Códigos utilizados no trabalho do G19, escrito para rodar em *Scilab*.
Adaptado e melhorado em relação ao código utilizado em projeto do ano anterior.

Para obter os resultados das simulações alterar o valor conforme dados utilizados e indicados no corpo do texto do projeto.

Evandro Uehara Viaro – 4364056
Gabriel Apicella Giannoni – 9348290
José Felipe Félix Rafael – 10333139

```
//================|_LINEARIZADO_VS_NÃO_LINEAR_|====================//
clear() //limpa as variáveis anteriores
//--------------------------parâmetros--------------------------//
//parâmetros hidrodinâmicos
Xu = 82.30 //coeficiente linear de arrasto em avanço [kg/s]
Xuu = 309.70 //coeficiente quadrático de arrasto em avanço [kg/m]
Yv = 8.50 //coeficiente linear de arrasto em deriva [kg/s]
Yvv = 505.45 //coeficiente quadrático de arrasto em avanço [kg/s]
Kr = 18.21 //coeficiente linear de arrasto em guinada [kg/s]
Krr = 94.72 //coeficiente quadrático de arrasto em guinada [kg/m]
//parâmetros do ROV
mu = 450.87 //massa virtual de avanço [kg]
mv = 558.92 //massa virtual de deriva [kg]
Jr = 239.44 //inércia virtual de guinada [kg*m^2]
//-----------------------------tempo-----------------------------//
t0 = 0 //tempo inicial [s]
dt = 0.01 //passo de integração [s]
tspan = 60 //intervalo de integração [s]
t = t0:dt:tspan //vetor linha do tempo [s]
//----------------------trajetória_nominal----------------------//
//referenciada ao SC móvel
//no nosso caso, definida por velocidades constantes
//velocidades nominais
un = 1 // velocidade de avanço [m/s]
vn = 0 // velocidade de deriva [m/s]
rn = 0.5 // velocidade de guinada [rad/s]
//deslocamentos nominais (lineares no tempo)
xn = un*t //avanço [m]
yn = vn*t //deriva [m]
psin = rn*t //guinada [rad]
//vetor das condiçoes iniciais da trajetória nominal
Xn0 = [xn(1);un;yn(1);vn;psin(1);rn]
//entradas nominais
Fun = -mv*vn*rn+Xu*un+Xuu*abs(un)*un //força dos propulsores para avanço
[N]
Fvn = mu*un*rn+Yv*vn+Yvv*abs(vn)*vn //força dos propulsores para deriva
[N]
Trn = (mv-mu)*vn*un+Kr*rn+Krr*abs(rn)*rn //momento dos propulsores para
guinada [Nm]
//vetor de entradas nominais
Un = [Fun;Fvn;Trn]
//---------------------condições_iniciais---------------------//
//referenciadas no SC móvel
x0 = 0 //posição x [m]
u0 = 0 //velocidade de avanço (em x) [m/s]
y0 = 0 //posição em y [m]
v0 = 0 //velocidade de deriva (em y) [m/s]
psi0 = 0 //posição em torno de z [rad]
r0 = 0 //velocidade de guinada (em torno de z) [rad/s]
//vetor das condições iniciais
```

```
X0 = [x0;u0;y0;v0;psi0;r0]
//---------------------------entradas----------------------------//
//magnitude das entradas
Fu = Fun //força de propulsão para avanço [N]
Fv = Fvn //força de propulsão para deriva [N]
Tr = Trn //momento de propulsão para guinada [Nm]
//vetor de entradas
U = [Fu;Fv;Tr]
//-----------------------espaço_de_estados-----------------------//
//variáveis de estado (referenciados no SC móvel)
//x1 = x | movimento de avanço (em x) [m]
//x2 = u | velocidade de avanço (em x) [m/s]
//x3 = y | movimento de deriva (em y) [m]
//x4 = v | velocidade de deriva (em y) [m/s]
//x5 = psi | movimento de guinada (em torno de z) [rad]
//x6 = r | velocidade de guinada (em torno de z) [rad/s]
//espaço de estados
function dx=ROV(t, x, U)
    dx(1) = x(2)
    dx(2) = (x(4)*x(6)*mv-x(2)*Xu-x(2)*abs(x(2))*Xuu+U(1))/mu
    dx(3) = x(4)
    dx(4) = (-x(2)*x(6)*mu-x(4)*Yv-x(4)*abs(x(4))*Yvv+U(2))/mv
    dx(5) = x(6)
    dx(6) = (x(4)*x(2)*(mu-mv)-x(6)*Kr-x(6)*abs(x(6))*Krr+U(3))/Jr
endfunction
//----------------------sistema_linearizado---------------------//
//d(deltaX)/dt = A*deltaX + B*deltaU
//deltaY = C*deltaX + D*deltaU
//A(nxn): matriz dinâmica
A = [0 , 1 , 0 , 0 , 0 , 0 ;..
0 , -(Xu+2*Xuu*abs(un))/mu , 0 , rn*mv/mu , 0 , vn*mv/mu ;..
0 , 0 , 0 , 1 , 0 , 0 ;..
0 , -rn*mu/mv , 0 , -(Yv+2*Yvv*abs(vn))/mv , 0 , -un*mu/mv ;..
0 , 0 , 0 , 0 , 0 , 1 ;..
0 , vn*(mu-mv)/Jr , 0 , un*(mu-mv)/Jr , 0 , -(Kr+2*Krr*abs(rn))/Jr ]
//B(nxp): matriz de entrada
B = [ 0 , 0 , 0 ;..
1/mu , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 1/mv , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 1/Jr ]
//C(mxn): matriz de saídas
C = [ 1 , 0 , 0 , 0 , 0 , 0 ;..
0 , 1 , 0 , 0 , 0 , 0 ;..
0 , 0 , 1 , 0 , 0 , 0 ;..
0 , 0 , 0 , 1 , 0 , 0 ;..
0 , 0 , 0 , 0 , 1 , 0 ;..
0 , 0 , 0 , 0 , 0 , 1 ]
//D(mxp): matriz de incidência direta
D = [ 0 , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 0 ]
//define o sistema linear
ROVlin = syslin('c',A,B,C,D) //'c' indica que é contínuo
//vetor de variação da força dos propulsores
deltaU = (U-Un)*ones(t)
//vetor de variação das condições iniciais
deltaX0 = X0-Xn0
//--------------------------simulações--------------------------//
```

```
//sistema não linear (ODE)
X = ode(X0,t0,t,ROV)
//sistema linearizado (csim)
deltaX = csim(deltaU,t,ROVlin,deltaX0)
//---------------------------resultados--------------------------//
//referenciados no SC móvel
//sistema não linear
u = X(2,:) //velocidade de avanço (em x) [m/s]
v = X(4,:) //velocidade de deriva (em y) [m/s]
r = X(6,:) //velocidade de guinada (em torno de z) [rad/s]
psi = X(5,:) //guinada (em torno de z) [rad]
//sistema linearizado
u_lin = deltaX(2,:)+un //velocidade de avanço [m/s]
v_lin = deltaX(4,:)+vn //velocidade de deriva [m/s]
r_lin = deltaX(6,:)+rn //velocidade de guinada [rad/s]
psi_lin = deltaX(5,:)+psin //guinada [rad]
//-----------------------mudança_de_base-------------------------//
//mudança das velocidades da base móvel para a base inercial
//sistema não linear
uI = cos(psi).*u-sin(psi).*v //velocidade de avanço no SC inercial [m/s]
vI = -sin(psi).*u-cos(psi).*v //velocidade de guinada no SC inercial [m/s]
//sistema linearizado
uI_lin = cos(psi_lin).*u_lin-sin(psi_lin).*v_lin //velocidade de avanço no
SC inercial [m/s]
vI_lin = -sin(psi_lin).*u_lin-cos(psi_lin).*v_lin //velocidade de guinada
no SC inercial [m/s]
//velocidades nominais
unI = cos(psin).*un-sin(psin).*vn //velocidade de avanço no SC inercial
[m/s]
vnI = -sin(psin).*un-cos(psin).*vn //velocidade de guinada no SC inercial
[m/s]
// .* faz a multiplicação termo a termo de duas matrizes de mesma dimensão
//---------------------------integração---------------------------//
//função que integra um intervalo de valores no tempo
function F=integra(f, t, F0)
    F= zeros(f)              //cria um vetor com as dimensões do intervalo
a ser integrado
    F(1) = F0                //define a condição inicial
    for i = 2:1:length(f)    //varre o intervalo (inclusive os extremos)
com passo 1
        //integração numérica por trapézios
        F(i) = inttrap(t(1:i),f(1:i))
    end
endfunction
//as posições no tempo são obtidas integrando-se as velocidades

//sistema não linear
xI = integra(uI,t,0) //posição x no SC inercial [m]
yI = integra(vI,t,0) //posição y no SC inercial [m]
psiI = -psi //posição psi (em torno de z) no SC inercial [rad]

//sistema linearizado
xI_lin = integra(uI_lin,t,0) //posição x no SC inercial [m]
yI_lin = integra(vI_lin,t,0) //posição y no SC inercial [m]
psiI_lin = -psi_lin //posição psi (em torno de z) no SC inercial [rad]

//deslocamento nominal
xnI = integra(unI,t,0) //posição x no SC inercial [m]
ynI = integra(vnI,t,0) //posição y no SC inercial [m]
psinI = -psin //posição psi (em torno de z) no SC inercial [rad]
//----------------------------plots-------------------------------//
xset('window',1)
plot2d(t,[u',u_lin',un*ones(t)'])//u' = u transposto
```

```
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Velocidade de avanço','t [s]','u [m/s]')
xgrid(1)

xset('window',2)
plot2d(t,[xI',xI_lin',xnI'])
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Avanço','t [s]','x [m]')
xgrid(1)

xset('window',3)
plot2d(t,[v',v_lin',vn*ones(t)'])
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Velocidade de deriva','t [s]','v [m/s]')
xgrid(1)

xset('window',4)
plot2d(t,[yI',yI_lin',ynI'])
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Deriva','t [s]','y [m]')
xgrid(1)

xset('window',5)
plot2d(t,[r',r_lin',rn*ones(t)'])
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Velocidade de guinada','t [s]','r [rad/s]')
xgrid(1)

xset('window',6)
plot2d(t,[psiI',psiI_lin',psinI'])
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Guinada','t [s]','psi [rad]')
xgrid(1)

xset('window',7)
plot2d([xI',xI_lin',xnI'],[yI',yI_lin',ynI'])
legend('Não linearizado','Linearizado','Trajetória nominal')
xtitle('Trajetória','x [m]','y [m]')
xgrid(1)
```

```
//======|_FUNÇÕES_DE_TRANSFERÊNCIA-RESPOSTA_EM_FREQUÊNCIA_|=======//
clear() //limpa as variáveis anteriores
//----------------------------parâmetros--------------------------//
///parâmetros hidrodinâmicos
Xu = 82.30 //coeficiente linear de arrasto em avanço [kg/s]
Xuu = 309.70 //coeficiente quadrático de arrasto em avanço [kg/m]
Yv = 8.50 //coeficiente linear de arrasto em deriva [kg/s]
Yvv = 505.45 //coeficiente quadrático de arrasto em avanço [kg/s]
Kr = 18.21 //coeficiente linear de arrasto em guinada [kg/s]
Krr = 94.72 //coeficiente quadrático de arrasto em guinada [kg/m]
//parâmetros do ROV
mu = 450.87 //massa virtual de avanço [kg]
mv = 558.92 //massa virtual de deriva [kg]
Jr = 239.44 //inércia virtual de guinada [kg*m^2]
//----------------------velocidades_nominais---------------------//
un = 0.5 //velocidade de avanço [m/s]
vn = 0.5 //velocidade de deriva [m/s]
rn = 0 //velocidade de guinada [rad/s]
//-----------------------sistema_linearizado---------------------//
//d(deltaX)/dt = A*deltaX + B*deltaU
//deltaY = C*deltaX + D*deltaU
```

```
//A(nxn): matriz dinâmica
A = [0 , 1 , 0 , 0 , 0 , 0 ;..
0 , -(Xu+2*Xuu*abs(un))/mu , 0 , rn*mv/mu , 0 , vn*mv/mu ;..
0 , 0 , 0 , 1 , 0 , 0 ;..
0 , -rn*mu/mv , 0 , -(Yv+2*Yvv*abs(vn))/mv , 0 , -un*mu/mv ;..
0 , 0 , 0 , 0 , 0 , 1 ;..
0 , vn*(mu-mv)/Jr , 0 , un*(mu-mv)/Jr , 0 , -(Kr+2*Krr*abs(rn))/Jr ]
//B(nxp): matriz de entrada
B = [ 0 , 0 , 0 ;..
1/mu , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 1/mv , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 1/Jr ]
//C(mxn): matriz de saídas
C = [ 0 , 1 , 0 , 0 , 0 , 0 ;..
0 , 0 , 0 , 1 , 0 , 0 ;..
0 , 0 , 0 , 0 , 0 , 1 ]
//D(mxp): matriz de incidência direta
D = [ 0 , 0 , 0 ;..
0 , 0 , 0 ;..
0 , 0 , 0 ]
//---------------------funções_de_transferência-------------------//
//definindo a variável complexa s
s = poly(0,'s')
//matriz resolvente (nxn)
PHI_s = inv(s*eye(A)-A)
//eye(A) retorna a matriz identidade com as dimensões de A
//denominador (polinômio de grau n)
E = det(s*eye(A)-A)
//polos (n polos)
polos = roots(E)
//matriz de transferência (mxp); G(i,j) = Y(i)/U(j)
G = syslin('c',C*PHI_s*B)
//-----------------exibindo_informações_relevantes----------------//
disp('un = '+string(un)+'; vn = '+string(vn)+'; rn = '+string(rn))
disp('E:')
disp(E)
disp('polos:')
disp(polos)
disp('G:')
disp(G)
//------------------------------plots-----------------------------//
xset('window',1)
plzr(G)
//-------------------------diagramas_de_Bode----------------------//
if G(1,1) ~= 0 then
    xset('window',2)
    bode(G(1,1))
    xtitle('Diagramas de Bode: u/F_u')
end
if G(1,2) ~= 0 then
    xset('window',3)
    bode(G(1,2))
    xtitle('Diagramas de Bode: u/F_v')
end
if G(1,3) ~= 0 then
    xset('window',4)
    bode(G(1,3))
    xtitle('Diagramas de Bode: u/T_r')
end
if G(2,1) ~= 0 then
    xset('window',5)
```

```
    bode(G(2,1))
    xtitle('Diagramas de Bode: v/F_u')
end
if G(2,2) ~= 0 then
    xset('window',6)
    bode(G(2,2))
    xtitle('Diagramas de Bode: v/F_v')
end

if G(2,3) ~= 0 then
    xset('window',7)
    bode(G(2,3))
    xtitle('Diagramas de Bode: v/T_r')
end
if G(3,1) ~= 0 then
    xset('window',8)
    bode(G(3,1))
    xtitle('Diagramas de Bode: r/F_u')
end

if G(3,2) ~= 0 then
    xset('window',9)
    bode(G(3,2))
    xtitle('Diagramas de Bode: r/F_v')
end
if G(3,3) ~= 0 then
    xset('window',10)
    bode(G(3,3))
    xtitle('Diagramas de Bode: r/T_r')
end
```