

Modélisation et développement d'un système aquaponique avec surveillance métrologique pour l'étude du cycle de l'azote

Auteur : Stalport, Benoît

Promoteur(s) : 879

Faculté : Gembloux Agro-Bio Tech (GxABT)

Diplôme : Master en bioingénieur : sciences et technologies de l'environnement, à finalité spécialisée

Année académique : 2016-2017

URI/URL : <http://hdl.handle.net/2268.2/3009>

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.

LIVRET ANNEXE - CODE MATLAB

**« MODÉLISATION ET DÉVELOPPEMENT D'UN SYSTÈME AQUAPONIQUE AVEC
SURVEILLANCE MÉTROLOGIQUE POUR L'ÉTUDE DU CYCLE DE L'AZOTE »**

BENOÎT STALPORT

**TRAVAIL DE FIN D'ÉTUDES PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MASTER BIOINGÉNIEUR EN SCIENCES ET TECHNOLOGIES DE L'ENVIRONNEMENT**

ANNÉE ACADÉMIQUE 2016-2017

PROMOTEUR: PR. FRÉDÉRIC LEBEAU

© Toute reproduction du présent document par quelque procédé que ce soit ne peut être réalisée qu'avec l'autorisation de l'auteur et du Doyen de Gembloux Agro-Bio Tech.

Le présent document n'engage que son auteur.

« La meilleure façon de prédire l'avenir, c'est de le créer »

Peter F. Drucker

TABLE DES MATIÈRES

TABLE DES MATIÈRES	4
1 INTRODUCTION.....	5
2 SCRIPTS MATLAB	6
2.1 EXÉCUTION DU MODÈLE	6
2.2 PARAMÈTRES DU MODÈLE	6
2.3 DONNÉES IMPORTÉES	9
2.4 RÉOLUTION	12
2.5 FONCTIONS	16
2.5.1 Croissance des poissons en taille « <i>length_fish_growth.m</i> »	16
2.5.2 Croissance des poissons en poids « <i>length_fish_growth.m</i> »	16
2.5.3 Evolution des degrés-jours « <i>f_dtt.m</i> »	16
2.5.4 Evolution du LAI relatif « <i>f_lairel.m</i> »	16
2.5.5 Evolution du LAI « <i>f_lai.m</i> »	16
2.5.6 Evolution de la biomasse sans facteur lumineux « <i>plant_growth_nosrad.m</i> »	17
2.5.7 Evolution de l'évapotranspiration « <i>evapotranspiration.m</i> »	17
2.5.8 Fonctionnement de la pompe entre l'aquarium et le système hydroponique « <i>f_pump3a.m</i> »	17
2.5.9 Fonctionnement de la pompe de remplissage de l'aquarium « <i>f_pump2a.m</i> »	18
2.5.10 Production de nitrates « <i>nitrates.m</i> »	18
2.5.11 Consommation de nitrates « <i>nitrates_consumption.m</i> »	18
2.5.12 Fonctions limitant la croissance des poissons	19
2.5.12.1 Température de l'eau « <i>temperature_factor.m</i> »	19
2.5.12.2 pH de l'eau « <i>f_fish_pH.m</i> »	19
2.5.12.3 Concentration en oxygène dissous « <i>DO_factor.m</i> »	19
2.5.12.4 Concentration en ammoniac « <i>UAm_factor.m</i> »	20
2.5.12.5 Concentration en nitrates « <i>f_NO3_fish.m</i> »	20
2.5.12.6 Nourriture « <i>food_factor.m</i> »	20
2.5.13 Fonctions limitant la croissance des laitues	20
2.5.13.1 Température de la solution « <i>f_temp_solution.m</i> »	20
2.5.13.2 pH de la solution « <i>f_pH.m</i> »	21
2.5.13.3 Concentration en oxygène dissous « <i>f_DO_lettuces.m</i> »	21
2.5.13.4 Concentration en nitrates « <i>f_NO3_concentration.m</i> »	21
2.5.13.5 Humidité relative de l'air « <i>f_HR.m</i> »	21
2.6 GRAPHIQUES	22

1 INTRODUCTION

Le livret présenté ici fournit le code *Matlab* développé dans le cadre du travail de fin d'étude de Benoît Stalport, « *Modélisation et développement d'un système aquaponique avec surveillance métrologique pour l'étude du cycle de l'azote* ».

Le code est organisé en un ensemble de fonctions. Chacune d'entre-elles est utilisée uniquement lorsqu'elle est appelée au sein d'un autre script. Le document présenté ici commence par fournir les scripts généraux et évolue vers les fonctions plus spécialisées.

De plus, le modèle a été développé en anglais. Afin de rester cohérent avec la rédaction du travail de fin d'étude, les annotations des graphiques sont toutefois formulées en français.

2 SCRIPTS MATLAB

2.1 EXÉCUTION DU MODÈLE

Le script « *run_model.m* » est responsable du fonctionnement global du modèle. C'est lui qu'il faut exécuter afin de lancer le modèle.

```
%% Initialization %%

clear; clc; close all;
clearvars; clear all;

%% Data %%

run data.m
run external_data.m
run data_analysis_seneye.m
run data_analysis_climatisation.m
run data_analysis_manual.m
run data_analysis_irrigation.m

%% RESOLUTION %%

run resolution.m

%% Graphs %%

run graphs.m
```

2.2 PARAMÈTRES DU MODÈLE

Le script « *data.m* » contient l'ensemble des paramètres du modèle mais n'effectue aucune opération sur ces dernières.

```
%% Constants %%

global epsilon Tc kmin T Tmin Tmax HRopt HRmin HRmax HR counter_waste ECopt global ECmin EC Topt1 Topt2 Kw
evap_buffer pump_flow_rate counter3 counter3a global evap hydroponic_EC j DO DOmin DOopt food_percentage A
Acrit Amax global fishnbr LAImax OUTcoef INcoef K Linf
global taul tau2 taud pHopt gamma DO_min lettuces DO_lettuces DO_opt_lettuces global nitrogen_food_content ETci
food_qty eloss n_plant RUE ni Winf pH Vaq0 global pHmin NO3newwater pHmax pHopt_1 n_lettuces pHopt_2 NO3opt
NO3min global food_quantity Cext regulation Cco2 Tpopt Tpbase Tpmin Tpmx TTM c1 c2
global temp temp_min temp_opt_1 temp_opt_2 temp_max temp_solution dt
global conv_factor tf pH_fish temp_solution_min temp_solution_opt_1
global temp_solution_opt_2 temp_solution_max pH_fish_min pH_fish_max
global pH_fish_opt1 pH_fish_opt2 NO3_fish_ideal NO3_fish_max

%% Time Parameters

t0=0;
tf=1200;
dt = 1;

%% Fish tank %%

counter3a=0;
```

```

nitrogen_food_content = 0.0569;
food_qty = 4;

% von Bertalanffy growth equation %

% Length

K = 0.651;
Linf = 374.26;

% pH

pH_fish_min = 6.5;
pH_fish_max = 8;
pH_fish_opt1 = 7.2;
pH_fish_opt2 = 7.6;
pH_fish = 7.5;

% NO3

NO3_fish_ideal = 60;
NO3_fish_max = 3000;

T = 19; % [°C] water temperature
Tmin = 0; % [°C] min water temperature
Tmax = 30; % [°C] max water temperature
Topt1 = 18; % [°C] opt water temperature
Topt2 = 22; % [°C] opt water temperature

DO = 12; % [ppm] measured dissolved oxygen
DOmin = 4; % [ppm] minimum dissolved oxygen
DOopt = 8; % [ppm] critical dissolved oxygen

A = 0.1; % [ppm] measured ammonia
Acrit = 0.2; % [ppm] critical ammonia
Amax = 4; % [ppm] deadly ammonia

food_percentage = 0.02; % [-] portion of the fish weight fed

% Fish tank

Vaq0 = 200; % [l] water volume of the fish tank
fishnbr = 19; % [fishes] number of fish

length_fish0 = 119.72; % [mm] initial average length of one goldfish
fish0 = 1.168*(10^-6)*length_fish0^3.5077; % [g] initial weight of one goldfish

length_fish10=85;
fish10 = 1.168*(10^-6)*length_fish10^3.5077; % [g] initial weight of one goldfish

length_fish20=115;
fish20 = 1.168*(10^-6)*length_fish20^3.5077; % [g] initial weight of one goldfish

length_fish30=140;
fish30 = 1.168*(10^-6)*length_fish30^3.5077; % [g] initial weight of one goldfish

% Nitrates production

eloss = 0.05; % [-] portion of feed lost into the water

NO3aqp0 = 0.0015; % [ppm/hrs] initial nitrate production
NO3aq0 = 55; % [ppm] initial nitrate concentration in fish tank
x0 = 0.005; % State variable (accumulation in the digestive system)

%% Buffer Tank %%

Vtamp0 = 0;
NO3buff0 = 0;

%% Hydroponic solution %%

Vhyd0 = 130;
NO3hyd0 = 850;

%% Lettuces %%

% Dissolved Oxygen

```



```

DO_min_lettuces = 4; % [ppm]
DO_opt_lettuces = 7; % [ppm]
DO_lettuces = 8; % [ppm]

n_lettuces = 15*6; % [-] number of lettuces

Ppop = 12/0.55; % plant/m²
Rowspc = 0.2; % m
Cext = max(0, 1.5-0.768*(Ppop*Rowspc^2)^0.1);
Cco2 = 200E-6;

OUTcoef = 0.9 ;
INcoef = 1 - OUTcoef;
OUT0 = 0;
IN0 = 0;

LAI0 = 0;
LAIrel0 = 0;
Biomass0 = 0;

LAImax = 4;

% Light --- modalities

n_plant = 96;
n_exp = 18;
RUE = 2.5; % g/MJ
conv_factor = 4;

% pH

pH = 6;
pHmin = 3;
pHmax = 11;
pHopt = 6;
pHopt_1 = 6;
pHopt_2 = 8;

% Nitrates

NO3opt = 700;
NO3min = 100;

% EC

ECopt = 2000;
ECmin = 300;
EC = 2000;

% HR

HR = 85;
HRmin = 10;
HRmax = 100;
HRopt = 85;

% air temperatures

temp = 19.5;
temp_min = 6;
temp_opt_1 = 16;
temp_opt_2 = 25;
temp_max = 33;

% solution temperatures

temp_solution = 20;
temp_solution_min = 15;
temp_solution_opt_1 = 18;
temp_solution_opt_2 = 23;
temp_solution_max = 30;

%

c1 = 6;
c2 = 7;

Tpopt = 24;
Tpbase = 3.5;

```

```
Tpmin = 19;
Tpmax = 20.5;
TTC0 = 0;
TTrel0 = 0;
TTm = 800; % Total thermal time needed

%% Others %%

pump_flow_rate = 3; % [l/hrs]
ETci = 70;
NO3newwater = 14;
Tot_t0 = 1;
ni = 1;
counter3 = 0;
counter_waste=0;
```

2.3 DONNÉES IMPORTÉES

Les scripts de la forme « *data_analysis_...* » ont été générés par l'outil *import data* de *Matlab* et servent à importer les données de fichiers générés lors des mesures. Ceux-ci sont constitués de nombreuses lignes inintéressantes et ne sont pas présentés ici. Le script « *external_data.m* », quant à lui, rassemble toutes les données non produites par le modèle et non importées d'autres fichier.

[illegible]

```

16000 4000 6000 5000 7000 6000 6000 6000 1500 3100 5000 5230 3500 3000];

%% Evapotranspiration %%

Evapotranspiration_times = [0 43 115 144 162 354 426.5 451.5 472.17 544.17 574 595 618 648 699.83 739 763.5
816.75 844 859 883.83 906.5 931 953.5 979.58 1003.17 1014.67 1027 1051 1075 1098.5 1120];
Measured_evapotranspiration = [NaN 0.73 0.43 1.44 NaN NaN 0.06 0.72 0.79 0.81 0.72 1.22 1.30 1.46 1.47 1.90
1.79 1.04 6.52 2.96 2.68 2.45 3.17 3.70 3.19 4.71 2.42 6.98 5.79 12.11 4.96 4.65];
Evapotranspiration_vector = [0.42 0.42 0.42 0.42 0.42 0.42 0.42 0.42 0.46 0.50 0.68 0.77 0.85 0.93 1.06 1.31 1.55
1.72 2.15 2.41 2.56 2.85 3.13 3.47 3.81 4.25 4.70 4.93 5.19 5.74 6.35 7.01 7.67];

%% Nitrates %%

NO3_time = [240 264 285 310 367 406 429 454 475 499 523 547 578 598 621 651 718 742 767 796 820 871 886 911 934
958 980 1059 1078 1102 1126 1147];
Fish_tank_NO3 = [104.70 111.80 119 122.50 92.50 120.20 127.40 134.40 140.20 140.20 140.20 140.20 155.40 159.40
161.30 165.40 172.90 174.65 176.40 175.20 174.00 182.00 167.60 165.40 170.90 178.80 146.14 163.41 168.29 176.26
176.70 184.67];
Hydroponic_NO3 = [820 841 818 806 696 686 783 801 835 835 835 835 891 871 835 837 813 741 711 652 580 509.85
422.35 385 565 538 512.57 487.14 473.86 447.51 421.16 407.87];

NO3_time_2 = [240 547 911 1059];
Fish_tank_NO3_2 = [104.7 140.20 165.40 163.4];
Hydroponic_NO3_2 = [820 835 837 487.14];

%% Fish food %%

Food_vector=zeros(1,tf+1);

Feeding_times = [1 8 46 54 70 75 95 102 120 128 142 150 173 214 221 238 245 263 267 287 294 313 315 338 358 370
382 409 432 437 457 460 478 504 527 552 558 580 584 601 606 624 631 653 671 706 725 749 774 799 822 842 872 888
892 910 918 934 941 959 982 1013 1032 1056 1061 1079 1085 1101 1110 1127 1134 1149 1157];
Feeding_quantities = [4.45 4.01 3.99 3.97 4.01 3.97 4.04 4.02 4.02 4.04 3.99 4.05 4.04 4 3.99 3.99 4.01 3.99
4.01 4.03 4.04 3.98 4.07 4.7 4.11 3.94 4.07 4.05 4.04 4.07 4.03 4.05 4 4.55 4.15 4 4.04 4.07 4.08 4.04 4.04
4.03 4.04 4.08 4.06 4.05 4.05 4.03 4.03 4.02 4.05 4 4 6.03 6.02 5.9 5.95 6.2 5.99 6 6.03 5.97 5.95 4.96 5 5.12
5 4.95 4.97 5.05 4.96 4.93 4.98];

food_count = 1;

for loop = 1:(tf-1)
    if loop == Feeding_times(food_count)
        Food_vector(loop) = Feeding_quantities(food_count);
        if food_count < length(Feeding_times)
            food_count = food_count+1;
        else
            food_count = food_count;
        end
    else
        Food_vector(loop) = 0;
        food_count = food_count;
    end
end

%% Fish %%

Weighting_time = [144 432 504 600 744 840 936 1008 1104 1176];

fish_1 = [10.17 10.7 11.6 11.93 11.25 11.1 11.76 12.11 12.49 12.51];
fish_2 = [22.3 23.43 25.83 24.65 23.25 24.22 23.67 24.71 26.83 26.93];
fish_3 = [40.67 45.24 47.79 48.55 40.8 46.69 52.41 52.9 54.36 54.4];

fish_1_length = [NaN NaN NaN NaN 84.3 93.1 NaN NaN NaN 105];
fish_2_length = [NaN 122.7 118.7 131.9 126.1 126.2 NaN NaN NaN 135];
fish_3_length = [138 150.2 158.3 150.1 148.1 152.7 NaN NaN NaN 155];

%% Fertilizer %%

fertilizer_time =[216 406 911];
fertilizer_quantities = [115 150 360];

clock1 = 1;
clock2 = 1;

for i = 1:tf+1
    if clock1 == fertilizer_time(clock2)
        fertilizer_vector(i)= fertilizer_quantities(clock2);
        if clock2 == length(fertilizer_time)
            else
                clock2 = clock2+1;
            end
        else
            else

```

```

        fertilizer_vector(i)=0;
    end
    clock1 = clock1 + 1;
end

%% External water exchanges %%

external_exchanges_time = [168 367];
external_exchanges_quantities = [40 50];

clock1 = 1;
clock2 = 1;

for i = 1:tf+1
    if clock1 == external_exchanges_time(clock2)
        external_exchanges_vector(i)= external_exchanges_quantities(clock2);
        if clock2 == length(external_exchanges_time)
            else
                clock2 = clock2+1;
            end
        else
            external_exchanges_vector(i)=0;
        end
        clock1 = clock1 + 1;
    end

    %% Overflowing water %%

    overflowing_water_time = [168 367];
    overflowing_water_quantities = [40 50];

    clock1 = 1;
    clock2 = 1;

    for i = 1:tf+1
        if clock1 == overflowing_water_time(clock2)
            overflowing_water_vector(i)= overflowing_water_quantities(clock2);
            if clock2 == length(overflowing_water_time)
                else
                    clock2 = clock2+1;
                end
            else
                overflowing_water_vector(i)=0;
            end
            clock1 = clock1 + 1;
        end

        %% Lettuces %%

        lettuce_number = 1:96;

        harvest_times = [936 984 1080 1152 1152 936 984 1080 1152 720 936 984 1080 1080 1152 1152 936 984 1080 1152
        1152 936 984 1080 1152 408 936 984 1080 1080 1152 1152 936 984 1080 1152 1152 936 984 1080 1152 576 936 984
        1080 1080 1152 1152 936 984 1080 1152 1152 936 984 1080 1152 816 936 984 1080 1080 1152 1152 936 984 1080 1152
        1152 936 984 1080 1152 480 936 984 1080 1080 1152 1152 936 984 1080 1152 1152 936 984 1080 1152 576 936 984
        1080 1080 1152 1152];

        lettuce_weight = [2.31 6.53 7.99 11.64 8.42 3.97 7.38 9.55 12.63 1.41 1.41 3.87 5.46 3.99 4.79 3.46 1.86 4.65
        6.86 7.82 8.45 3.17 4.92 5.81 8.56 0.043 0.97 2.04 3.3 2.69 3.56 2.52 2 4.43 4.97 6.97 6.5 4.78 8.16 8.16 11.64
        0.404 2.11 2.72 4.24 3.82 4.32 3.5 0.92 3.57 4.16 4.63 4.08 3.2 5.11 7.02 6.91 1.984 0.67 2.03 2.32 2.39 2.51
        2.9 1.1 3.48 4.84 4.57 3.36 2.82 4.37 5.08 4.34 0.17 1.44 3.45 3.06 4.29 4.83 2.68 1.824 3.58 5.57 4.75 4.65
        3.19 4.04 5.3 5.27 0.404 2.58 4.05 4.41 4.55 5.28 3.27];

```

2.4 RÉSOLUTION

Le script « *resolution.m* » est responsable de la résolution des équations du modèle par différences finies, du calcul du facteur de dimensionnement et des calculs des facteurs de qualité des prédictions du modèle.

```
global Real_t0 food_2 OUTcoef food NO3newwater dt EC EC_2 food_qty n_exp global fertilizer_vector n_plant
conv_factor Food_vector tf

%% Initialization

p15=zeros(tf,n_exp);
p16=zeros(tf,n_exp);
p17=zeros(tf,n_exp);
p13=zeros(tf,n_exp);
biomass=zeros(tf,n_plant);
food=zeros(1,tf+1);

%% Time

time_r = zeros(1,((tf+1)/dt));
Real_t0 = zeros(1,((tf+1)/dt));
Real_t0(1)=0;
time_r(1)=0;
food_2(1)=0;
food(1) = Food_vector(1);
t(1) = 0;

%% Initialization

jj=1;

p1(1)=Tot_t0;
p2(1)=length_fish0;
p3(1)=fish0;
p21(1)=length_fish10;
p31(1)=fish10;
p22(1)=length_fish20;
p32(1)=fish20;
p23(1)=length_fish30;
p33(1)=fish30;
p4(1)=x0;
p5(1)=NO3aq0;
p6(1)=NO3buff0;
p7(1)=Vaq0;
p8(1)=Vtamp0;
p9(1)=Vhyd0;
p10(1)=TTC0;
p11(1)=TTrel0;
p12(1)=LAIrel0;
p13(1)=(NO3hyd0*Vhyd0);
p14(1)=LAI0;
p15(1)=Biomass0;
p16(1)=OUT0;
p17(1)=IN0;
p18(1)=NO3hyd0;
p19(1)=ETci;

for iter = 1:n_plant
    biomass(1,iter) = 0;
end

biomass_total(i) = 0;

%% Differentials

for i = 2:1:((tf+1)/dt)

    % Measurements vectors

if Food_vector(i) == 0
```

```

    food(i)=food(i-1);
else
    food(i)=Food_vector(i);
end

% Differentials

dp1(i) = 1;
dp2(i) = length_fish_growth(p2(i-1),p3(i-1),i,p5(i-1));
dp3(i) = fish_weight_growth(p2(i-1),p3(i-1));
dp21(i) = length_fish_growth(p21(i-1),p3(i-1),i,p5(i-1));
dp31(i) = fish_weight_growth(p21(i-1),p31(i-1));
dp22(i) = length_fish_growth(p22(i-1),p3(i-1),i,p5(i-1));
dp32(i) = fish_weight_growth(p22(i-1),p32(i-1));
dp23(i) = length_fish_growth(p23(i-1),p3(i-1),i,p5(i-1));
fish_weight_growth(p23(i-1),p33(i-1));
dp4(i) = nitrates((fish_growth(p3(i-1),i,t(i))),i);
dp5(i) = ((dp4(i)+(NO3newwater*f_pump2a(p7(i-1)))-(p5(i-1)*f_pump3a(p9(i-1)))-
    (external_exchanges_vector(i)*p5(i-1)))/(p7(i-1)));
dp6(i) = 0;
dp7(i) = (f_pump2a(p7(i-1)) - f_pump3a(p9(i-1))-external_exchanges_vector(i));
dp8(i) = 0;
dp9(i) = (0 - (p19(i-1)/1000) + f_pump3a(p9(i-1))+external_exchanges_vector(i)-overflowing_water_vector(i));
dp10(i) = f_DTT(Tpmin,Tpmax);
dp11(i) = ((f_DTT(Tpmin,Tpmax))/TTm);
dp12(i) = f_LAIrel(p12(i-1),p11(i-1));
dp14(i) = f_LAI(f_LAIrel(p12(i-1),p11(i-1)), p18(i-1),import_pH_vector(i));

d_biomass_noSrad(i) = plant_growth_noSrad(p14(i-1),p18(i-1),import_pH_vector(i),import_CO2_vector(i))*dt;

for iter = 1:n_plant

    if i <= 408
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR0(iter)*conv_factor*n_lettuce0(1,iter);
    elseif i <= 480
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR0(iter)*conv_factor*n_lettuce0(2,iter);
    elseif i <= 576
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR0(iter)*conv_factor*n_lettuce0(3,iter);
    elseif i <= 720
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR0(iter)*conv_factor*n_lettuce0(4,iter);
    elseif i <= 816
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR0(iter)*conv_factor*n_lettuce0(5,iter);
    elseif i <= 936
        if i <= 912
            d_biomass(i,iter) = d_biomass_noSrad(i)*PAR0(iter)*conv_factor*n_lettuce0(6,iter);
        else
            d_biomass(i,iter) = d_biomass_noSrad(i)*PAR912(iter)*conv_factor*n_lettuce0(6,iter);
        end
    elseif i <= 984
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR912(iter)*conv_factor*n_lettuce0(7,iter);
    elseif i <= 1080
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR912(iter)*conv_factor*n_lettuce0(8,iter);
    elseif i <= 1152
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR912(iter)*conv_factor*n_lettuce0(9,iter);
    else
        d_biomass(i,iter) = d_biomass_noSrad(i)*PAR912(iter)*conv_factor*n_lettuce0(10,iter);
    end

end

dp13(i) = -nitrates_consumption(sum(d_biomass(i,1:n_plant)));
dp18(i) = ((dp13(i)-(overflowing_water_vector(i)*p18(i-1)))+(external_exchanges_vector(i)*p5(i-1))+(p5(i-1)*f_pump3a(p9(i-1))))/p9(i-1))+fertilizer_vector(i);
dp19(i) = evapotranspiration(i,p19(i-1));

% New values

p1(i) = p1(i-1) + dp1(i)*dt;
p2(i) = p2(i-1) + dp2(i)*dt;
p3(i) = p3(i-1) + dp3(i)*dt;
p21(i) = p21(i-1) + dp21(i)*dt;
p31(i) = p31(i-1) + dp31(i)*dt;
p22(i) = p22(i-1) + dp22(i)*dt;
p32(i) = p32(i-1) + dp32(i)*dt;
p23(i) = p23(i-1) + dp23(i)*dt;
p33(i) = p33(i-1) + dp33(i)*dt;
p4(i) = p4(i-1) + dp4(i)*dt;
p5(i) = p5(i-1) + dp5(i)*dt;
p6(i) = p6(i-1) + dp6(i)*dt;
p7(i) = p7(i-1) + dp7(i)*dt;

```

```

p8(i) = p8(i-1) + dp8(i)*dt;
p9(i) = p9(i-1) + dp9(i)*dt;
p10(i) = p10(i-1) + dp10(i)*dt;
p11(i) = p11(i-1) + dp11(i)*dt;
p12(i) = p12(i-1) + dp12(i)*dt;
p13(i) = p13(i-1) + dp13(i)*dt;
p14(i) = p14(i-1) + dp14(i)*dt;
p18(i) = p18(i-1) + dp18(i)*dt;
p19(i) = p19(i-1) + dp19(i)*dt;

for iter = 1:n_plant
    if i <= 408
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(1,iter);
    elseif i <= 480
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(2,iter);
    elseif i <= 576
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(3,iter);
    elseif i <= 720
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(4,iter);
    elseif i <= 816
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(5,iter);
    elseif i <= 936
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(6,iter);
    elseif i <= 984
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(7,iter);
    elseif i <= 1080
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(8,iter);
    elseif i <= 1152
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(9,iter);
    else
        biomass(i,iter) = biomass(i-1,iter) + d_biomass(i,iter)*dt;
        biomass(i,iter) = biomass(i,iter)*n_lettuce0(10,iter);
    end

biomass_tot(i) = sum(biomass(i,:));

end

biomass_total(i) = sum(biomass(i,1:n_plant));

end

%% Additionnal calculations

% Creation of calculated lettuces weight vector to compare with real data

lettuce_calculated_weight = zeros(1,n_plant);

xyx = 0:1:1000;
xyy = xyx;

for i =1:96
    lettuce_calculated_weight(i) = OUTcoef*biomass(harvest_times(i),i);
end

% Linear regression and R²

b1 = lettuce_calculated_weight/lettuce_weight;
yCalcl = b1*lettuce_weight;
R2_lettuces = 1 - sum((lettuce_calculated_weight - yCalcl).^2)/sum((lettuce_calculated_weight - mean(lettuce_calculated_weight)).^2)
RMSE_lettuces = sqrt((sum((lettuce_calculated_weight-lettuce_weight).^2))/length(lettuce_weight))

% Creation of calculated fish weight and measured fish weight vector

Fish_calculated_weight = zeros(1,30);

Fish_weighting_times = [144 432 504 600 744 840 936 1008 1104 1176 144 432 504 600 744 840 936 1008 1104 1176
144 432 504 600 744 840 936 1008 1104 1176];

```

```

Fish_weights = [10.17 10.7 11.6 11.93 11.25 11.1 11.76 12.11 12.49 12.51 22.3 23.43 25.83 24.65 23.25 24.22
23.67 24.71 26.83 26.93 40.67 45.24 47.79 48.55 40.8 46.69 52.41 52.9 54.36 54.4];

for i =1:30
    if i < 11
        Fish_calculated_weight(i) = p31(Fish_weighting_times(i));
    elseif i < 21
        Fish_calculated_weight(i) = p32(Fish_weighting_times(i));
    else
        Fish_calculated_weight(i) = p33(Fish_weighting_times(i));
    end
end

% Linear regression and R²

b2 = Fish_calculated_weight/Fish_weights;
yCalc2 = b2*Fish_weights;
R2_fishes = 1 - sum((Fish_calculated_weight - yCalc2).^2)/sum((Fish_calculated_weight -
mean(Fish_calculated_weight)).^2)
RMSE_fishes = sqrt((sum((Fish_calculated_weight-Fish_weights).^2))/length(Fish_weights))

% Creation of calculated nitrate concentration and measured nitrate concentration vector

Nitrate_hydroponic_calculated_concentration = zeros(1,32);
Nitrate_fish_tank_calculated_concentration = zeros(1,32);

Nitrate_measured_times = [240 264 285 310 367 406 429 454 475 499 523 547 578 598 621 651 718 742 767 796 820
871 886 911 934 958 980 1059 1078 1102 1126 1147];
Nitrate_hydroponic_measured = [820 841 818 806 696 686 783 801 835 835 835 891 871 835 837 813 741 711 652
580 509.85 422.35 385 565 538 512.57 487.14 473.86 447.51 421.16 407.87];
Nitrate_fish_tank_measured = [104.7 111.8 119 122.5 92.5 120.2 127.4 134.4 140.2 140.2 140.2 140.2 155.4 159.4
161.3 165.4 172.9 174.65 176.4 175.2 174 182 167.6 165.4 170.9 178.8 146.14 163.41 168.29 176.26 176.7 184.67];

for i =1:32
    Nitrate_predicted_hydroponic(i) = p18(Nitrate_measured_times(i));
    Nitrate_predicted_fish_tank(i) = p5(Nitrate_measured_times(i));
end

% Linear regression and R²

b3 = Nitrate_predicted_hydroponic/Nitrate_hydroponic_measured;
yCalc3 = b3*Nitrate_hydroponic_measured;
R2_nitrates_hydroponic = 1 - sum((Nitrate_predicted_hydroponic - yCalc3).^2)/sum((Nitrate_predicted_hydroponic
- mean(Nitrate_predicted_hydroponic)).^2)
RMSE_nitrate_hydroponic = sqrt((sum((Nitrate_predicted_hydroponic-
Nitrate_hydroponic_measured).^2))/length(Nitrate_hydroponic_measured))

b4 = Nitrate_predicted_fish_tank/Nitrate_fish_tank_measured;
yCalc4 = b4*Nitrate_fish_tank_measured;
R2_nitrates_fish_tank = 1 - sum((Nitrate_predicted_fish_tank - yCalc4).^2)/sum((Nitrate_predicted_fish_tank -
mean(Nitrate_predicted_fish_tank)).^2)
RMSE_nitrate_fish_tank = sqrt((sum((Nitrate_predicted_fish_tank-
Nitrate_fish_tank_measured).^2))/length(Nitrate_fish_tank_measured))

% Nitrates production/nitrates consumption ratio

ratio = zeros(1,tf/dt);
nit_prod = zeros(1,tf/dt);
nit_cons = zeros(1,tf/dt);
t_rat = zeros(1,tf/dt);
t_rat(1) = 1;

for rat = 1:1:(tf/dt)
    t_rat(rat+1) = t_rat(rat) +1;
    ratio(rat) = ((p4(rat+1))-p4(rat))/(-(p13(rat+1))-p13(rat)));
    nit_prod(rat+1) = nit_prod(rat)+((p4(rat+1))-p4(rat));
    nit_cons(rat+1) = nit_cons(rat)-((p13(rat+1))-p13(rat));
end

Oversizing = abs(sum(dp4)/sum(dp13))

```


2.5 FONCTIONS

Ce chapitre présente les différentes fonctions utilisées par le modèle.

2.5.1 CROISSANCE DES POISSONS EN TAILLE « LENGTH_FISH_GROWTH.M »

```
function length_growth = length_fish_growth(fish_length,individual_fish_weight,i,NO3_concentration)

global T A DO K Linf pH_fish

% This function predicts the additional average hourly growth of the fish

f_limitante = [temperature_factor(T) DO_factor(DO) UAm_factor(A) food_factor(individual_fish_weight,i)
f_NO3_fish(NO3_concentration) f_fish_pH(pH_fish)];
length_growth = ((Linf*(1-exp(-K*(i/(24*365))+1.8)))-fish_length)/(24*365))*min(f_limitante);

end
```

2.5.2 CROISSANCE DES POISSONS EN POIDS « LENGTH_FISH_GROWTH.M »

```
function weight_growth = fish_weight_growth(individual_fish_length,fish_weight)

% This function predicts the additional average hourly growth of the fish

weight_growth = 1.168*10^(-6)*individual_fish_length^(3.5077) - fish_weight;

end
```

2.5.3 EVOLUTION DES DEGRÉS-JOURS « F_DTT.M »

```
function DTT = f_DTT (Tmin,Tmax )

global Tpbase Tpopt

DTT = ((min(Tpopt,max(Tpbase,Tmin))+min(Tpopt,max(Tpbase,Tmax)))/2)-Tpbase)/(24);

end
```

2.5.4 EVOLUTION DU LAI RELATIF « F_LAIREL.M »

```
function LAIrel_increase = f_LAIrel(old_relative_LAI, TTrel)

global c1 c2

LAIrel_increase = ( TTrel / (TTrel + exp(c1 - (c2*TTrel)))) - old_relative_LAI ;

end
```

2.5.5 EVOLUTION DU LAI « F_LAI.M »

```
function LAI_variation = f_LAI(dLAIrel,NO3_hydroponic_concentration, pH)

global LAImax DO_lettuces HR temp temp_solution

min_matrix_LAI = [f_DO_lettuces(DO_lettuces) f_HR(HR) f_temp_solution(temp_solution) f_temp(temp) f_pH(pH)
f_NO3_concentration(NO3_hydroponic_concentration)];
LAI_variation = dLAIrel*LAImax*min(min_matrix_LAI);

end
```

2.5.6 EVOLUTION DE LA BIOMASSE SANS FACTEUR LUMINEUX « PLANT_GROWTH_NOSRAD.M »

```
function plant_weight_growth = plant_growth_noSrad(LAI,NO3_hydroponic_concentration,pH, Cco2)

global RUE Cext DO_lettuces HR temp_solution

min_matrix = [f_DO_lettuces(DO_lettuces) f_HR(HR) f_temp_solution(temp_solution) f_pH(pH)
f_NO3_concentration(NO3_hydroponic_concentration)];
plant_weight_growth = 0.5*RUE*(1 - exp(-Cext*LAI))*f_CO2(Cco2)*min(min_matrix)*0.0864/(24*(12/0.55));

end
```

2.5.7 EVOLUTION DE L'ÉVAPOTRANSPIRATION « EVAPOTRANSPIRATION.M »

```
function dETC = evapotranspiration(time,ETC)

if time <= 408
    n_lettuce = 96;
elseif time <= 480
    n_lettuce = 95;
elseif time <= 576
    n_lettuce = 93;
elseif time <= 720
    n_lettuce = 92;
elseif time <= 816
    n_lettuce = 91;
elseif time <= 936
    n_lettuce = 90;
elseif time <= 984
    n_lettuce = 72;
elseif time <= 1080
    n_lettuce = 54;
elseif time <= 1152
    n_lettuce = 30;
else
    n_lettuce = 0;
end

if time <= 427
    %dETC = (0.5/427)*n_lettuce;
    dETC = 0;
else
    ETC_uni = 0.0695*exp(0.0042*time);
    ETC_all = n_lettuce*ETC_uni+31;
    dETC = ETC_all-ETC;
end

end
```

2.5.8 FONCTIONNEMENT DE LA POMPE ENTRE L'AQUARIUM ET LE SYSTÈME HYDROPONIQUE « F_PUMP3A.M »

```
function pump3a_flow_rate = f_pump3a(hydroponic_level)

global counter3a pump_flow_rate

% Pump 3a regulation

if counter3a ==0
    if hydroponic_level <= 128
        pump3a_flow_rate = pump_flow_rate;
        counter3a =1;
    else
        pump3a_flow_rate = 0;
        counter3a =0;
    end
else
    if hydroponic_level < 130
        pump3a_flow_rate = pump_flow_rate;
        counter3a =1;
    end
end
```

```

else
    pump3a_flow_rate = 0;
    counter3a = 0;
end
end
end

```

2.5.9 FONCTIONNEMENT DE LA POMPE DE REMPLISSAGE DE L'AQUARIUM « F_PUMP2A.M »

```

function pump2a_flow_rate = f_pump2a(aquarium_volume)

global Vaq0 counter3a

% Pump 2a regulation
if counter3a ==1
    pump2a_flow_rate = 0;
else
    if aquarium_volume < 198
        pump2a_flow_rate = (Vaq0-aquarium_volume);
    else
        pump2a_flow_rate = 0;
    end
end
end

```

2.5.10 PRODUCTION DE NITRATES « NITRATES.M »

```

function nitrates_evolution = nitrates(fish_growth,i)

global dt nitrogen_food_content Food_vector

if i <= 2/dt
    a shift of two hours between
    food_eaten = 0;
    the nitrate production
else
    matlab error (Matrix dimensions must agree)
    food_eaten = Food_vector(i-(2/dt))*0.98;
end

%%%%

if food_eaten == 0

    nitrates_evolution = 0;

else

    total_nitrogen_eaten = food_eaten*nitrogen_food_content;
    /6.25 to have the nitrogen content
    total_nitrogen_absorbed = (fish_growth*0.5*0.2)/6.25;
    proteins, /6,25 for nitrogen content
    total_nitrogen_ejected = (total_nitrogen_eaten - total_nitrogen_absorbed)*0.85;

    nitrogen_moles_evolution = total_nitrogen_ejected/14;
    nitrates_moles_evolution = nitrogen_moles_evolution;
    nitrates_evolution = nitrates_moles_evolution*62*1000;

end
end

```

% This step is to include
 % the food consumption and
 % first "if" is to avoid
 % waste = 2%
 % 33% of proteines and
 % 50% of muscles, 20% of
 % [g], 15% is insoluble
 % [mg]

2.5.11 CONSOMMATION DE NITRATES « NITRATES_CONSUMPTION.M »

```

function nitrates_consumption_value = nitrates_consumption(lettuce_growth)

fresh_growth = lettuce_growth*(100/4);
weight !
proteins_growth = 0.015*fresh_growth;
nitrogen_growth = proteins_growth/6.25;
nitrogen_growth_mol = nitrogen_growth/14;

% 96% of water, following equations based on fresh
% [g] 1.3% of proteins
% [g] /6.25 for nitrogen content
% [moles]

```

```

nitrates_mol = nitrogen_growth_mol;           % [moles]
nitrates_growth = nitrates_mol*62;           % [g] (62 g/mol)

nitrates_consumption_value = nitrates_growth*1000; % [mg] 1000 for mg

end

```

2.5.12 FONCTIONS LIMITANT LA CROISSANCE DES POISSONS

2.5.12.1 Température de l'eau « *temperature_factor.m* »

```

function tau = temperature_factor(T)

global Topt1 Topt2 Tmax Tmin

if T<=Tmin
    tau = 0;
elseif T<= Topt1
    tau = (T-Tmin)/(Topt1 - Tmin);
elseif T<= Topt2
    tau =1;
elseif T <= Tmax
    tau = (Tmax-T)/(Tmax - Topt2);
else
    tau = 0;
end
end

```

2.5.12.2 pH de l'eau « *f_fish_pH.m* »

```

function pH_fish_factor = f_fish_pH(pH)

global pH_fish_min pH_fish_max pH_fish_opt1 pH_fish_opt2

if pH <= pH_fish_min
    pH_fish_factor = 0;
elseif pH <= pH_fish_opt1
    pH_fish_factor = (pH - pH_fish_min)/(pH_fish_opt1-pH_fish_min);
elseif pH <= pH_fish_opt2
    pH_fish_factor = 1;
elseif pH <= pH_fish_max
    pH_fish_factor = (pH_fish_max - pH)/(pH_fish_max-pH_fish_opt2);
else
    pH_fish_factor = 0;
end
end

```

2.5.12.3 Concentration en oxygène dissous « *DO_factor.m* »

```

function delta = DO_factor(DO)

global DMin DOpt

if DO < DMin % [-] dissolved oxygen factor
    delta = 0;
elseif DO < DOpt
    delta = (DO - DMin)/(DOpt - DMin);
else
    delta = 1;
end
end

```

2.5.12.4 Concentration en ammoniac « *UAm_factor.m* »

```
function nu = UAm_factor(A)

global Acrit Amax

if A < Acrit                                     % Unionized-Ammonia
    nu = 1;
elseif A < Amax
    nu = (Amax-A)/(Amax-Acrit);
else
    nu = 0;
end
end
```

2.5.12.5 Concentration en nitrates « *f_NO3_fish.m* »

```
function NO3_concentration_factor = f_NO3_fish(NO3_concentration)

global NO3_fish_ideal NO3_fish_max

if NO3_concentration <= NO3_fish_ideal
    NO3_concentration_factor = 1;
elseif NO3_concentration <= NO3_fish_max
    NO3_concentration_factor = (NO3_fish_max - NO3_concentration)/(NO3_fish_max-NO3_fish_ideal);
else
    NO3_concentration_factor = 0;
end
end
```

2.5.12.6 Nourriture « *food_factor.m* »

```
function f = food_factor(individual_fish_weight,i)

global r food_percentage food

R = individual_fish_weight*food_percentage/(2);           % [g/half-day/fish] maximum food quantity
r = food(i-1);                                           % [g/half-day/fish] optimal food quantity
f = r/R;                                                  % feeding level

end
```

2.5.13 FONCTIONS LIMITANT LA CROISSANCE DES LAITUES

2.5.13.1 Température de la solution « *f_temp_solution.m* »

```
function solution_temperature_factor = f_temp_solution(temperature)

global temp_solution_min temp_solution_opt_1 temp_solution_opt_2 temp_solution_max

if temperature <= temp_solution_min
    solution_temperature_factor = 0;
elseif temperature >= temp_solution_max
    solution_temperature_factor = 0;
elseif temperature <= temp_solution_opt_1
    solution_temperature_factor = (temperature - temp_solution_min)/(temp_solution_opt_1-temp_solution_min);
elseif temperature >= temp_solution_opt_2
    solution_temperature_factor = (temp_solution_max - temperature)/(temp_solution_max - temp_solution_opt_2);
else
    solution_temperature_factor = 1;
end
end
```

2.5.13.2 pH de la solution « *f_pH.m* »

```
function pH_factor = f_pH(pH)

global pHopt pHmin pHmax

if pH <= pHopt
    pH_factor = (pH - pHmin)/(pHopt-pHmin);
else
    pH_factor = (pHmax - pH)/(pHmax-pHopt);
end
end
```

2.5.13.3 Concentration en oxygène dissous « *f_DO_lettuces.m* »

```
function DO_factor = f_DO_lettuces(DO_lettuces)

global DO_min_lettuces DO_opt_lettuces

if DO_lettuces <= DO_min_lettuces
    DO_factor = 0;
elseif DO_lettuces <= DO_opt_lettuces
    DO_factor = (DO_lettuces - DO_min_lettuces)/(DO_opt_lettuces-DO_min_lettuces);
else
    DO_factor = 1;
end
end
```

2.5.13.4 Concentration en nitrates « *f_NO3_concentration.m* »

```
function NO3_concentration_factor = f_NO3_concentration(NO3_concentration)

global NO3opt NO3min

if NO3_concentration <= NO3min
    NO3_concentration_factor = 0;
elseif NO3_concentration <= NO3opt
    NO3_concentration_factor = (NO3_concentration - NO3min)/(NO3opt-NO3min);
else
    NO3_concentration_factor = 1;
end
end
```

2.5.13.5 Humidité relative de l'air « *f_HR.m* »

```
function HR_factor = f_HR(HR)

global HROpt HRmin HRmax

if HR <= HRmin
    HR_factor = 0;
elseif HR <= HROpt
    HR_factor = (HR - HRmin)/(HROpt-HRmin);
else
    HR_factor = (HRmax - HR)/(HRmax-HROpt);
end
end
```

2.6 GRAPHIQUES

Le script « *graphs.m* » produit les résultats finaux sous forme de graphiques.

```
global OUTcoef

figure(1)
    hold on
    plot(t,p31,'linewidth', 2);
    plot(t,p32,'linewidth', 2);
    plot(t,p33,'linewidth', 2);
    scatter(Weighting_time,fish_1,60,'filled')
    scatter(Weighting_time,fish_2,60,'filled')
    scatter(Weighting_time,fish_3,60,'filled')
    legend('Poisson le plus léger - modélisation','Poisson de masse moyenne - modélisation','Poisson le
plus lourd - modélisation','Poisson le plus léger - mesures','Poisson de masse moyenne - mesures','Poisson le
plus lourd - mesures','location','best');
    ylabel('Masse (g)')
    xlabel ('Temps (h)')
    grid on
    hold off

figure(2)
    hold on
    plot(t,p21,'linewidth', 2);
    plot(t,p22,'linewidth', 2);
    plot(t,p23,'linewidth', 2);
    scatter(Weighting_time,fish_1_length,60,'filled')
    scatter(Weighting_time,fish_2_length,60,'filled')
    scatter(Weighting_time,fish_3_length,60,'filled')
    legend ('Poisson le plus petit - modélisation','Poisson de taille moyenne - modélisation','Poisson
le plus grand - modélisation','Poisson le plus petit - mesures','Poisson de taille moyenne - mesures','Poisson
le plus grand - mesures','location','best');
    ylabel('Longueur (mm)')
    xlabel ('Temps (h)')
    grid on
    hold off

figure(3)
    hold on
    plot(t,p5,'linewidth', 2);
    plot(t,p18,'linewidth', 2);
    scatter(NO3_time,Fish_tank_NO3,60,'filled')
    scatter(NO3_time,Hydroponic_NO3,60,'filled')
    legend ('Aquaculture - modélisation','Hydroponie - modélisation','Aquaculture -
mesures','Hydroponie - mesures','location','best');
    ylabel('Concentration en nitrates (NO3 - ppm)')
    xlabel ('Temps (h)');
    grid on
    hold off

figure(4)
    hold on
    plot(t,p7,'linewidth', 1);
    plot(t,p9,'linewidth', 1);
    legend ('Aquarium','Hydroponic solution','location','best');
    ylabel('Volume (l)')
    xlabel ('Time (hrs)');
    grid on
    hold off

figure(5)
    hold on
    plot(t_rat,nit_prod, 'linewidth', 2);
    plot(t_rat,nit_cons, 'linewidth', 2);
    legend ('Production cumulée','Consommation cumulée','location','best');
    ylabel('Masse de nitrates (mg)')
    xlabel ('Temps (h)');
    grid on
    hold off

figure(6)
    plot(t,p14,'r', 'linewidth', 1);
    legend ('Evolution of lettuces LAI');
```

```

        ylabel('LAI')
        xlabel('Time (hrs)')
        grid on

figure(7)
hold on
    plot(OUTcoef*biomass(:,9), 'linewidth', 2);
    plot(OUTcoef*biomass(:,32), 'linewidth', 2);
    plot(OUTcoef*biomass(:,36), 'linewidth', 2);
    legend('Laitue 9','Laitue 32','Laitue 36','location','best');
    ylabel('Masse sèche (g)')
    xlabel('Temps (h)');
    grid on

hold off

figure(8)
hold on
    plot(t,p19, 'linewidth', 2);
    ylabel('Evapotranspiration totale (ml/h)')
    xlabel('Temps (h)');
    grid on

hold off

figure(9)
hold on
    plot(Evapotranspiration_times,Evapotranspiration_vector,'k', 'linewidth', 1);
    scatter(Evapotranspiration_times, Measured_evapotranspiration,'filled','k')
    ylabel('Evapotranspiration (ml/hrs.lettuce)')
    xlabel('Time (hrs)');
    legend('Fitted evapotranspiration','Measured evapotranspiration','location','best')
    grid on

hold off

figure(10)
hold on
    plot(biomass_total,'k', 'linewidth', 1);
    ylabel('Total lettuce dry biomass (g)')
    xlabel('Time (hrs)');
    grid on

hold off

figure(11)
hold on
    plot(abs(dp13), 'linewidth', 2);
    ylabel('Consommation totale de nitrates (mg/h)')
    xlabel('Temps (h)');
    grid on

hold off

figure(12)
hold on
    plot(OUTcoef*biomass_tot,'linewidth', 2);
    plot(19*p3,'linewidth', 2);
    ylabel('Masse totale (g)')
    xlabel('Temps (h)');
    legend('Laitues, masse sèche','Poissons')
    xlim([0 1201])
    ylim([0 1201])
    grid on

hold off

figure(13)
hold on
    plot(xyx,xyy,'k', 'linewidth', 1);
    scatter(lettuce_weight,lettuce_calculated_weight,30,PAR912,'filled')
    plot(lettuce_weight,yCalcl)
    ylabel('Masse sèche modélisée des salades [g]')
    xlabel('Masse sèche mesurée des salades [g]');
    colorbar_lettuce = colorbar;
    colorabar_v = get(colorbar_lettuce,'Title');
    set(colorabar_v,'string','PAR [W/m²]');
    xlim([0 14])
    ylim([0 14])
    grid on

hold off

figure(14)
hold on
    plot(xyx,xyy,'k','linewidth', 2);
    plot(lettuce_weight,yCalcl,'linewidth', 2)

```



```

scatter(lettuce_weight(harvest_times==408),lettuce_calculated_weight(harvest_times==408),60,PAR912(harvest_time
s==408),'x')

scatter(lettuce_weight(harvest_times==480),lettuce_calculated_weight(harvest_times==480),60,PAR912(harvest_time
s==480),'+')

scatter(lettuce_weight(harvest_times==576),lettuce_calculated_weight(harvest_times==576),60,PAR912(harvest_time
s==576),'*')

scatter(lettuce_weight(harvest_times==720),lettuce_calculated_weight(harvest_times==720),60,PAR912(harvest_time
s==720),'.')

scatter(lettuce_weight(harvest_times==816),lettuce_calculated_weight(harvest_times==816),60,PAR912(harvest_time
s==816),'h','filled')

scatter(lettuce_weight(harvest_times==936),lettuce_calculated_weight(harvest_times==936),60,PAR912(harvest_time
s==936),'p','filled')

scatter(lettuce_weight(harvest_times==984),lettuce_calculated_weight(harvest_times==984),60,PAR912(harvest_time
s==984),'s','filled')

scatter(lettuce_weight(harvest_times==1080),lettuce_calculated_weight(harvest_times==1080),60,PAR912(harvest_ti
mes==1080),'d','filled')

scatter(lettuce_weight(harvest_times==1152),lettuce_calculated_weight(harvest_times==1152),60,PAR912(harvest_ti
mes==1152),'o','filled')
    ylabel('Masse sèche modélisée des salades (g)')
    xlabel('Masse sèche mesurée des salades (g)');
    h_lettuce = legend('Axe y = x','Droite de régression','05-06-17','08-06-17','12-06-17','18-06-
17','22-06-17','27-06-17','29-06-17','03-07-17','06-07-17','location','best');
    v_lettuce = get(h_lettuce,'Title');
    set(v_lettuce,'string','Légende et dates de récolte')
    colorbar_lettuce = colorbar;
    colorabar_v = get(colorbar_lettuce,'Title');
    set(colorabar_v,'string','PAR (W/m²)');
    xlim([0 14])
    ylim([0 14])
    grid on

hold off

figure(15)
hold on
    plot(xyx,xyy,'k','linewidth',2);
    scatter(Fish_weights,Fish_calculated_weight,60,'filled','b')
    plot(Fish_weights,yCalc2,'linewidth',2)
    ylabel('Masses modélisées des poissons (g)')
    xlabel('Masses mesurées des poissons (g)');
    xlim([0 60])
    ylim([0 60])
    grid on

hold off

figure(16)
hold on
    plot(xyx,xyy,'k','linewidth',2);
    scatter(Nitrate_hydroponic_measured,Nitrate_predicted_hydroponic,60,'filled')
    plot(Nitrate_hydroponic_measured,yCalc3,'linewidth',2)
    ylabel('Concentrations modélisées en nitrates (ppm)')
    xlabel('Concentrations mesurées en nitrates (ppm)');
    legend('Hydroponie')
    xlim([0 900])
    ylim([0 900])
    grid on

hold off

figure(17)
hold on
    plot(xyx,xyy,'k','linewidth',2);
    scatter(Nitrate_fish_tank_measured,Nitrate_predicted_fish_tank,60,'filled')
    plot(Nitrate_fish_tank_measured,yCalc4,'linewidth',2)
    ylabel('Concentrations modélisées en nitrates (ppm)')
    xlabel('Concentrations mesurées en nitrates (ppm)');
    legend('Aquaculture')
    xlim([0 250])
    ylim([0 250])
    grid on

hold off

figure(18)
hold on

```

```

scatter(harvest_times,lettuce_weight,40,PAR0,'o','filled')
xlabel('Heure de récolte (h)')
ylabel('Masse sèche (g)')
colorbar_lettuce_2 = colorbar;
colorabar_v_2 = get(colorbar_lettuce_2,'Title');
set(colorabar_v_2,'string','PPFD initial (W/m²)');

grid on
hold off

figure(19)
hold on
scatter(harvest_times,lettuce_weight,40,PAR912,'o','filled')
xlabel('Heure de récolte (h)')
ylabel('Masse sèche (g)')
colorbar_lettuce_3 = colorbar;
colorabar_v_3 = get(colorbar_lettuce_3,'Title');
set(colorabar_v_3,'string','PPFD final (W/m²)');

grid on
hold off

figure(20)
hold on
plot(t,p5,'linewidth', 2);
scatter(NO3_time,Fish_tank_NO3,60,'filled')
legend ('Aquaculture - modélisation','Aquaculture - mesures','location','best');
ylabel('Concentration en nitrates (NO3 - ppm)')
xlabel ('Temps (h)');
grid on
hold off

```