Anexo A.

En este anexo se muestra el código utilizado en Matlab® para realizar la estimación del $k_l a$, el cual es de vital importancia para encontrar el modelo matemático de la transferencia de oxigeno del biorreactor del IEIM. Los datos empleados se encuentran en el archivo *Kla2.xlsx*.

Estimación: *kla.m*

```
% Construir vector de % de oxigeno y RPMs
Oxigeno = xlsread('Kla2.xlsx','E2:E3907');
RPM = xlsread('Kla2.xlsx','B2:B3907');
VVM = xlsread('Kla2.xlsx','C2:C3907');
% Vector de tiempos
tiempo=0:10:(length(Oxigeno)-1)*10;
% Matriz de comportamiento
M = [9 315 200 93.3 0.2 ;
     1190 1384 200 92.5 0.5 ;
     1880 2265 200 95.6 1   ;
     2803 2960 200 97.8 1.5 ;
     3335 3495 200 98.7 2   ;
     540 647 500 94.9 0.2 ;
     1460 1523 500 95.8 0.5 ;
     2474 2522 500 96.8 1   ;
     3133 3172 500 97.5 1.5 ;
     3575 3609 500 100.0 2   ;
     733 801 700 94.3 0.2 ;
     1582 1629 700 96.3 0.5 ;
     2582 2613 700 96.6 1   ;
     3194 3228 700 98.3 1.5 ;
     3628 3650 700 99.3 2   ;
     874 954 850 93.8 0.2 ;
     1673 1747 850 96.4 0.5 ;
     2667 2688 850 96.2 1   ;
     3251 3260 850 97.0 1.5 ;
     3667 3687 850 99.3 2   ;
     1052 1130 950 93.4 0.2 ;
     1790 1842 950 95.7 0.5 ;
     2749 2782 950 96.4 1   ;
     3289 3327 950 98.5 1.5 ;
     3703 3720 950 98.5 2   ];

%Gráfica del progreso del porcentaje de oxigeno y RPMs
figure
subplot(2,2,1)
plot(tiempo,RPM)
ylabel('Agitacion [rpm]')
xlabel('Tiempo [s]')
subplot(2,2,2)
plot(tiempo,VVM)
ylabel('Flujo [L/min]')
xlabel('Tiempo [s]')
subplot(2,2,3:4)
plot(tiempo,Oxigeno)
ylabel('Oxigeno disuelto [%]')
xlabel('Tiempo [s]')
set(gcf,'color','w')

% Parámetros del sistema
dt= 10; % Tiempo de muestreo
alpha= 0.0278; % Inverso de la constante de tiempo del sensor (90/3600)=0.0278
epsilon= 0.07; % Tolerancia de error de la diferencia real a la calculada
options =optimset('Disp','off','MaxIter',10000,'MAxFunEval', 1000);
for i=1:25
    t_inicial=M(i,1); %Tiempo inicial de cada exponencial en el vector de datos
    t_final=M(i,2); %Tiempo final de cada exponencial en el vector de datos
```

```
    y=Oxigeno(t_inicial:t_final)/M(i,4); %Normalizacion de los valores que representan cada una de las exponenciales
    K0 = 0.5*alpha; %Condicion inicial para el problema de optimización
    %Limites inferior y superior para la restricción  (LB <= k <= UB)
    LB = 1e-5;
    UB = 1000/3600;
    %Vector de restricciones no-lineales en funcion de la variable k (c(k)<=0 ; ceq(k)=0)
    ConjuntoFPS = @(k)FPS(k,y,epsilon,alpha,dt);
    %Minimización y maximización para encontrar el valor inferior y superior del Kla
    [Kla_min(i,1),~,Bandera_min(i,1)] = fmincon(@(k)k,K0,[],[],[],[],LB,UB,ConjuntoFPS,options);
    [Kla_max(i,1),~,Bandera_max(i,1)] = fmincon(@(k)-k,K0,[],[],[],[],LB,UB,ConjuntoFPS,options);
end

%Determinar las soluciones factibles y no factibles
T = size(Bandera_min);
Kla_Opt = ones(T(1),1);
for i=1:T(1)
   Kla_Opt(i,1) = 0.5*(Kla_max(i,1)+Kla_min(i,1))*3600;
   Error(i,1) = (Kla_max(i,1)*3600)-Kla_Opt(i,1);
end

%Relación KLa - w
figure
Fin = [0.2 0.5 1 1.5 2];
for i=1:5
        d= (5*(i-1)) + (1:5);
        if i==1
        errorbar(Fin,Kla_Opt(d,1),Error(d,1),'m','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Flujo de aire [VVM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('w = 200 RPM');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==2
        hold on
        errorbar(Fin,Kla_Opt(d,1),Error(d,1),'b','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Flujo de aire [VVM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('w = 500 RPM');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==3
        hold on
        errorbar(Fin,Kla_Opt(d,1),Error(d,1),'g','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Flujo de aire [VVM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('w = 700 RPM');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==4
        hold on
        errorbar(Fin,Kla_Opt(d,1),Error(d,1),'c','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Flujo de aire [VVM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('w = 850 RPM');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==5
        hold on
        errorbar(Fin,Kla_Opt(d,1),Error(d,1),'r','LineWidth',1.5);
        grid on
```

```
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Flujo de aire [VVM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('w = 950 RPM');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
end
h = legend('w = 200 [RPM]','w = 500 [RPM]','w = 700 [RPM]','w = 850 [RPM]','w = 950 [RPM]');
set(h,'Interpreter','none')

% Relación KLa - fin
h=1;
for k=1:5
    d=k;
    for j=1:5
    curvas(h,1)= Kla_Opt(d,1);
    Kla_fin(j,k)=Kla_Opt(d,1);
    d=d+5;
    h=h+1;
    end
end
h=1;
for k=1:5
    d=k;
    for j=1:5
    Ecurvas(h,1)= Error(d,1);
    d=d+5;
    h=h+1;
    end
end
figure
W= [200 500 700 850 950];
for i=1:5
        d= (5*(i-1)) + (1:5);
        if i==1
        errorbar(W,curvas(d,1),Ecurvas(d,1),'m','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Velocidad de agitación [RPM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('Fin = 0.2 [VVM]');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==2
        hold on
        errorbar(W,curvas(d,1),Ecurvas(d,1),'b','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Velocidad de agitación [RPM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('Fin = 0.5 [VVM]');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==3
        hold on
        errorbar(W,curvas(d,1),Ecurvas(d,1),'g','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Velocidad de agitación [RPM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('Fin = 1 [VVM]');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==4
        hold on
        errorbar(W,curvas(d,1),Ecurvas(d,1),'c','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
```

```
        xlabel('Velocidad de agitación [RPM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('Fin = 1.5 [VVM]');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
        if i==5
        hold on
        errorbar(W,curvas(d,1),Ecurvas(d,1),'r','LineWidth',1.5);
        grid on
        title(sprintf('Kla para una tolerancia de %g',epsilon))
        xlabel('Velocidad de agitación [RPM]')
        ylabel('K_{La} [h^{-1}]')
        h = legend('Fin = 2 [VVM]');
        set(h,'Interpreter','none')
        set(gcf,'color','w')
        end
end
h = legend('Fin = 0.2 [VVM]','Fin = 0.5 [VVM]','Fin = 1 [VVM]','Fin = 1.5 [VVM]','Fin = 2 [VVM]');
set(h,'Interpreter','none')

% Gráfico 3D - Resultados de Kla óptimo
figure
[X,Y]=meshgrid([0.2,0.5,1,1.5,2],[200,500,700,850,950]);
Z=zeros(5,5);
k=1;
for i=1:5
    for j=1:5
        Z(i,j)=Kla_Opt(k);
        k=k+1;
    end
end
surf(X,Y,Z)
colormap(autumn)
xlabel('Flujo de aire [VVM]');
ylabel('Agitación [RPM]');
zlabel('K_{La} [h^{-1}]');
set(gcf,'color','w')

% Interpolacion del Kla
xq=1:0.01:5;
v=interp1(Kla_fin,xq);
AW(1:101)=200:(500-200)/100:500;
AW(101:201)=500:(700-500)/100:700;
AW(201:301)=700:(850-700)/100:850;
AW(301:401)=850:(950-850)/100:950;

% Sacar las rectas de agitacion
for i=1:5
Aprox(1,i)=((sum(W'.*Kla_fin(:,i))-((sum(Kla_fin(:,i)).*sum(W'))/5))/(sum(W'.^2)-((sum(W')).^2/5)));
Aprox(2,i)=(sum(Kla_fin(:,i))/5)-(Aprox(1,i)*(sum(W')/5));

figure
z=(Aprox(1,i).*W)+Aprox(2,i);
plot(W,z,'r--','LineWidth',1.5)
hold on
d= (5*(i-1)) + (1:5);
errorbar(W,curvas(d,1),Ecurvas(d,1),'LineWidth',1);
grid on
xlabel('Agitación [RPM]')
ylabel('K_{La} [h^{-1}]')
valor=Fin(i);
title(['Flujo de aire ',num2str(valor),' VVM'])
legend('Aproximacion','Valor estimado')
set(gcf,'color','w')
end

for k=1:5
Aprox(1,5+k)=((sum(Fin'.*Kla_fin(k,:)')-((sum(Kla_fin(k,:)').*sum(Fin'))/5))/(sum(Fin'.^2)-((sum(Fin')).^2/5)));
Aprox(2,5+k)=(sum(Kla_fin(k,:)')/5)-(Aprox(1,5+k)*(sum(Fin')/5));
end
```

Comprobación del modelo:

```
Csonda=xlsread('carsonda.xlsx','E2:G26');
Tm=0:10:249;
figure
plot(Tm,Csonda)
grid on
ylabel('COD [%]')
xlabel('Tiempo [s]')
title('Caracterizacion de la sonda')
legend('Exp. 1','Exp. 2','Exp. 3')
set(gcf,'color','w')

% Validacion
for i=1:2
if i==1
    FE=1.5;
    WE=850;
    sim('validacion')
    figure
    subplot(1,2,1)
    plot(com.Time(1:21)*3600,com.Data(1:21))%valor estimado
    hold on
    plot(Tm(1:end-1),Csonda(2:end,3))%Experimento
    size(com.Time)
    grid on
    ylabel('COD [%]')
    xlabel('Tiempo [s]')
    title('a) Fin=1.5 vvm  w=850rpm')
    legend('respuesta del Kla_opt','Comportamiento real')
    prom=sum(Csonda(1:19,3))/length(Csonda(1:19,3));
    indFIT(i)=100*(1-(norm(Csonda(1:19,3)-com.Data(1:19),2)/norm(com.Data(1:19)-prom,2))); %indice fit
 else
    FE=0.5;
    WE=500;
    sim('validacion')
    cs=xlsread('carsonda.xlsx','M2:M65');
    Tm=0:10:639;
    subplot(1,2,2)
    plot(com.Time*3600,com.Data) %valor estimado
    hold on
    plot(Tm(1:end-3),cs(4:end)) %experimento
    grid on
    ylabel('COD [%]')
    xlabel('Tiempo [s]')
    title('b) Fin=0.5 vvm  w=500rpm')
    set(gcf,'color','w')
    legend('respuesta con Kla_opt','Comportamiento real')
    prom=sum(cs(4:end))/length(cs(4:end));
    indFIT(i)=100*(1-(norm(cs(4:end)-com.Data(1:61),2)/norm(com.Data(1:61)-prom,2))); %indice fit
 end

end
```

Anexo B.

En este anexo se explica el proceso que se llevó a cabo para la sintonización del controlador de rango medio modificado, a partir de los dos controladores PI que manipulan las variables de control.

B.1. Sintonización del controlador PI de velocidad de agitación.

El diagrama empleado para realizar la sintonización de cada uno de los controladores PI del conjunto, correspondiente a los cinco puntos de operación del flujo entrante de aire de la ecuación 1.1 en la herramienta Control System Designer de Matlab® versión 2017a se observa en la Fig. 1.

$$F_{in} = [0.2; 0.5; 1.0; 1.5; 2.0]_{vvm} \tag{1.1}$$

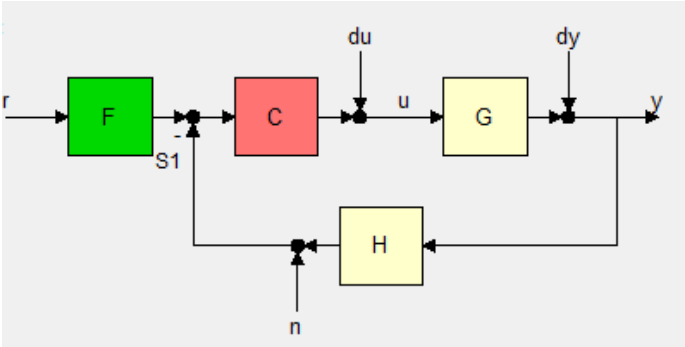

Figura 1. Diagrama de bloques de cada controlador del conjunto.

Donde C es el controlador clásico PI, H es la función de trasferencia de la sonda de oxígeno, F es unitario, dy, du y n son iguales a cero y G es la función de trasferencia correspondiente al modelo lineal de acuerdo con la constante de flujo entrante de aire, el código que encuentra estas funciones se estipula a continuación:

```
CO=0; % Punto de equilibrio para la concentracion de oxigeno disuelto
s=tf('s');
w_op=450; % Punto de equilibrio para la agitacion
sensor=1/((alpha*s)+1); % Funcion de transferencia para el sensor Knick

k1=(Aprox(1,1)*(100-CO))/((Aprox(1,1)*w_op)+Aprox(2,1));
Tao1=1/((Aprox(1,1)*w_op)+Aprox(2,1));
sys_lineal1=(k1/((Tao1*s)+1)); %Flujo 0.2

k2=(Aprox(1,2)*(100-CO))/((Aprox(1,2)*w_op)+Aprox(2,2));
Tao2=1/((Aprox(1,2)*w_op)+Aprox(2,2));
sys_lineal2=(k2/((Tao2*s)+1)); %Flujo 0.5

k3=(Aprox(1,3)*(100-CO))/((Aprox(1,3)*w_op)+Aprox(2,3));
Tao3=1/((Aprox(1,3)*w_op)+Aprox(2,3));
sys_lineal3=(k3/((Tao3*s)+1)); %Flujo 1.0

k4=(Aprox(1,4)*(100-CO))/((Aprox(1,4)*w_op)+Aprox(2,4));
Tao4=1/((Aprox(1,4)*w_op)+Aprox(2,4));
sys_lineal4=(k4/((Tao4*s)+1)); %Flujo 1.5

k5=(Aprox(1,5)*(100-CO))/((Aprox(1,5)*w_op)+Aprox(2,5));
Tao5=1/((Aprox(1,5)*w_op)+Aprox(2,5));
sys_lineal5=(k5/((Tao5*s)+1)); %Flujo 2.0
```

En la Fig. 1.1 se muestra la respuesta del sistema con el controlador para cada punto de operación, dando valides a los requerimientos establecidos.

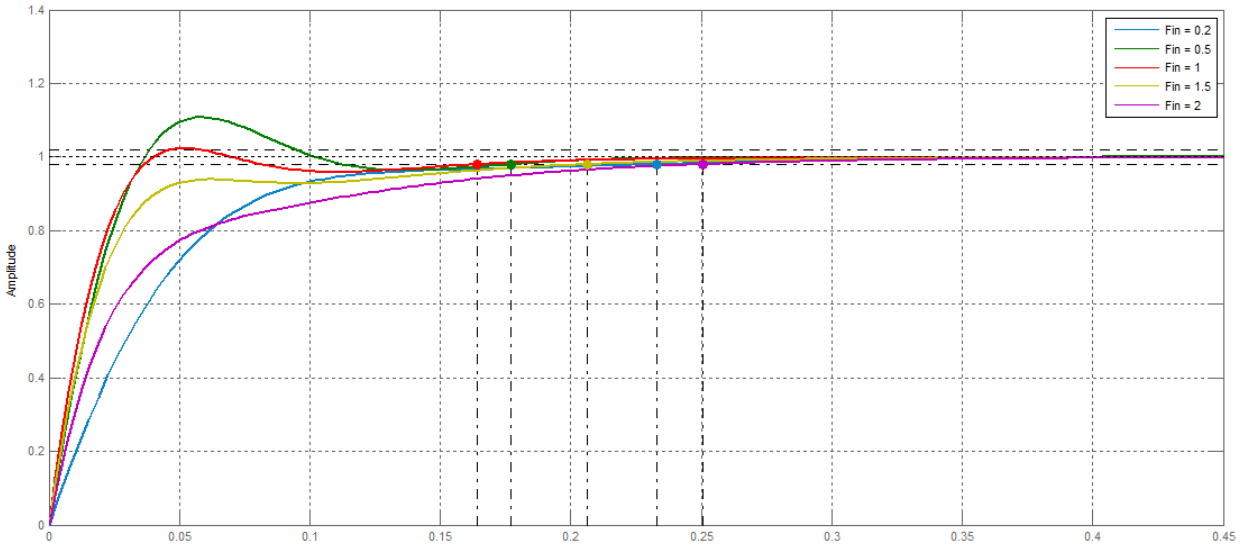

Figura 1.1. Respuesta del sistema.

B.2. Sintonización del controlador PI del flujo entrante de aire.

Para sintonizar el controlador PI de flujo entrante de aire es necesario sacar las funciones de transferencia del sistema en malla abierta para cada uno de los cinco puntos de operación de la velocidad de agitación de la ecuación 1.2, el código empleado en Matlab® se muestra a continuación:

$$\omega = [200;\ 500;\ 700;\ 850;\ 950]_{rpm} \tag{1.2}$$

```
fin_op=0.5; % punto de equilibrio para el flujo entrante de aire
CO=0; % punto de equilibrio para la concentracion de oxigeno
s=tf('s');

kf1=(Aprox(1,6)*(100-CO))/((Aprox(1,6)*fin_op)+Aprox(2,6));
Taof1=1/((Aprox(1,6)*fin_op)+Aprox(2,6));
sys_200=(-kf1/((Taof1*s)+1)); %Agitacion 200

kf2=(Aprox(1,7)*(100-CO))/((Aprox(1,7)*fin_op)+Aprox(2,7));
Taof2=2/((Aprox(1,7)*fin_op)+Aprox(2,7));
sys_500=(-kf2/((Taof2*s)+1)); %Agitacion 500

kf3=(Aprox(1,8)*(100-CO))/((Aprox(1,8)*fin_op)+Aprox(2,8));
Taof3=1/((Aprox(1,8)*fin_op)+Aprox(2,8));
sys_700=(-kf3/((Taof3*s)+1)); %Agitacion 700

kf4=(Aprox(1,9)*(100-CO))/((Aprox(1,9)*fin_op)+Aprox(2,9));
Taof4=1/((Aprox(1,9)*fin_op)+Aprox(2,9));
sys_850=(-kf4/((Taof4*s)+1)); %Agitacion 850

kf5=(Aprox(1,10)*(100-CO))/((Aprox(1,10)*fin_op)+Aprox(2,10));
Taof5=1/((Aprox(1,10)*fin_op)+Aprox(2,10));
sys_950=(-kf5/((Taof5*s)+1)); %Agitacion 500
```

Ya teniendo las funciones de trasferencia se utiliza la función pidtune de Matlab® que permite encontrar las ganancias del controlador PI como se muestra en el siguiente código:

```
[C200,inf2]=pidtune(sys_200,'PI',0.005);
[C500,inf5]=pidtune(sys_500,'PI',0.008);
[C700,inf7]=pidtune(sys_700,'PI',0.008);
[C850,inf8]=pidtune(sys_850,'PI',0.0095);
[C950,inf9]=pidtune(sys_950,'PI',0.0075);
```

La frecuencia de corte de cada controlador se escogió acorde al diagrama de bode de la función de trasferencia como se muestra en la Fig. 2, con el fin de tener una respuesta estable. Razón por la cual los valores de este parámetro son tan inferiores a 1 rad/h, ya que entre menor sea este valor más estable el sistema. En la Fig. 2.1 se muestra el diagrama de bode del sistema en malla abierta para cada condición de operación.

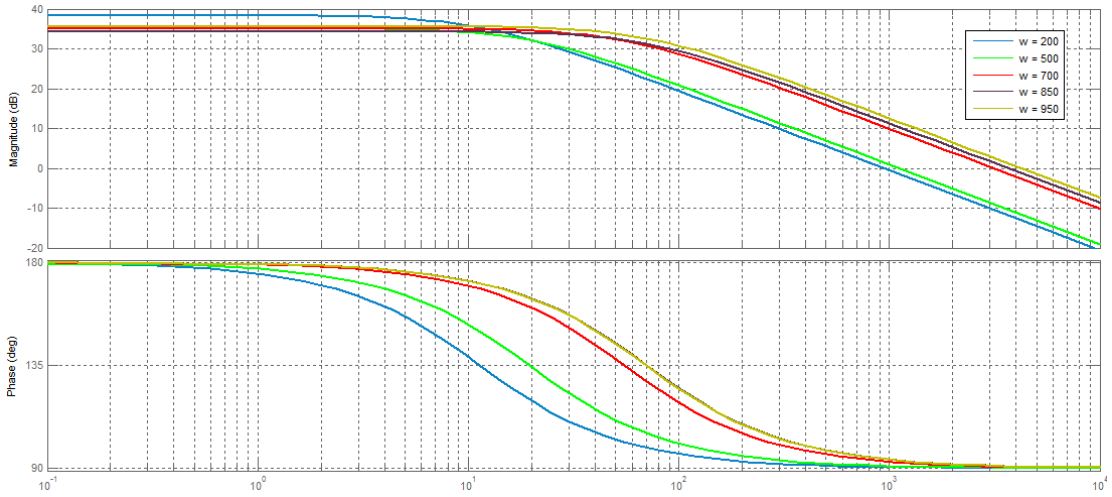

Figura 2. Diagrama de bode de las funciones de transferencia.

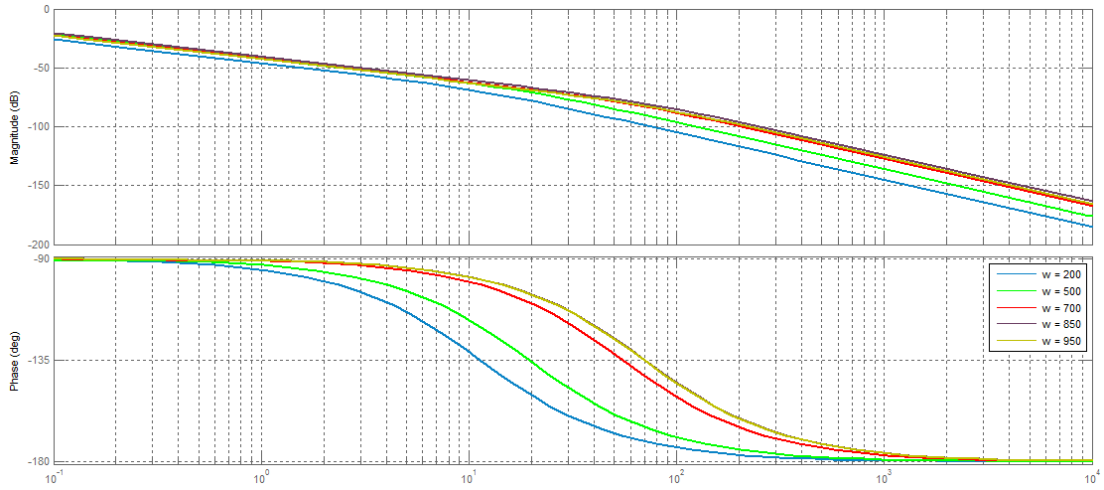

Figura 2.1. Diagrama de bode del sistema en malla abierta.

Anexo C.

C.1. Modelo de predicción.

Partiendo del modelo lineal digital de la concentración de oxígeno disuelto de la ecuación 1.1 se obtienen las predicciones de los estados a partir de los valores actuales de estos y de la señal de control.

$$\underbrace{\begin{bmatrix} \boldsymbol{COD_{\%_{k+1}}} \\ \boldsymbol{\omega_k} \\ \boldsymbol{F_{in_k}} \end{bmatrix}}_{\boldsymbol{x+1}} = \underbrace{\begin{bmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} COD_{\%_k} \\ \omega_{k-1} \\ \mathcal{F}_{in_{k-1}} \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} \boldsymbol{\mathcal{B}} & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{B}} \begin{bmatrix} \Delta\omega_k \\ \Delta\mathcal{F}_{in_k} \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{\mathcal{B}_d} \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{B_1}} OUR_{\%_k}$$

$$y_k = \underbrace{[\boldsymbol{\mathcal{C}} \quad 0 \quad 0]}_{\boldsymbol{C}} \begin{bmatrix} COD_{\%_k} \\ \omega_{k-1} \\ \mathcal{F}_{in_{k-1}} \end{bmatrix} \tag{1.1}$$

$$\begin{aligned} k = 0: \quad & x_1 = A\,x_0 + B\,\Delta u_0 + B_1\,OUR_{\%_0} \\ k = 1: \quad & x_2 = A\,x_1 + B\,\Delta u_1 + B_1\,OUR_{\%_1} \\ & = A\left(A\,x_0 + B\,\Delta u_0 + B_1\,OUR_{\%_0}\right) + B\,\Delta u_1 + B_1\,OUR_{\%_1} \\ & = A^2 x_0 + AB\,\Delta u_0 + AB_1\,OUR_{\%_0} + B\,\Delta u_1 + B_1\,OUR_{\%_1} \end{aligned}$$

$$\vdots$$

$$\begin{aligned} k = N_p - 1: \quad & x_1 = A^{N_p}x_0 + A^{N_p-1}B\,\Delta u_0 + A^{N_p-1}B_1\,OUR_{\%_0} + A^{N_p-2}B\,\Delta u_1 \\ & + A^{N_p-2}B_1\,OUR_{\%_1} + \ldots + B\,\Delta u_1 + B_1\,OUR_{\%_1} \end{aligned}$$

Con las ecuaciones anteriores se obtiene la predicción de la señal de salida:

$$\begin{aligned} k = 0: \quad & y_0 = C\,x_0 \\ k = 1: \quad & y_1 = C\,x_1 \\ & = CA\,x_0 + CB\,\Delta u_0 + C\,B_1\,OUR_{\%_0} \end{aligned}$$

$$\vdots$$

$$\begin{aligned} k = N_p - 1: \quad & x_1 = CA^{N_p}x_0 + CA^{N_p-1}B\,\Delta u_0 + CA^{N_p-1}B_1\,OUR_{\%_0} + CA^{N_p-2}B\,\Delta u_1 \\ & + CA^{N_p-2}B_1\,OUR_{\%_1} + \ldots + CB\,\Delta u_1 + CB_1\,OUR_{\%_1} \end{aligned}$$

Dichas ecuaciones se pueden representar de manera matricial para lograr el modelo de predicción de la ecuación 1.2.

$$y_k = S_y\,x_{o_k} + M_y\,\Delta u_k + M_b\,OUR_{\%_k} \tag{1.2}$$

Donde:

$$y_k = [y_0 \quad y_1 \quad y_2 \quad \cdots \quad y_{N_p-1}]^T$$

$$\Delta u_k = \left[\Delta u_0 \quad \Delta u_1 \quad \Delta u_2 \quad \cdots \quad \Delta u_{N_p-1}\right]^T$$

$$OUR_{\%_k} = \begin{bmatrix} OUR_{\%_0} & OUR_{\%_1} & OUR_{\%_2} & \cdots & OUR_{\%_{N_p-1}} \end{bmatrix}^T$$

$$S_y = \begin{bmatrix} C\,A \\ C\,A^2 \\ \vdots \\ C\,A^{N_p-1} \end{bmatrix} \quad M_y = \begin{bmatrix} C\,B & 0 & \cdots & 0 \\ C\,A\,B & C\,B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\,A^{N_p-1}\,B & C\,A^{N_p-2}\,B & \cdots & C\,B \end{bmatrix} \quad M_b = \begin{bmatrix} C\,B_1 & 0 & \cdots & 0 \\ C\,A\,B_1 & C\,B_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\,A^{N_p-1}\,B_1 & C\,A^{N_p-2}\,B_1 & \cdots & C\,B_1 \end{bmatrix}$$

El modelo de predicción de la ecuación 1.2 permite obtener el comportamiento futuro del sistema utilizando datos de pre visualización del consumo (OUR), con el fin de mejorar el rendimiento del MPC debido a la anticipación de la respuesta del control a las perturbaciones, logrando una mayor estabilidad del sistema. Esta técnica requiere mayor costo computacional dado al estimador de biomasa que se necesita, sin embargo, el modelo que se usa en este trabajo asume que solo es posible medir el dato en tiempo actual más no futuro. Por tanto, se asume que las perturbaciones no se encuentran presentes en los instantes futuros de tiempo. Teniendo en cuenta lo anterior, se transforma la matriz M_b en un vector escalar, definiendo nuevamente dicha matriz como se muestra en la ecuación 1.3.

$$M_b = \begin{bmatrix} C\,B_1 \\ C\,A\,B_1 \\ \vdots \\ C\,A^{N_p-1}\,B_1 \end{bmatrix} \tag{1.3}$$

Para trasformar la función de costo de la ecuación 1.4 primero es necesario armar las matrices $\mathcal{Q}$ y $\mathcal{R}$ a partir de las predicciones del error de seguimiento y del esfuerzo de control, para luego pasar a forma matricial la función de costo como se muestra en la ecuación 1.5.

$$J = Q\left\|y_{k+i} - \widetilde{COD}_{\%_{k+i}}\right\|_2^{\,2} + R\|\Delta u_{k+i}\|_2^{\,2} \tag{1.4}$$

$$r_k = \begin{bmatrix} \widetilde{COD}_{\%_0} & \widetilde{COD}_{\%_1} & \widetilde{COD}_{\%_2} & \cdots & \widetilde{COD}_{\%_{N_p-1}} \end{bmatrix}^T$$

$$J = \begin{bmatrix} y_0 - r_o & \cdots & y_{N_p-1} - r_{N_p-1} \end{bmatrix} \underbrace{\begin{bmatrix} Q & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q \end{bmatrix}}_{\mathcal{Q}} \begin{bmatrix} y_0 - r_o \\ \vdots \\ y_{N_p-1} - r_{N_p-1} \end{bmatrix} + \begin{bmatrix} \Delta u_0 & \cdots & \Delta u_{N_p-1} \end{bmatrix} \underbrace{\begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix}}_{\mathcal{R}} \begin{bmatrix} \Delta u_0 \\ \vdots \\ \Delta u_{N_p-1} \end{bmatrix}$$

$$J = (y_k - r_k)^T\,\mathcal{Q}\,(y_k - r_k) + \Delta u_k{}^T\,\mathcal{R}\,\Delta u_k \tag{1.5}$$

Teniendo la función de costo en forma matricial se remplaza y_k por el modelo de predicción de la ecuación 1.2, para que el nuevo problema de optimización sea función únicamente del esfuerzo de control.

$$J = \left(S_y\,x_{o_k} + M_y\,\Delta u_k + M_b\,OUR_{\%_k} - r_k\right)^T \mathcal{Q}\left(S_y\,x_{o_k} + M_y\,\Delta u_k + M_b\,OUR_{\%_k} - r_k\right) + \Delta u_k{}^T\,\mathcal{R}\,\Delta u_k$$

Al expandir la función de costo anterior se obtiene la función de la ecuación 1.6.

$$J = \Delta u_k{}^T\left(\mathcal{R} + M_y{}^T\mathcal{Q}\,M_y\right)\Delta u_k + 2\left(S_y\,x_{o_k} + M_b\,OUR_{\%_k} - r_k\right)^T\mathcal{Q}\,M_y\,\Delta u_k \tag{1.6}$$

Ya teniendo la función de costo en términos del esfuerzo de control, las restricciones del problema de optimización igualmente se tienen que transformar. Para ello, las dos restricciones se unen en una sola como se muestra en la ecuación 1.7, ya que las dos son dependientes entre sí.

$$\left.\begin{array}{c} \underbrace{\begin{bmatrix}200\\0.2\end{bmatrix}}_{u_{min}} \leq \begin{bmatrix}\omega_k\\ \mathcal{F}_{in_k}\end{bmatrix} \leq \underbrace{\begin{bmatrix}950\\2\end{bmatrix}}_{u_{max}} \\ \Delta u_k = u_k - u_{k-1} \end{array}\right\} Restricciones$$

$$\underbrace{\begin{bmatrix}200-u_{k-1}\\0.2-u_{k-1}\end{bmatrix}}_{u_{min}-u_{k-1}} \leq \underbrace{\begin{bmatrix}\omega_k-u_{k-1}\\ \mathcal{F}_{in_k}-u_{k-1}\end{bmatrix}}_{\Delta u_k} \leq \underbrace{\begin{bmatrix}950-u_{k-1}\\2-u_{k-1}\end{bmatrix}}_{u_{max}-u_{k-1}} \quad \therefore \quad \Delta u_k \leq \begin{bmatrix}u_{max}-u_{k-1}\\-u_{min}+u_{k-1}\end{bmatrix} \tag{1.7}$$

Dado que el problema de optimización resultante de la función de costo de la ecuación 1.6 es cuadrática las restricciones se deben escribir de la siguiente manera:

$$\underbrace{\begin{bmatrix}1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 1\\ -1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & -1\end{bmatrix}}_{M_u} \underbrace{\begin{bmatrix}\Delta u_0\\ \Delta u_1\\ \vdots\\ \Delta u_{N_p-1}\end{bmatrix}}_{\Delta u_k} \leq \underbrace{\begin{bmatrix}u_{max}\\ u_{max}-u_o\\ \vdots\\ u_{max}-u_{N_p-2}\\ -u_{min}\\ -u_{min}+u_o\\ \vdots\\ -u_{min}+u_{N_p-2}\end{bmatrix}}_{U_{lim}}$$

C.2. Código MPC.

A continuación se muestra el código empleado en Matlab® para seguir el Algoritmo 1. Los parámetros de sintonización se muestran en *ControlMPC.m:*

```
global Qbar Mu Rbar N DeltaT Funcion vec_agi inter
st=0.0028; %0.0028 en horas, 0.167 en minutos (10 seg)
N=30; %Horizonte de prediccion
DeltaT=12/3600;  %Tiempo discreto(h) 15s
Q=0.3;
R=[0.1 0;0 100];
Qbar=[];
Rbar=[];
Mu=[eye(2*N);-eye(2*N)];

for i=0:N-1
    Qbar=blkdiag(Qbar,Q);
    Rbar=blkdiag(Rbar,R);
end
```

Función de costo: *MPCobj.m*

```
function y = MPCobj(u)
tic;
global Qbar Mu N Rbar
Cod=u(1); %Concentracion de oxigeno disuelto
Fin=u(3); %Flujo de aire
w=u(2); %Agitacion
yref=u(5); %Referencia
Our=u(4); %Perturbacion
```

```
[Ad,Bd,Cd,Bdo]=modelo(u); %Modelo

for i=0:N-1
    sy(i+1,:)=Cd*Ad^(i+1);
    so(i+1,:)=Cd*Ad^(i)*Bdo;
    FilaMy=zeros(size(Cd,1),size(Bd,2)*N); %m=size(Cd,1) r=size(Bd,2)
    aux=1;
    for m=0:i
       FilaMy(:,aux:aux+1)=[Cd*Ad^(i-m)*Bd];
       aux=aux+2;
    end
     My(i+1,:)=FilaMy;
end

H=Rbar+(My'*Qbar*My);
x0=[Cod;w;Fin]; %Estado aumentados
F=(((sy*x0)-yref)+(so*Our))'*Qbar*My;

%Limitaciones del controlador
Umin=[200;0.2];
Umax=[950;2];
DUmin=[-0.1;-0.1];
DUmax=[200;1];
aux2=1;
for i=1:N
   b(aux2:aux2+1,1)=Umax-x0(2:3);
   b(aux2+(2*N):aux2+1+(2*N),1)=-Umin+x0(2:3);
   a(aux2:aux2+1,1)=DUmax;
   a(aux2+(2*N):aux2+1+(2*N),1)=-DUmin;
   aux2=aux2+2;
end
Delta=Mu; %[Mu;Mu];
Res=b; %[b;a];

%Optimizador
  [Y,fval]=quadprog(2*H,2*F,Delta,Res);
  y=[Y(1)+w Y(2)+Fin];
  tiempo = toc;
```

Linealización sucesiva: *Modelo*

```
function [Ad,Bd,Cd,Bdo]=modelo(u)

global DeltaT Funcion
Cod=u(1); %Concentracion de oxigeno disuelto
Fin=u(3); %Flujo de aire
w=(2); %Agitacion
MF=[0.2 0.5 1 1.5 2];
[mf,pf]=min(abs(MF-Fin));

%Modelo linealizado
dc=-((w*Funcion(1,pf))+Funcion(2,pf));
dw=Funcion(1,pf)*(100-Cod);
do=-1;
%Discretizar
[Ak,Bk,Ck,Dk]=c2dm(dc,[dw do],1,zeros(1,2),DeltaT);
%Sistema aumentado
Ad=[Ak Bk(1) 0;zeros(2,1) eye(2)];
Bd=[Bk(1) 0;eye(2)];
Bdo=[Bk(2);zeros(2,1)];
Cd=[Ck zeros(1,2)];
```