MATHEMATICAL AND COMPUTER MODELING OF NONLINEAR BIOSYSTEMS I

COMPUTER LABORATORY XI: TC model of hemodialysis, data fitting

Ph. D. Programme 2013/2014

PhD STUDIES

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Project co-financed by European Union within the framework of European Social Fund

One-compartment model of hemodialysis (HD)

The dynamics imposed on the total solute mass in the total body water compartment is the following

$$\frac{d\left(V(t)C(t)\right)}{dt} = -K_d C(t) + G - K_r C(t)$$

where $C(t)$ is the solute concentration, G is the generation rate of the solute, K_r is the residual kidney clearance, and K_d is the dialyzer clearance.

We assume that the change of the total water volume in the body is described by the linear relationship

$$V(t) = V(0) + (G_w - UF)t$$

where G_w describes the water intake and ultrafiltration rate (UF) is non-zero (positive) only during the HD interval.

One-compartment model of HD

Question: How well the one-compartment model describes the reality?

In order to address that question we need to have access to clinical data concerning solute concentration dynamics during HD.

You can download such data sheet from:
www.mimuw.edu.pl/∼poleszczuk/files/dane.csv

Goal for today: Using MATLAB compare the one-compartment model with that data set.

Step 1 - loading data from file

Implement MATLAB function that reads the data from the csv file.

csv file contents:

- blood concentrations of urea during three subsequent hemodialysis (PUrea, mg/dL);
- total amount of ultrafiltrated water (UF, ml);
- total amount of drunk water (Gwhd, ml);
- measured total body water (TBW, ml);
- each patient in separate column;
- three subsequent dialysis sessions.

Requirements:

- output data structure with fields: PUrea, UF, Gwhd, TBW, times.

Step 1 - solution

```
function data = loadData()

b=importdata('dane.csv', ';', 1);

data.PUrea=b.data(strcmp(b.textdata(2:end,1),'PUrea'),2:end)/100;
data.UF=b.data(strcmp(b.textdata(2:end,1),'UF'),2:end);
data.Gwhd=b.data(strcmp(b.textdata(2:end,1),'Gwhd'),2:end);
data.TBW=b.data(strcmp(b.textdata(2:end,1),'TBW'),2:end);

data.times = b.data(strcmp(b.textdata(2:end,1),'PUrea'),1);

end
```

Step 2 - solving the model

Implement MATLAB function that return solution to the one-compartment model of HD.

$$\frac{d\left(V(t)C(t)\right)}{dt} = -K_d C(t) + G - K_r C(t)\,,$$

$$V(t) = V(0) + (G_w - UF)t$$

Input variables:

- timespan for the solution;
- structure with parameters values;
- initial condition.

Output parameters:

- if input timespan was given as a mesh of points, then return the solution evaluated at that mesh. Return default ode45 solver output, otherwise.

Step 2 - solution

```
function sol = oneCompartmentSolve( T, par, init )

    sol = ode45(@model, [0 T(end)],init);
    if length(T)>1
        sol = deval(sol,T);
    end

    function y = model(~,x)
       y = zeros(2,1);
       y(2) = (-par.Uf+par.Gw);
       y(1) = (-par.Kd*x(1)+par.G-par.Kr*x(1)-x(1)*y(2))/x(2);
    end

end
```

Step 3 - create setting for specific patient

Implement MATLAB function that selects the data for a given patient, initiates parameters and returns the initial conditions. Select data for first HD session only. Assume that there was no urea generation and no residual kidney function.

Input variables:

- structure with clinical data from step 1;
- patients number.

Output variables:

- structure with parameters values;
- initial conditions;
- structure with data for a given patient.

Step 3 - solution

```
function [params, init, data] = patientFromData(data, which)
    params.Thds = data.times(1:5);
    params.Uf = data.UF(1,which)/params.Thds(end);
    params.Gw = data.Gwhd(1,which)/params.Thds(end);

    params.G = 0;
    params.Kr = 0;
    params.Kd = 150;

    data.times = data.times(1:5);
    data.PUrea = data.PUrea(1:5,which);

    init = [data.PUrea(1); data.TBW(1,which)];
end
```

Step 4 - data fitting procedure

Implement MATLAB function that estimates dialyzer clearance for a specific patient.

Input variables:

- default parameters;
- initial conditions for the model solution;
- data specific to a given patient.

Output variables:

- structure with estimated parameters;
- fit error.

Step 4 - solution

```
function [paramsF, err] = fitToData( params, init, dataPat )
    paramsF = params;

    [paramsF.Kd, err] = fminsearch(@F,params.Kd);

    function y = F(x)
       params.Kd = x;
       sol = oneCompartmentSolve(dataPat.times, params, init);
       y = sum((sol(1,:)-dataPat.PUrea').^2);
    end
end
```

Step 5 - plotting function

Implement two MATLAB plotting functions:

- Plot a single patient solution with corresponding data points.
- Plot an averaged solution and averaged data (with standard deviation).

Input variables:

- time mesh of the solution;
- model solution (averaged model solution);
- data specific to a given patient (data for all patients).

Step 5 - plotting function 1

```
function plotPatient(tmesh, sol, dataPat )

 set(0,'DefaultAxesFontSize',18)

    figure(1)
    pl = plot(tmesh,sol(1,:),dataPat.times,dataPat.PUrea);
    set(pl(1), 'LineWidth',2);
    set(pl(2), 'Marker','o','LineStyle','none',...
               'MarkerFaceColor','r');
    legend({'Model','Data'})
    xlabel('Time (min)')
    ylabel('Urea concentration (mg/ml)')
end
```

Step 5 - plotting function 2

```
function plotComparizon(tmesh, solAv, data )

 set(0,'DefaultAxesFontSize',18)

    figure(2)
    clf
    hold on
    plot(tmesh,solAv(1,:), 'LineWidth',2);
    errorbar(data.times(1:5), mean(data.PUrea(1:5,:),2),...
             std(data.PUrea(1:5,:),[],2),'or');
    hold off
    legend({'Model','Data'})
    xlabel('Time (min)')
    ylabel('Urea concentration (mg/ml)')

end
```

Step 6 - main script

Implement MATLAB script in which for each patient:

- dialyzer clearance is estimated;
- model solution and data are plotted.

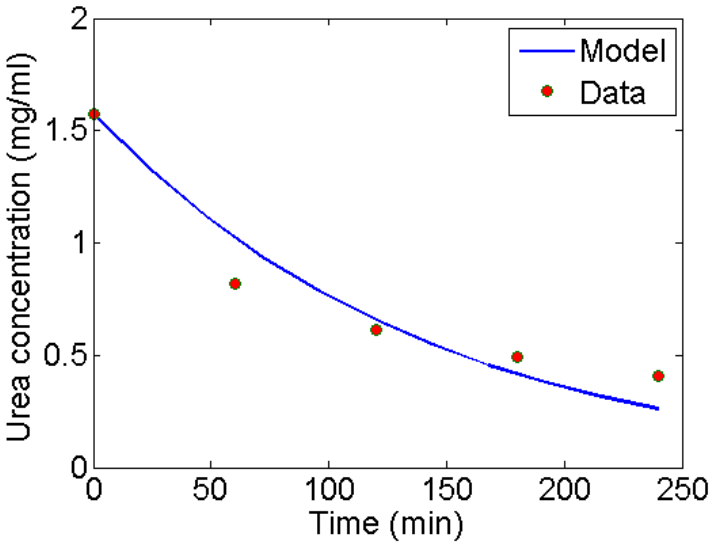
Plot also averaged solution together with the averaged data (with standard deviations).

Step 6 - solution

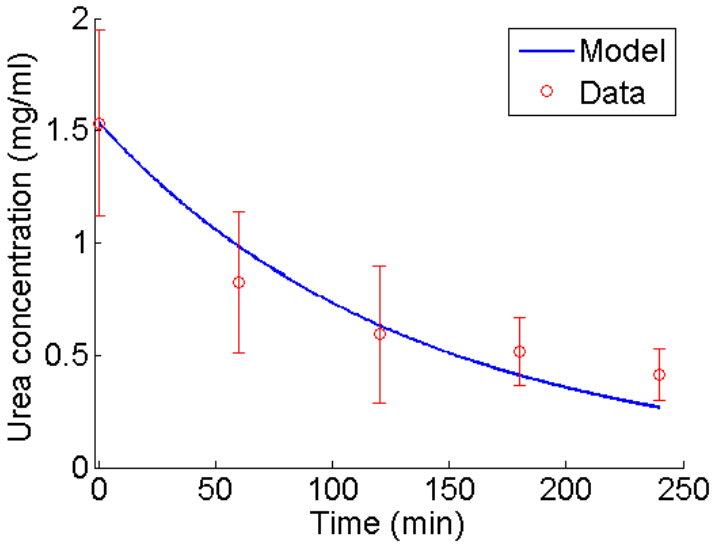
```
clear all;
data = loadData();
solAv = 0;
noPatients = size(data.UF,2);

for i=1:noPatients
    [params, init, dataPat] = patientFromData(data, i);

    paramsF = fitToData(params, init, dataPat);

    sol = oneCompartmentSolve(0:240, paramsF, init);

    plotPatient(0:240, sol, dataPat )
    pause
    solAv = solAv + sol(1,:)/noPatients;
end

plotComparizon(0:240, solAv, data )
```

Results

Exemplary patient



Averaged trajectory



One-compartment model fails to correctly predict the clinically obtained data.

Two-compartment model of HD

In the two-compartment model it is assumed that the body fluid is divided into two compartments: directly ($V_e(t)$) and indirectly accessible for the dialyzer ($V_e(t)$). The following dynamics is imposed

$$\begin{cases} \dfrac{d\left(V_e(t)C_e(t)\right)}{dt} = K_c(C_i(t) - C_e(t)) - K_d C_e(t) + G - K_r C_e(t)\,, \\ \dfrac{d\left(V_i(t)C_i(t)\right)}{dt} = -K_c(C_i(t) - C_e(t))\,. \end{cases}$$

Total body water is described by the same equation as previously

$$V(t) = V(0) + (G_w - UF)t\,,$$

and $V_e(t) = \alpha V(t)$, $V_e(t) + V_i(t) = V(t)$.

Goal: Compare the TC model with the clinical data. Modify functions used for one-compartment model. Estimate K_c. Assume $C_e(0) = C_i(0)$.

Step 1 - model solution

```
function sol = twoCompartmentSolve( T, par, init )

    sol = ode45(@model, [0 T(end)],[init(1);init(1);init(2)]);
    if length(T)>1
        sol = deval(sol,T);
    end

    function y = model(~,x)
       y = zeros(3,1);
       y(3) = (-par.Uf+par.Gw);
       y(1) = (par.Kc*(x(2)-x(1))-par.Kd*x(1)+par.G-...
           par.Kr*x(1)...
           -par.alpha*x(1)*y(3))/(par.alpha*x(3));
       y(2) = (-par.Kc*(x(2)-x(1))-...
              (1-par.alpha)*x(2)*y(3))/((1-par.alpha)*x(3));
    end

end
```

Step 2 - modify patientFromData function

```
function [params, init, data] = patientFromData(data, which)
    params.Thds = data.times(1:5);
    params.Uf = data.UF(1,which)/params.Thds(end);
    params.Gw = data.Gwhd(1,which)/params.Thds(end);

    params.G = 0;
    params.Kr = 0;

    params.Kd = 150;

    params.Kc = 300;
    params.alpha = 0.63;

    data.times = data.times(1:5);
    data.PUrea = data.PUrea(1:5,which);

    init = [data.PUrea(1); data.TBW(1,which)];
end
```

Step 3 - modify data fitting procedure

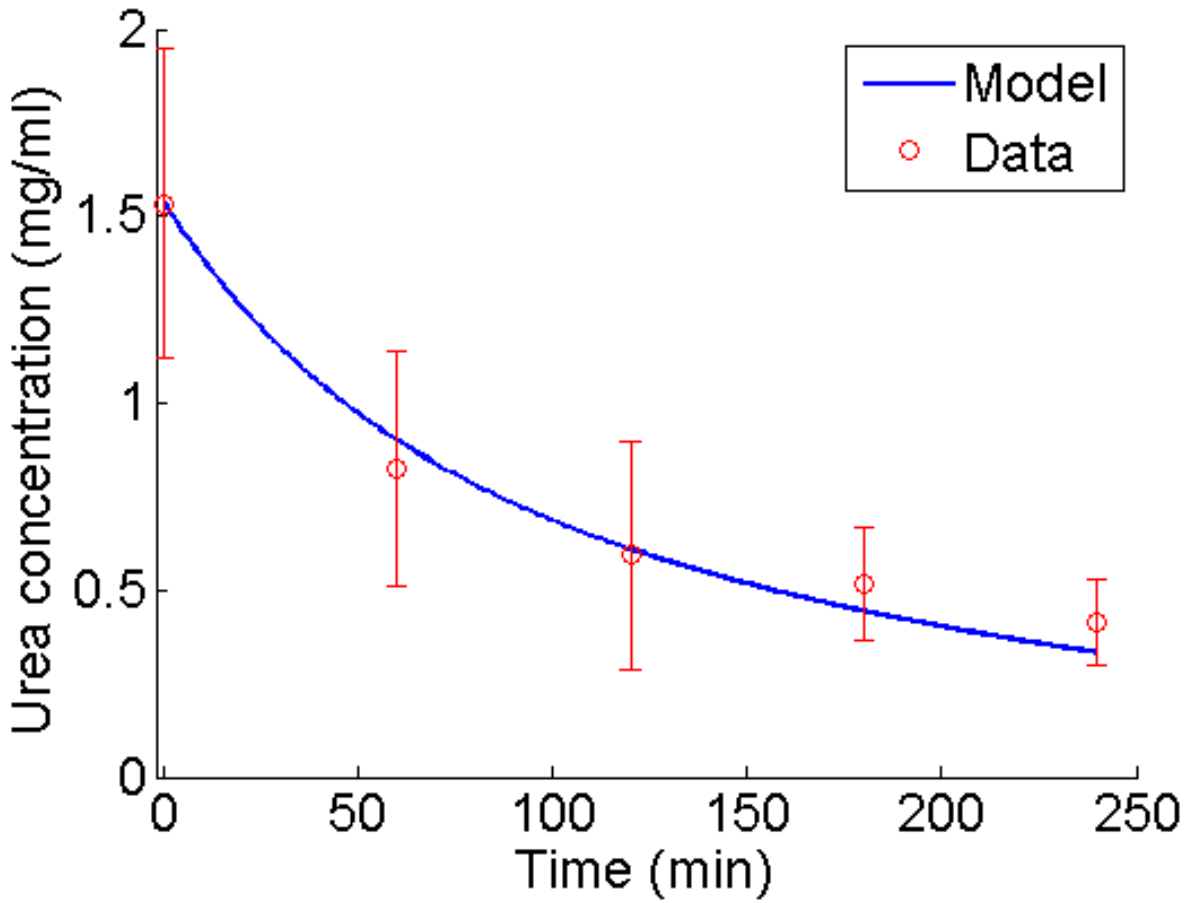
```
function [paramsF, err] = fitToDataTC( params, init, dataPat )
    paramsF = params;

    [par, err] = fminsearch(@F,[params.Kd params.Kc]);
    paramsF.Kd = par(1);
    paramsF.Kc = par(2);

    function y = F(x)
       params.Kd = x(1);
       params.Kc = x(2);
       sol = twoCompartmentSolve(dataPat.times, params, init);
       y = sum((sol(1,:)-dataPat.PUrea').^2);
    end
end
```

Step 4 - modify the main script

```
clear all;
data = loadData();
solAv = 0;
noPatients = size(data.UF,2);

for i=1:noPatients%for each patient
    [params, init, dataPat] = patientFromData(data, i);

    paramsF = fitToDataTC(params, init, dataPat);

    sol = twoCompartmentSolve(0:240, paramsF, init);

    plotPatient(0:240, sol, dataPat )
    pause
    solAv = solAv + sol(1,:)/noPatients;
end

plotComparizon(0:240, solAv, data )
```

Results

Averaged trajectory



Two-compartment model predicts the clinically obtained data much better than the one-compartment model.