Modeling of contaminant biodegradation and compound-specific isotope fractionation in chemostats at low dilution rates

Mehdi Gharasoo[†,‡,*], Benno N. Ehrl[‡], Olaf A. Cirpka[§], and Martin Elsner[†,‡,*]

[†]Technical University of Munich, Chair of Analytical Chemistry and Water Chemistry, Marchioninistr. 17, 81377 Munich, Germany
[‡]Helmholtz Zentrum München, Institute of Groundwater Ecology, Ingolstädter Landstr. 1, 85764 Neuherberg, Germany
[§]University of Tübingen, Center for Applied Geoscience, Hölderlinstr. 12, 72074 Tübingen, Germany
[*]Corresponding authors: Tel. +49 89 2180-78232, E-mail: m.elsner@tum.de, mehdi.gharasoo@helmholtz-muenchen.de

Supplementary material (number of pages: 5, number of listings:6)

The source code for all the MATLAB files essential for the modeling are listed here. Listing S.1 shows the source for the main file **`chemo_run.m`** where the parameter values are given as input. **`chemobio.m`** (Listing S.2) and **`chemo.m`** (Listing S.3) are called through **`chemo_run.m`** depending which system of equations (Eq. 1 or Eq. 2 in the main text) is intended to be solved. **`chemobio.m`** (Listing S.2) solves Eq. 2 where mass-transfer limitations are considered and uses the function file **`chmoFbiomm.m`** (Listing S.5) to call the relevant ODEs. In contrast, **`chemo.m`** (Listing S.3) solves Eq. 1 where mass-transfer limitations are neglected and uses the function file **`chmoFmm.m`** (Listing S.4) to call the relevant ODEs. Note that the ordinary differential equations are entered directly into the function files **`chmoFbiomm.m`** and **`chmoFmm.m`**. The smoothed periodic inlet is generated by **`chmoP.m`** (Listing S.6).

Listing S.1: Source code for the file **`chemo_run.m`**.

```
clearvars
close all
tic;

% final time
endtime=4*3600;
%Chemostat volume [ml]
chemo_vol=2000;
%Volume of drops [ml]
drop_vol=0.1;
%Time distance between the drops [s]
```

```
drop_rate= 20;
%Smoothing time of a droplet [s]
%(the larger the number, the better numerical stability is achieved)
drop_time=4;
% how the drop period will be generated.
drop_type = 2; %1:to reach to a specific value, 2:to have a specific area
%Time step for solution
time_step = drop_time/20;

%input concentration
Sin=3e4;% microgr/ L
%biomass filtration from outlet
%0=no filtration at outlet so biomass fully leaves the system (chemostat)
%1=complete filtration at outlet and so no biomass exits (full retentostat)
frac = 0;
%max growth rate of culture
mio = 0.11/3600;%/hr==>/s
%maintenance term
mm = 0.1/3600; %/hr ==> /s
%growth yield
yield = .018;
%maximum conversion rate
qmax = mio/yield -mm;
%half-saturation constant
km=237;% microgr/ L
%isotope fractionation factor
alfa = .9946;
%initial susbtrate concentration in tank
Sinit = 65; %microgr/ L
%initial concentration of biomass in tank
Xinit = 550;%microgr/ L
% mass limiting coefficient
ktr = .0025; %1/s

% To run the model with mass-transfer limitations at cell membrane, Eq. 2
chemobio
outlet_c_light = mean(SXmm(end-5*drop_rate:end,1))
delta13cbio = mean(deltac(end-5*drop_rate:end))

% To run the model without mass-transfer limitations at cell membrane, Eq. 1
chemo
delta13c = mean(deltac(end-10*drop_rate:end))

toc%shows the time elapsed
```

Listing S.2: Source code for the file **chemobio.m**.

```
tvin=0:time_step:endtime;
vin=drop_vol/chemo_vol.*chemoP(tvin,drop_time,drop_rate,drop_type); %Dilution
    rate

```

```
% Solving the ODEs
% with mass-transfer of substrate across the membrane and maintenance energy
SX0 = [Sinit Sinit*0.0112372 0 0 Xinit];
tspan = [0 drop_rate];
opts = odeset('RelTol',1e-5,'AbsTol',1e-15,'nonnegative',1:5,'MaxOrder',5);%,'
    MaxStep',time_step
sol = ode15s(@(t,SX) chemoFbiomm(t,SX,tvin,vin,Sin,frac,qmax,km,yield,alfa,ktr,mm
    ), tspan, SX0, opts);
tdiv = 40;%divide the distance of each period
t = transpose(linspace(0,drop_rate,tdiv));
SXmm = transpose(deval(sol,t));

% Extending the ODE solution for each drop
for iter=2:endtime/drop_rate

    solnew = odextend(sol,[],iter*drop_rate);
    tnew = transpose(linspace((iter-1)*drop_rate,iter*drop_rate,tdiv));
    SXnew = transpose(deval(solnew,tnew));

    t= [t; tnew];
    SXmm = [SXmm; SXnew];

    sol = solnew;
end

% plotting
figure;
subplot(4,1,1);plot(t,SXmm(:,1),'b-')
ylabel('^{12}S(\mu gr/l)');
subplot(4,1,2);plot(t,SXmm(:,2),'b-')
ylabel('^{13}S(\mu gr/l)');
subplot(4,1,3);plot(t,SXmm(:,5),'b-')
ylabel('Biomass(\mu gr/l)');
deltac = (SXmm(:,2)./SXmm(:,1)/0.0112372 -1).*1000;
deltac(SXmm(:,1)+SXmm(:,2)<1e-4)=nan;
subplot(4,1,4);plot(t,deltac,'b-')
ylabel('\delta ^{13}C');
xlabel('Time(s)');
```

Listing S.3: Source code for the file **chemo.m**.

```
tvin=0:time_step:endtime;
vin=drop_vol/chemo_vol.*chemoP(tvin,drop_time,drop_rate,drop_type); %Dilution
    rate

% Solving the ODEs
% no mass-transfer limittaions, with maintenance energy
SX0 = [Sinit Sinit*0.0112372 Xinit];
tspan = [0 drop_rate];
opts = odeset('RelTol',1e-5,'AbsTol',1e-15,'nonnegative',1:3,'MaxOrder',5);%,'
    MaxStep',time_step
```

```
sol = ode15s(@(t,SX) chemoFmm(t,SX,tvin,vin,Sin,frac,mio,km,yield,alfa,ktr,mm),
    tspan, SX0, opts);
t = transpose(linspace(0,drop_rate,40));
SXmm = transpose(deval(sol,t));
% Extending the ODE solution for each drop
for iter=2:endtime/drop_rate

    solnew = odextend(sol,[],iter*drop_rate);
    tnew = transpose(linspace((iter-1)*drop_rate,iter*drop_rate,40));
    SXnew = transpose(deval(solnew,tnew));

    sol = solnew;

    t= [t; tnew];
    SXmm = [SXmm; SXnew];
end

% Plotiing dilution rates
figure(1)
plot(tvin,vin)
axis([0 100 0 1e-4])
ylabel('Dilution rate (1/s)');
xlabel('Time(s)');
%plotting concentrations and isotope signatures
figure(2)
subplot(4,1,1);plot(t,SXmm(:,1),'r-')%,t,SX(:,2),'r-.',t,SX(:,3),'-o'
ylabel('^{12}S(\mu gr/l)');
subplot(4,1,2);plot(t,SXmm(:,2),'r-')%
ylabel('^{13}S(\mu gr/l)');
subplot(4,1,3);plot(t,SXmm(:,3),'r-')
ylabel('Biomass(\mu gr/l)');
deltac = (SXmm(:,2)./SXmm(:,1)/0.0112372 -1).*1000;
deltac(SXmm(:,1)+SXmm(:,2)<1e-4)=nan;
subplot(4,1,4);plot(t,deltac,'r-')
ylabel('\delta ^{13}C');
xlabel('Time(s)');
```

Listing S.4: Source code for the function file **chemoFmm.m**.

```
% ODEs of Eq. 1 in manuscript
% no mass-transfer limittaions, with maintenance energy
function dSX = chemoFmm(t,SX,tvin,vin,Sin,frac,miom,km,yield,alfa,ktr,mm)
    dSX = zeros(3,1);

    vint=interp1(tvin, vin, t,'pchip'); % Interpolate between tvin and vin at
        times t
    lSin=Sin;
    hSin=Sin*0.0112372;

    dSX(1) = vint*(lSin-SX(1))-SX(3)*miom*SX(1)/(SX(1)+SX(2)+km)/yield; %Light
        isotopologue
```

```
    dSX(2)= vint*(hSin-SX(2))-alfa*SX(3)*miom*SX(2)/(SX(1)+SX(2)+km)/yield; %
        Heavy isotopologue
    dSX(3) = SX(3)*miom*(SX(1)+alfa*SX(2))/(SX(1)+SX(2)+km)-(1-frac)*vint*SX(3)-
        mm*yield*SX(3); %Biomass
```

Listing S.5: Source code for the function file **chemoFbiomm.m**.

```
% ODEs of Eq. 2 in manuscript
% with mass-transfer of substrate across the membrane + maintenance energy
function dSX = chemoFbiomm(t,SX,tvin,vin,Sin,frac,qm,km,yield,alfa,ktr,mm)
    % 1: light S, 2: Heavy S, 3: bioavailable light S, 4: bioavailable heavy S,
        5: biomass X
    dSX = zeros(5,1);

    vint=interp1(tvin, vin, t,'pchip'); % Interpolate the data set (tvin, vin) at
         time t

    lSin=Sin;
    hSin=Sin*0.0112372;

    %% mass transfer limitations
    dSX(1) = vint*(lSin-SX(1)) - ktr*(SX(1)-SX(3)); %light
    dSX(2) = vint*(hSin-SX(2)) - ktr*(SX(2)-SX(4)); %Heavy
    dSX(3) = + ktr*(SX(1)-SX(3)) - SX(5)*qm*SX(3)/(SX(3)+SX(4)+km); %light bio
    dSX(4) = + ktr*(SX(2)-SX(4)) - alfa*SX(5)*qm*SX(4)/(SX(3)+SX(4)+km); %heavy
        bio
    dSX(5) = SX(5)*qm*yield*(SX(3)+alfa*SX(4))/(SX(3)+SX(4)+km) - (1-frac)*vint*
        SX(5)-mm*yield*SX(5); %biomass
```

Listing S.6: Source code for the function file **chemoP.m**.

```
% periodic function of t with period T.
% For 0 < t < r, the function is a polynomial spike with either height or area 1.
% For r < t < T, the function is zero.  Outside of 0 < t < T, the function is
    defined to be periodic.
function y = chemoP(t,r,T,sw)
    t = mod(t,T);
    switch sw
        case 1 %the function is a polynomial spike with height 1
            y = (t<r) .* (256*t.^4 .* (t/r-1).^4 ./ r.^4);
        case 2 %the function is a polynomial spike with area 1
            y = (t<r) .* (630*t.^4 .* (t/r-1).^4 ./ r.^5);
        otherwise
            error('Switch must be either 1 or 2!')
    end
```