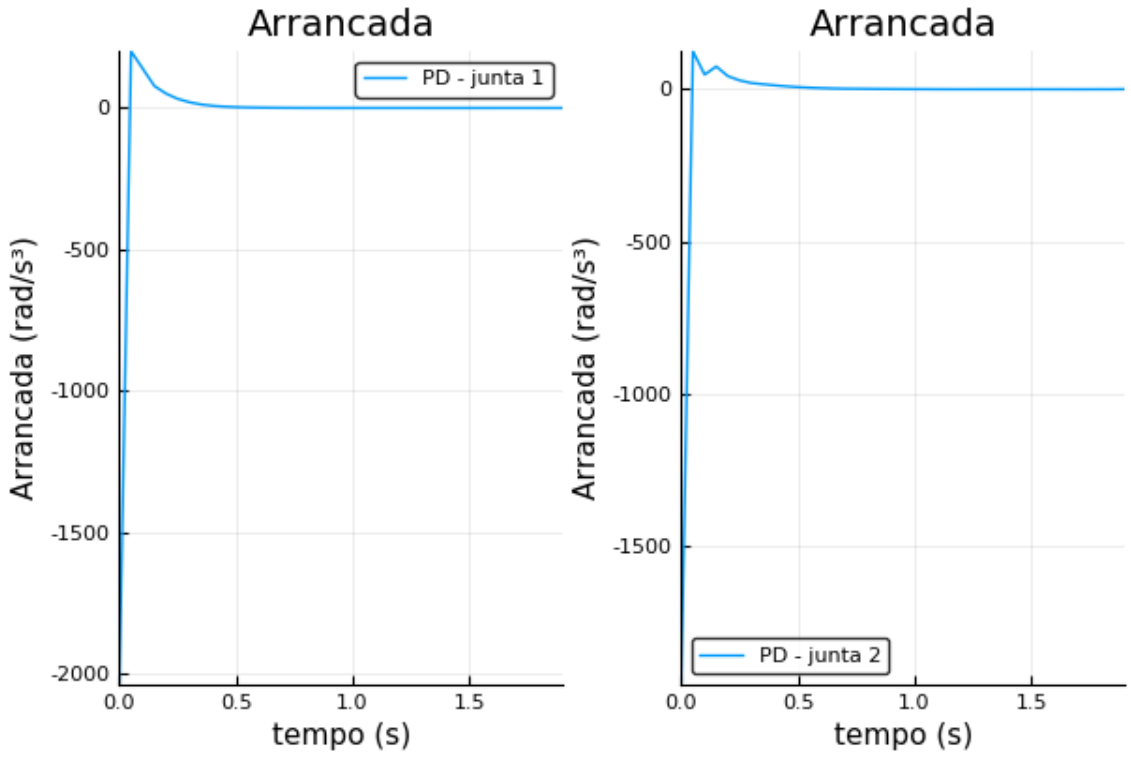
In [21]: plota_pid()

Out[21]:



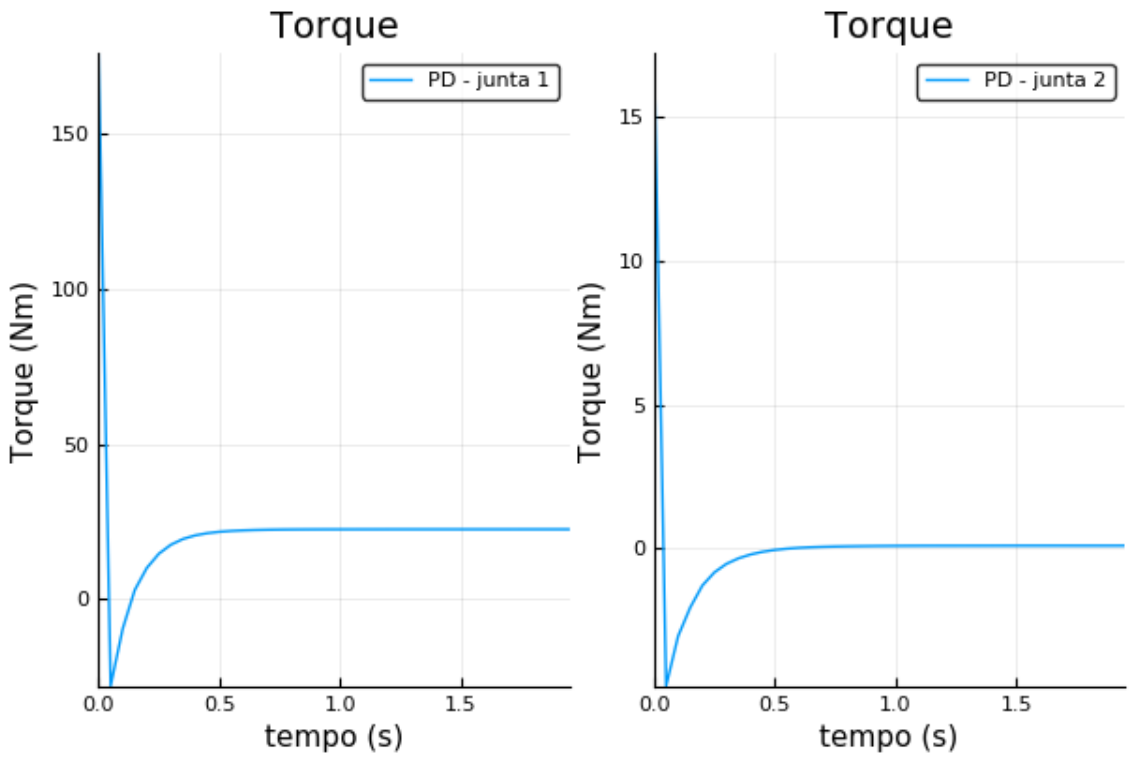
In [22]: plotj_pd()

Out[22]:



In [23]: plotTau_pd()

Out[23]:



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

```
In [24]: tabela(j_pid,"Jerk (PD)")
```

Out[24]:

—	junta 1	junta 2
Jerk (PD) máximo	2034.53	1954.23
Jerk (PD) mínimo	0.0	0.01
Jerk (PD) total	2587.32	2361.25

```
In [25]: tabela(_pid,"Torque (PD)")
```

Out[25]:

—	junta 1	junta 2
Torque (PD) máximo	175.93	17.22
Torque (PD) mínimo	2.96	0.0
Torque (PD) total	993.2	33.11

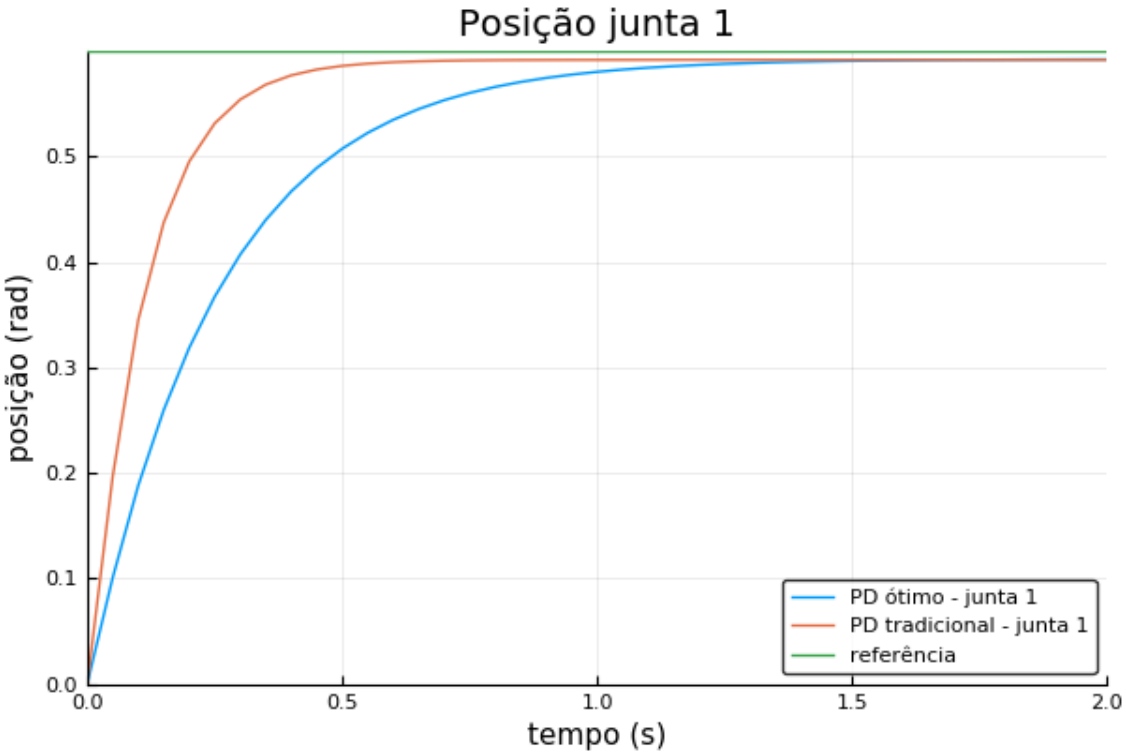
4 Gráficos comparativos

4.1 Posição

4.1.1 Posição junta 1

```
In [26]: 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
```

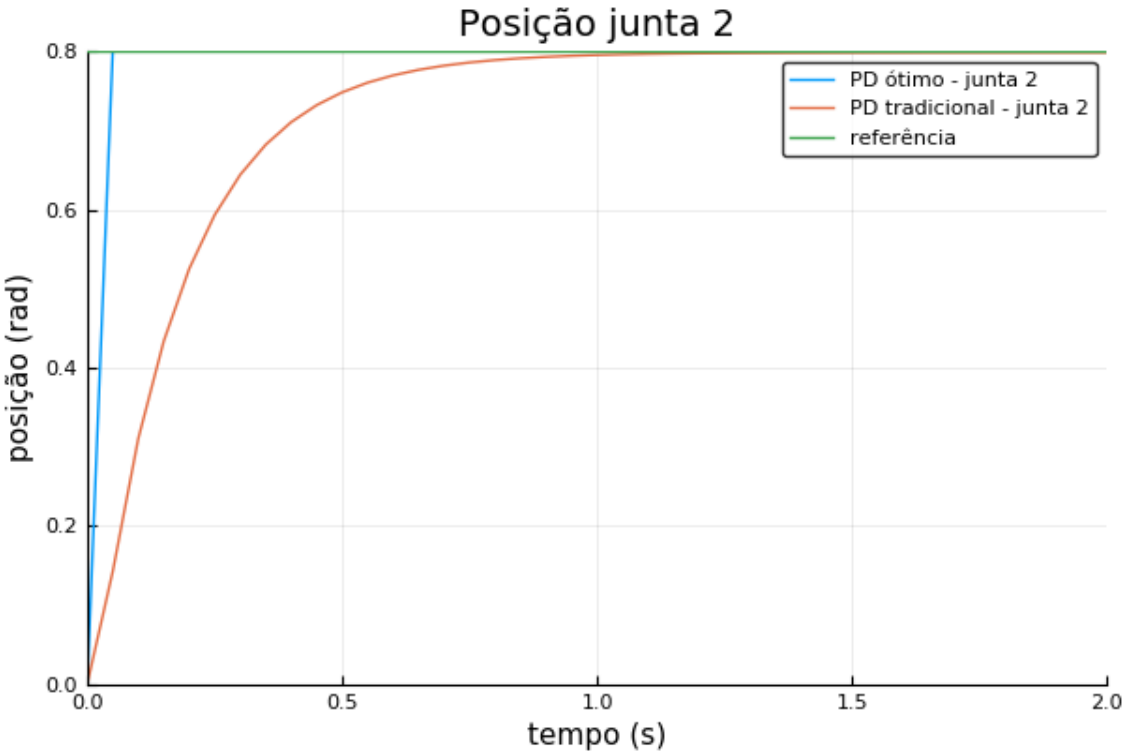
Out[26]:



4.1.2 Posição junta 2

```
In [27]: 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
```

Out[27]:

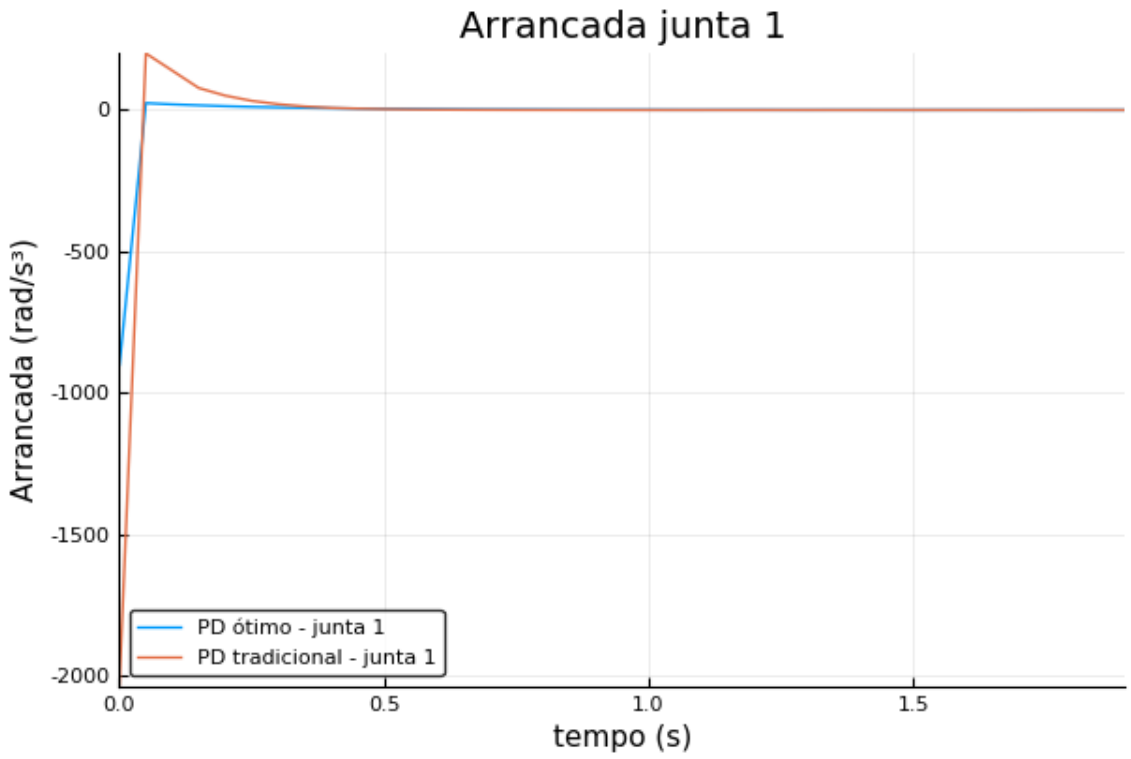


4.2 Jerk

4.2.1 Jerk junta 1

```
In [28]: plot(tj,j[1],label = "PD ótimo - junta 1")
         plot!(tj,j_pid[1],label = "PD tradicional - junta 1", title = "Arrancada junta 1", xla
```

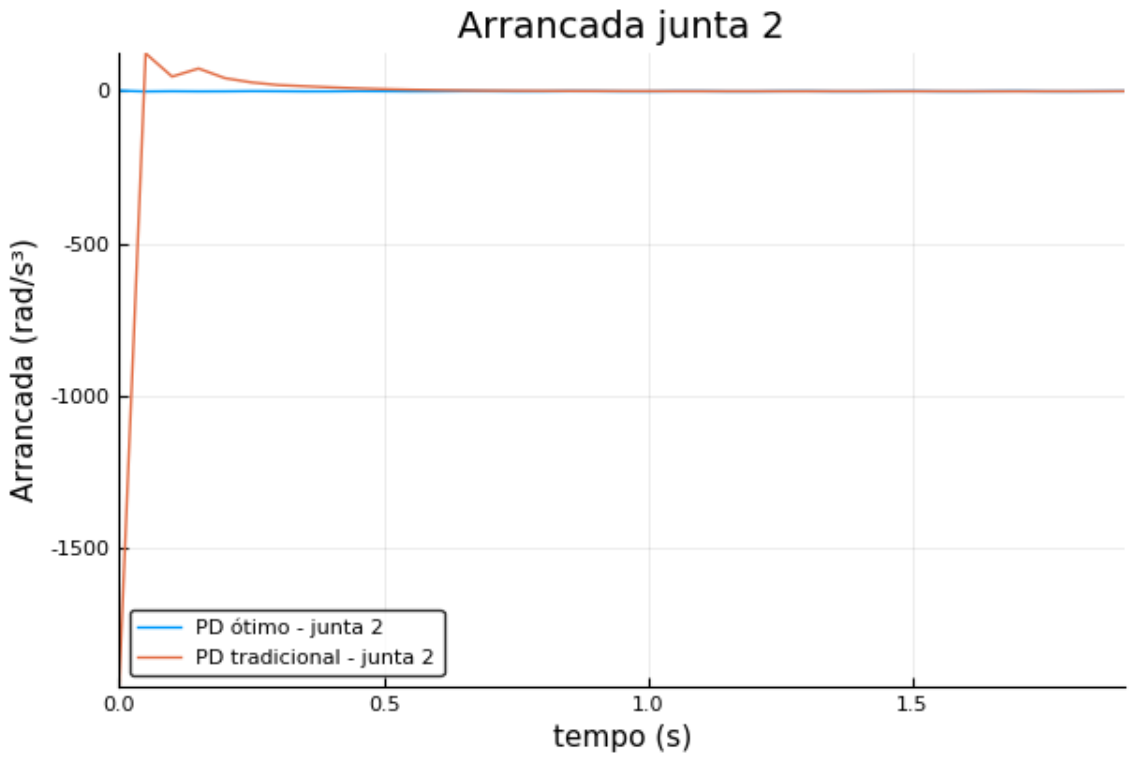
Out[28]:



4.2.2 Jerk junta 2

```
In [29]: plot(tj,j[2],label = "PD ótimo - junta 2")
         plot!(tj,j_pid[2],label = "PD tradicional - junta 2", title = "Arrancada junta 2", xla
```

Out[29]:

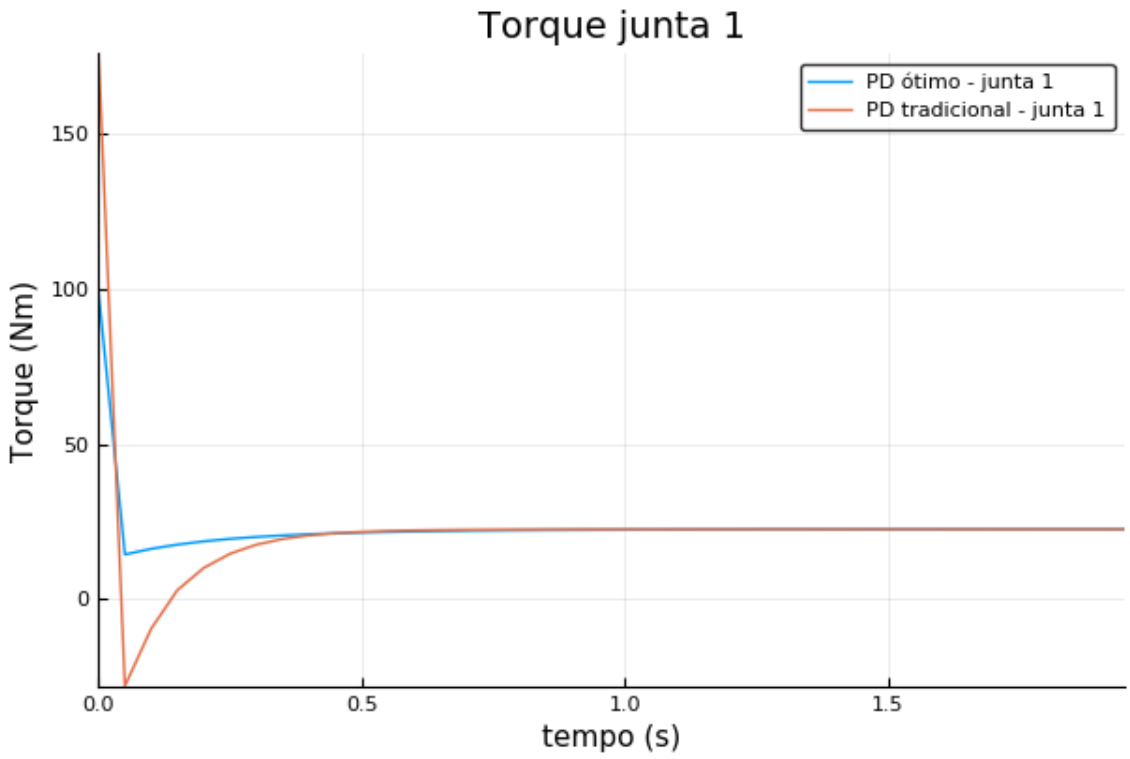


4.3 Torque

4.3.1 Torque junta 1

```
In [30]: 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",
```

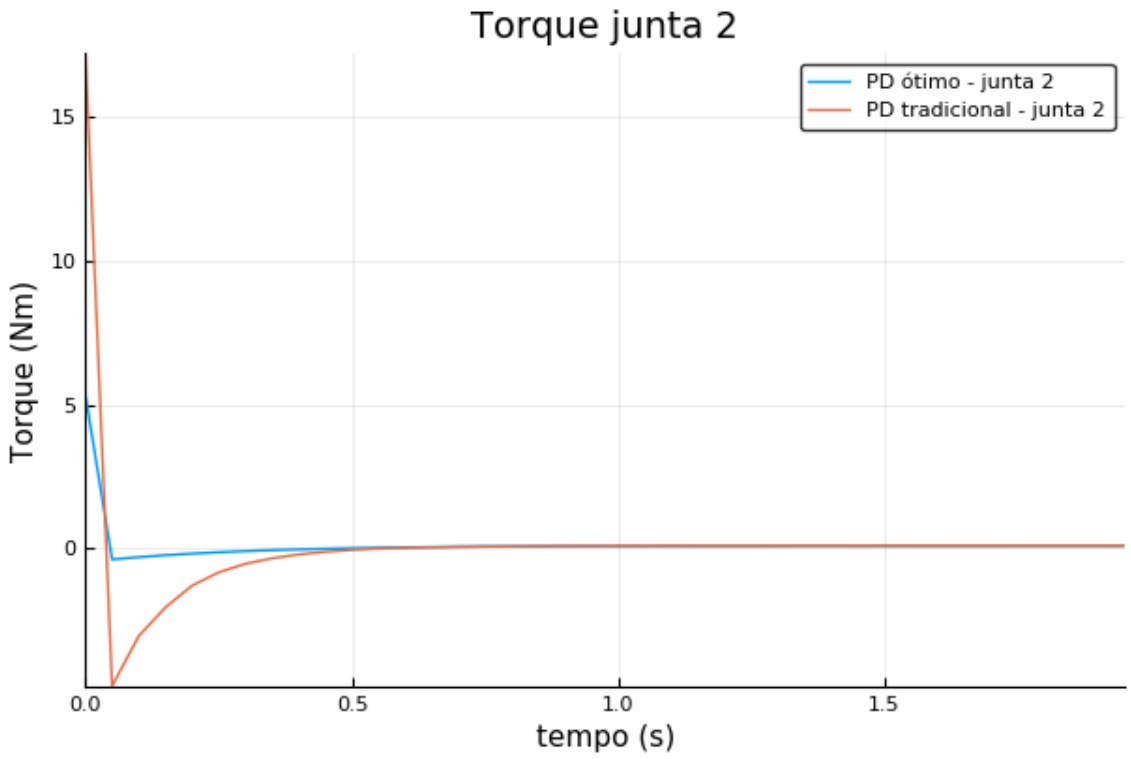
Out[30]:



4.3.2 Torque junta 2

```
In [31]: 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[31]:

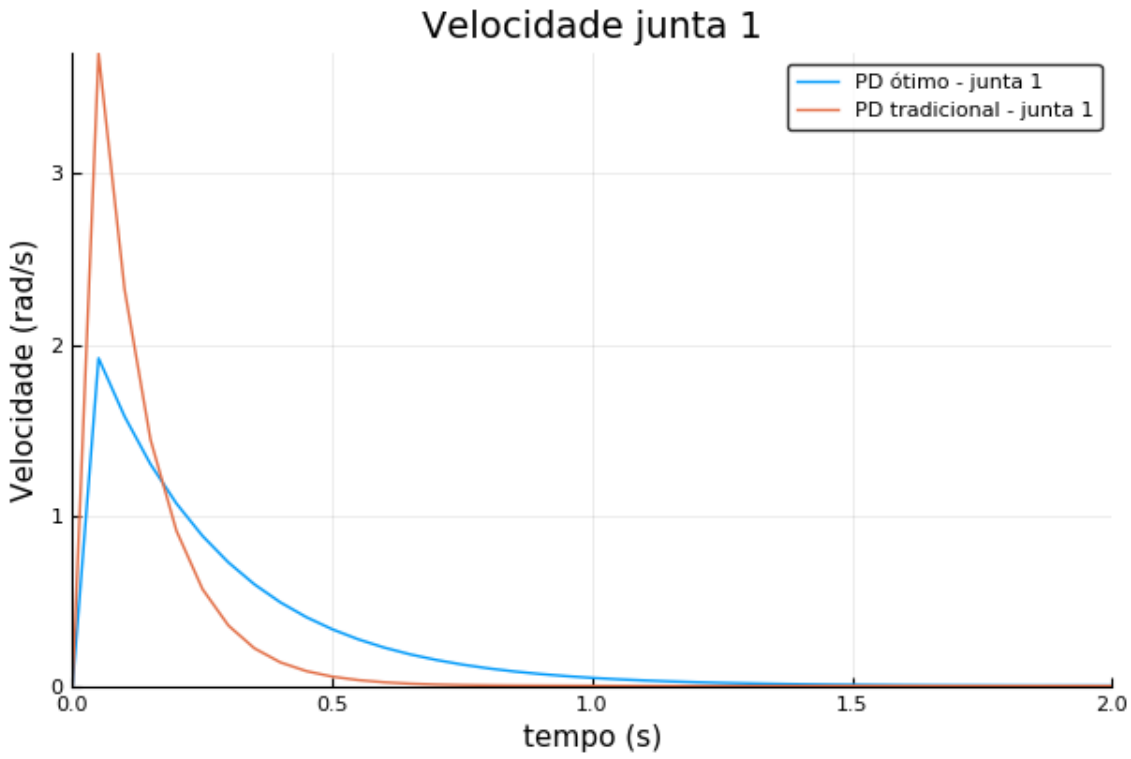


4.4 Velocidade

4.4.1 Velocidade junta 1

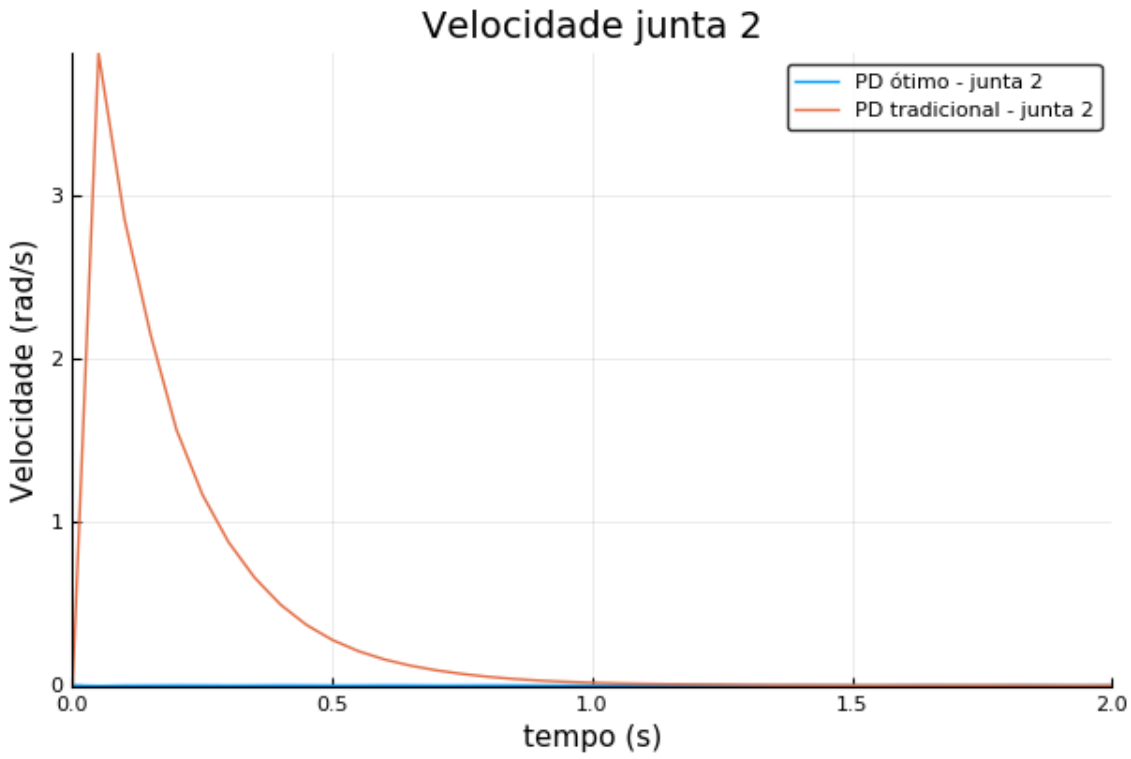
```
In [32]: plot(t,v[1],label = "PD ótimo - junta 1")
         plot!(t,v_pid[1],label = "PD tradicional - junta 1", title = "Velocidade junta 1", xla
```

Out[32]:



4.4.2 Velocidade junta 2

```
In [33]: plot(t,v[2],label = "PD ótimo - junta 2")
         plot!(t,v_pid[2],label = "PD tradicional - junta 2", title = "Velocidade junta 2", xla

   Out[33]:
```

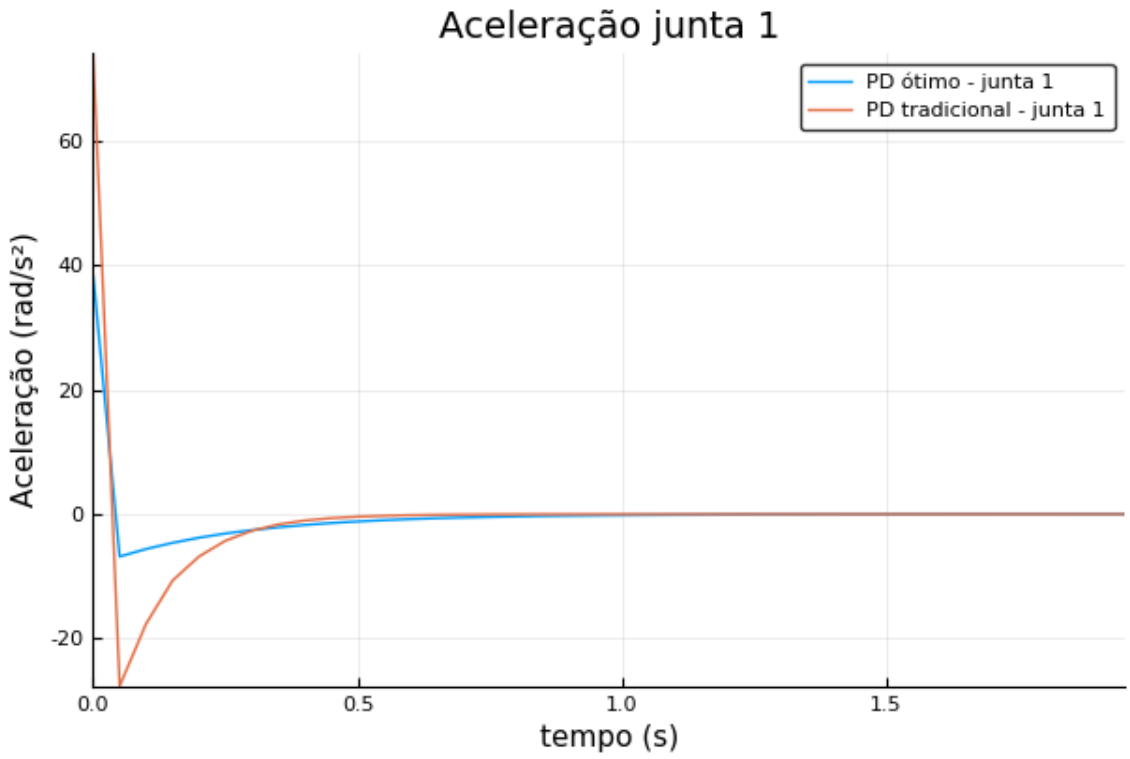


4.5 Aceleração

4.5.1 Aceleração junta 1

```
In [34]: plot(ta,a[1],label = "PD ótimo - junta 1")
         plot!(ta,a_pid[1],label = "PD tradicional - junta 1", title = "Aceleração junta 1", xl
```

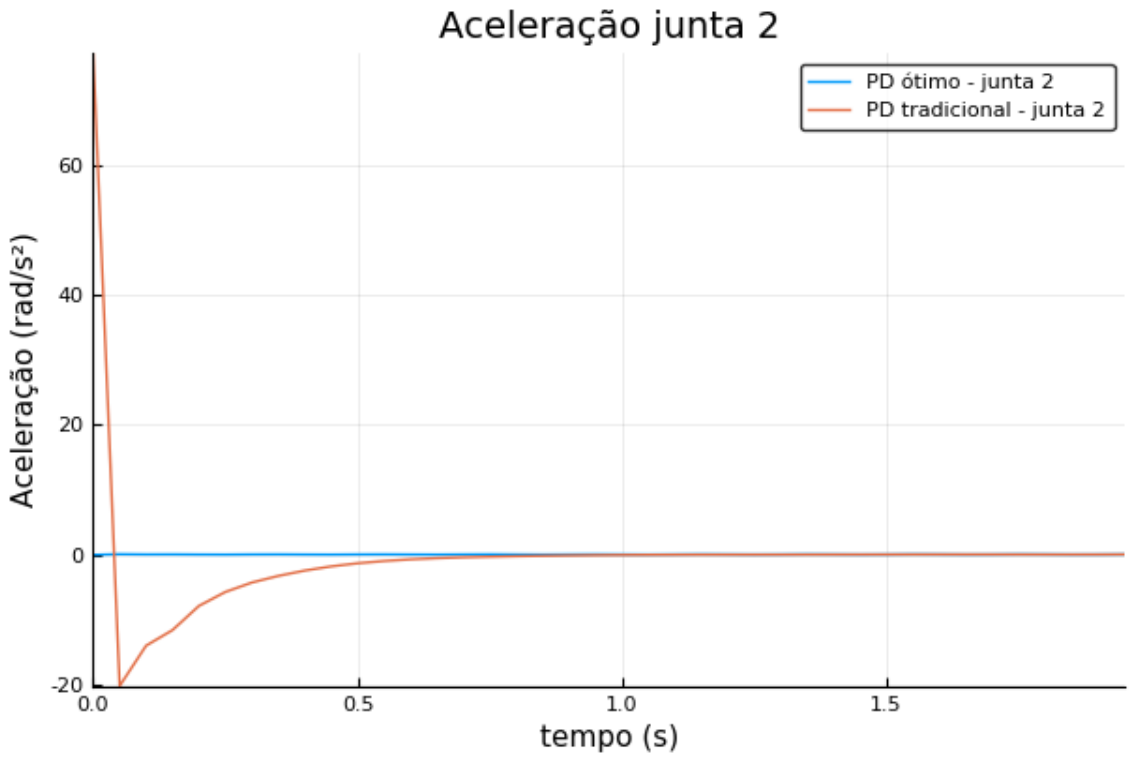
Out[34]:



4.5.2 Aceleração junta 2

```
In [35]: plot(ta,a[2],label = "PD ótimo - junta 2")
         plot!(ta,a_pid[2],label = "PD tradicional - junta 2", title = "Aceleração junta 2", xl
```

Out[35]:



5 Discussão

5.1 Sobre o torque

```
In [36]: tabela(, "Torque (ótimo)")
```

Out[36]:

—	junta 1	junta 2
Torque (ótimo) máximo	98.75	5.33
Torque (ótimo) mínimo	14.41	0.0
Torque (ótimo) total	937.44	9.37

```
In [37]: tabela(_pid, "Torque (clássico)")
```

Out[37]:

—	junta 1	junta 2
Torque (clássico) máximo	175.93	17.22
Torque (clássico) mínimo	2.96	0.0
Torque (clássico) total	993.2	33.11

5.2 Sobre o Jerk

```
In [38]: tabela(j, "Jerk (ótimo)")
```

Out[38]:

—	junta 1	junta 2
Jerk (ótimo) máximo	904.1	1.95
Jerk (ótimo) mínimo	0.0	0.1
Jerk (ótimo) total	1040.6	14.55

```
In [39]: tabela(j_pid, "Jerk (clássico)")
```

Out[39]:

—	junta 1	junta 2
Jerk (clássico) máximo	2034.53	1954.23
Jerk (clássico) mínimo	0.0	0.01
Jerk (clássico) total	2587.32	2361.25

5.2.1 Sobre a aceleração

```
In [40]: tabela(a, "Aceleração (ótimo)")
```

Out[40]:

—	junta 1	junta 2
Aceleração (ótimo) máximo	38.39	0.06
Aceleração (ótimo) mínimo	0.0	0.0
Aceleração (ótimo) total	76.76	0.49

```
In [41]: tabela(a_pid, "Aceleração (clássico)")
```

Out[41]:

—	junta 1	junta 2
Aceleração (clássico) máximo	74.1	77.4
Aceleração (clássico) mínimo	0.0	0.0
Aceleração (clássico) total	148.22	154.83

5.2.2 Sobre a velocidade

```
In [42]: tabela(v, "Velocidade (ótimo)")
```

Out[42]:

—	junta 1	junta 2
Velocidade (ótimo) máximo	1.92	0.0
Velocidade (ótimo) mínimo	0.0	0.0
Velocidade (ótimo) total	10.82	0.02

```
In [43]: tabela(v_pid, "Velocidade (clássico)")
```

Out[43]:

—	junta 1	junta 2
Velocidade (clássico) máximo	3.7	3.87
Velocidade (clássico) mínimo	0.0	0.0
Velocidade (clássico) total	9.89	15.17