Matlabsessie 1

# 1. Simuleren van substraat/productmodellen

Oplossing A

**SP modellen A**

close all

clear all

%%%variabelen en parameters

K\_s=1;%mM

mu\_max=1; %h^-1

Y\_xp=0.2; %g/g

S\_0=50; %mM

X\_0=5; %mM

P\_0=0; %mM

Y\_xs=0.5; %g/g

m\_x=0.1; %h^-1

%%%integratie

tspan=0:0.001:10;

values=[X\_0 S\_0 P\_0]; %beginwaarden

[t,y] = ode45(@(t,y)MonodModelA(t,y,mu\_max,K\_s,Y\_xs,Y\_xp),tspan,values);

%%%figuur

plot(t,y(:,1),'k'); %zwarte curve

hold on

plot(t,y(:,2),'g'); %groene curve

plot(t,y(:,3),'r'); %rode curve

legend('cel massa', 'substraat', 'product')

title('simulatie Batch met Monodkinetiek')

xlabel('tijd [h]')

ylabel ('concentratie')

%%%Opslaan van figuur in png-bestand

print('figuur1\_1A','-dpng');

**MonodModelA**

function M=MonodModelA(t,y,mu\_max,K\_s,Y\_xs,Y\_xp)

M=zeros(3,1);

X=y(1); %celconcentratie

S=y(2); %substraatconcentratie

P=y(3); %productconcentratie

M(1)=(mu\_max\*S\*X)/(K\_s+S);

M(2)=-(mu\_max\*S\*X)/(Y\_xs\*(K\_s+S));

M(3)=(mu\_max\*S\*X)/((K\_s+S)\*Y\_xp);

end

Oplossing B

Zie A, enkel andere formule voor dX/dt

# 2. Simuleren van enzymatische processen

Oplossing A

close all

clear all

%%%variabelen en parameters

k=0.5\*10^(-3); %h^-1

k\_p=100; %mM

k\_s=100; %mM

V\_maxs=12.5; %g/L\*h

K\_eq=10; %U/mM

enzym\_range= [0 50]; %U/g

K\_m=[0.1 0.2 0.3 0.4]; %mM OF K\_m=0.05:0.05:0.4

V\_max=[1 2 3 4]; %mM/min OF Vmax=1.5:1.5:4.5

S=0:0.01:10; %mM

%%%formule

for i=1:length(K\_m)

v(i,:)=V\_max(3).\*(S./(K\_m(i)+S)); %Michaelis Menten met variabele K\_m

end

for j=1:length(V\_max)

V(j,:)=V\_max(j).\*(S./(K\_m(2)+S)); %Michaelis Menten met variabele V\_max

end

%%%figuur

figure(1)

subplot(2,2,1)

plot(S,v)

title('Verschillende Km waarden')

xlabel('Substraat concentratie (mM)')

ylabel('Reactie snelheid (mM/min)')

subplot(2,2,2)

plot(S,V)

title('Verschillende Vmax waarden')

xlabel('Substraat concentratie (mM)')

ylabel('Reactie snelheid (mM/min)')

subplot(2,2,3)

plot(S,v,S,V)

title('Verschillende Km en Vmax waarden')

xlabel('Substraat concentratie (mM)')

ylabel('Reactie snelheid (mM/min)')

Oplossing B

Zie A, enkel andere formule gebruiken.

Oplossing C

clear all

close all

%opdracht C

%As we are calculating initial speed, where P=0, the graphs are IDENTICAL

%with the no inhibition Michaelis-Menten

Km=0.05:0.05:0.4 %mM

Vmax=1.5:1.5:4.5 %mM/min

S=0:0.01:100 %mM

Kp=10 %mM

P=100:0.01:1;

for i=1:length(S)

P(i)=100-S(i);

for ii=1:length(Km)

VKm(ii,i) = Vmax(2) .\*S(i)./(Km(ii)+S(i)+P(i)/Kp);%Michaelis-Menten formule: VKm-waarden berekenen% : in dim van VKm is voor varierende S

end

end

for i=1:length(S)

P(i)=100-S(i);

for jj=1:length(Vmax)

VVmax(jj,i) = Vmax(jj) .\*S(i)./(Km(4)+S(i)+P(i)/Kp);%Michaelis-Menten formule: VVmax-waarden berekenen

end

end

%Two subplots comparing Vmax and Km values

figure(1)

subplot(1,2,1)

h=plot(S,VKm)

xlabel('Substrate concentration [mM]')

ylabel('Reaction speed [mM/min]')

title('Different Km values (Vmax=3 mM/min)')

legend(h,num2str(Km),'Location','SouthEast')

subplot(1,2,2)

h2=plot(S,VVmax)

xlabel('Substrate concentration [mM]')

ylabel('Reaction speed [mM/min]')

title('Different Vmax values (Km=0.2 mM)')

legend(h2,num2str(Vmax),'Location','SouthEast')

% One plot

figure(2)

plot(S,VKm,S,VVmax) % plotting all curves on one graph

xlabel('Substrate concentration [mM]')

ylabel('Reaction speed [mM/min]')

title('Comparison of different Vmax and Km values')

for ii=1:length(Vmax)

text(S(round(end\*2/3)),VVmax(ii,round(2\*end/3)),strcat('Km=0.2 Vmax=',num2str(Vmax(ii))))

% puts the Vmax value around the 2/3 of the S range

end

for jj=1:length(Km)

int=floor(36+(jj-1)\*(110-36)/length(Km))-1

text(S(int-3),VKm(jj,int),num2str(Km(jj)),'Fontsize',5)

%puts the Km values on the graph, with increasing S

%It can be placed rigth on the diagonal, but that needs extra computing

end

Oplossing D

**EnzymkinetiekD**

close all

clear all

enzym\_range=0:5:50; %U/g

k=0.5d-3; %h^-1

tspan=[0:0.01:50]

for i=1:length(enzym\_range)

[t,y(i,:)]=ode45(@(t,y)enzymactiviteitD(t,y,k),tspan,enzym\_range(i));

end

figure(1)

subplot(1,2,1)

plot(t,y)

title('enzymactiviteit in functie van de tijd met k=0,5\*10^-3')

xlabel('tijd (h)')

ylabel('Enzymactiviteit (U/g)')

k=0.5d-2; %h^-1

for i=1:length(enzym\_range)

[t,y(i,:)]=ode45(@(t,y)enzymactiviteitD(t,y,k),tspan,enzym\_range(i));

end

figure(1)

subplot(1,2,2)

plot(t,y)

title('enzymactiviteit in functie van de tijd met k=0,5\*10^-2')

xlabel('tijd (h)')

ylabel('Enzymactiviteit (U/g)')

**EnzymactiviteitD**

function dy = enzymactiviteitD(t,y,k)

dy = zeros(1,1);

dy(1)= -k\*y(1);

end

Oplossing E

close all

clear all

%%%variabelen en parameters

K\_m=0.2; %mM

V\_max=3; %mM/min

substraat\_range=0:1:10; %mM

S0=substraat\_range(end);

tspan=[0:0.01:5];

%%%vergelijking

dy=@(t,y,V\_max,K\_m)(-V\_max \* y)/(y+K\_m);

[T,Y]=ode45(@(t,y)dy(t,y,V\_max,K\_m),tspan,S0);

S=Y;

v=V\_max.\*S./(K\_m+S);

%%%figuur

[ax,h1,h2]=plotyy(T,v,T,Y);

title('Enzyme reaction')

xlabel('Time [min]')

ylabel(ax(1),'Remaining Enzyme activity [-]')

ylabel(ax(2),'Substrate concentration [mM]')

Oplossing F

close all

clear all

%%%variabelen en parameters

K\_m=0.2; %mM

V\_max=3; %mM/min

k\_s=100; %mM

substraat\_range=0:1:10; %mM

S0=substraat\_range(end);

tspan=[0:0.01:5];

%%%vergelijking

dy=@(t,y,V\_max,K\_m,k\_s)(-V\_max \* y)/(y+K\_m+((y\*y)/k\_s));

[T,Y]=ode45(@(t,y)dy(t,y,V\_max,K\_m,k\_s),tspan,S0);

S=Y;

v=V\_max.\*S./(K\_m+S);

%%%figuur

plot(T,S);

title('Michaelis Menten met substraatinhibitie')

xlabel('tijd [h]')

ylabel ('substraatconcentratie [mM]')

Oplossing G

Zie E en F

Matlabsessie 2

# Opgave A

GROEIMODEL MET MONODKINETIEK

Fit de parameters van het volgende model met Monodkinetiek dat de groei en het substraatverbruik in een batchreactor weergeeft

dX/dt=(μ\_max\*S\*X)/(K\_S+S)  and  dS/dt=-1/Y\_(X/S)   (μ\_max\* S\*X)/(K\_S+S)

Dit is vergelijkbaar aan Opgave 1A van Matlabsessie1, maar zonder productvorming.

Input:

* Datafile\_OD met voor verschillende tijdstippen de hoeveelheid cellen en de substraatconcentratie.

Gevraagd:

* Om te beginnen schrijf je de functie die de twee differentiaalvergelijkingen integreert voor een gegeven parameterset. Gebuik als initiële waarden: µ\_max=2; K\_S=8 ; Y\_(X/S)=0.4
* Je schrijft een ‘cost’-functie die het verschil tussen de experimentele waarden en de gesimuleerde output berekent.
* Je gebruikt de lsqnonlinfunctie voor het bekomen van de gefitte parameters.

Output:

* Geschatte parameters
* Plot met data & modelsimulatie van X en S
* 3 programma’s

# Opgave C

Je start hier van het geschreven programma in Opgave A. Je past dit programma aan zodat de standaardafwijking op de parameters bepaald wordt door gebruik te maken van de covariantiematrix. De outputfile met parameters bevat nu ook de standaardafwijkingen.

In deze opgave wordt een figuur gevraagd bestaande uit vier subplots;

1. Boven 2 subplots het gefitte model met experimentele data voor celconcentratie en substraatconcentratie

2. Onder de corresponderende figuur de residuals. Gebruik hiervoor de functie bar().

Upload aangepaste programmafiles, datafile met parameterinformatie en figuur in Blackboard.

# Oplossing A&C

**Hoofdprogramma C**

close all

clear all

data = load('datafileOD.txt');

x0 = [2 8 0.4 data(1,2) data(1,3)];

%mu\_max=2; h^-1

%K\_s=8; g/L

%Y\_xs=0.4; g/g

%X\_0=data(1,2) (OD)

%S\_0=data(1,3) g/L

%lb=[0.1 0 0 0 0]; %lower bounds

%ub=[2 2.5 10 20 20]; %upper bounds

lb=[1d-10 1d-10 1d-5 1d-3 1d-3];

ub=[1d5 1d5 1d5 5 5 ];

% input lsqnonlin

options=optimset('MaxFunEvals',200000,'Tolfun',1e-14,'MaxIter',30000);

% optimisation met lsqnonlin

[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(@costC,x0, lb, ub,options,data);

%output lsqnonlin

mu\_max=x(1);

K\_s=x(2);

Y\_xs=x(3);

X\_0=x(4);

S\_0=x(5);

save('geoptimaliseerde parameters C', 'x');

SSE=resnorm;

J=jacobian;

tspan=[0 data(1,end)]; %!!!!

values=[X\_0 S\_0];

[t,y] = ode23s(@(t,y)MonodC(t,y,mu\_max,K\_s,Y\_xs),tspan,values);

% calculation of mean square error (MSE) = SLE/(nt-np)

% nt is number of datapoints = L

% np is number of parameters = 5

numberOfParameters = length(x);

L=length(data);

MSE = SSE / (L - numberOfParameters);

% calculation of covariance matrix

% covariance matrix ~= MSE \* inverse( J^(T)\* J)

covp = MSE\*inv(transpose(J)\*J);

%OFWEL

%covp = MSE\*inv(transpose(J)\*J);

%fisher=(1/MSE)\*J.'\*J % Fisher informatiematrix

%covp=inv(fisher) % variantie-covariantiematrix

% Calculation of standard deviations of the 5 parameters

% standard deviation = sqrt(diagonal elements)

SE\_mu\_max=sqrt(covp(1,1));

SE\_K\_s=sqrt(covp(2,2));

SE\_Y\_xs=sqrt(covp(3,3));

SE\_X\_0=sqrt(covp(4,4));

SE\_S\_0=sqrt(covp(5,5));

SEpar=[mu\_max SE\_mu\_max; K\_s SE\_K\_s; Y\_xs SE\_Y\_xs; X\_0 SE\_X\_0; S\_0 SE\_S\_0];

SEpar=full(SEpar);

save('parest\_SE.txt', 'SEpar','-ascii','-double')

%celconcentratie plotten

figure(1)

subplot(2,2,1)

plot(data(:,1),data(:,2), 'o') % experimentele data

hold on

plot(t,y(:,1),'-') %geoptimaliseerd model

hold on

title('Celconcentratie')

xlabel('tijd (h)')

ylabel('concentratie X (OD)')

legend('experimentele data', 'geoptimaliseerd model')

%substraatconcentratie plotten

subplot(2,2,2)

plot(data(:,1),data(:,3), 'o') % experimentele data

hold on

plot(t,y(:,2),'-') %geoptimaliseerd model

hold on

title('Substraatconcentratie')

xlabel('tijd (h)')

ylabel('concentratie S (g/L)')

legend('experimentele data', 'geoptimaliseerd model')

%residuals plotten

subplot(2,2,3)

bar(data(:,1),residual(1:10)) %residuals plotten

xlabel('tijd (h)')

title('Residuals cellen')

%residuals plotten

subplot(2,2,4)

bar(data(:,1),residual(11:20)) %residuals plotten

xlabel('tijd (h)')

title('Residuals substraat')

saveas(gcf,'figure3.eps','psc2')

**Monod C**

function M=MonodC(t,y,mu\_max,K\_s,Y\_xs)

M=zeros(2,1);

X=y(1);

S=y(2);

M(1)=(mu\_max\*S\*X)./(K\_s+S);

M(2)=-(1/Y\_xs)\*(mu\_max\*S\*X)./(K\_s+S);

end

**CostC**

function C=costC(x,data)

mu\_max=x(1);

K\_s=x(2);

Y\_xs=x(3);

X\_0=x(4);

S\_0=x(5);

tspan=data(:,1);

values=[X\_0 S\_0];

[t,y] = ode23s(@(t,y)MonodC(t,y,mu\_max,K\_s,Y\_xs),tspan,values);

%C=[(y(:,1)-data(:,2)) (y(:,2)-data(:,3))];

C = [(data(:,2)-y(:,1));(data(:,3)-y(:,2))];

end

# Oplossing B & D: modelleren van microbiële groei

**Hoofdprogramma D**

clear all

close all

load datafileB.dat;

datafileB(:,3)=log(datafileB(:,2));

data=[datafileB(:,1) datafileB(:,3)];

L=length(data);

N\_0=data(1,2);

Q\_0=1.0;

mu\_max=1;

N\_max=1;

x0 = [N\_0 Q\_0 mu\_max N\_max];

lb = [1 0 0 0]; %lower bounds

ub = [20 25 10 22]; %upper bounds

% input lsqnonlin

options=optimset('MaxFunEvals',200000,'Tolfun',1e-14,'MaxIter',30000);

% optimisation met lsqnonlin

[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(@costD,x0,lb,ub,options,data);

N\_0=x(1);

Q\_0=x(2);

mu\_max=x(3);

N\_max=x(4);

save('geoptimaliseerde parameters D', 'x');

SSE=resnorm;

J=jacobian;

tspan=data(:,1);

values=[N\_0 Q\_0];

[t,y] = ode23s(@bar\_rob,tspan,values,foptions,mu\_max,N\_max);

% calculation of mean square error (MSE) = SLE/(nt-np)

% nt is number of datapoints = L

% np is number of parameters = 5

numberOfParameters = length(x);

L=length(data);

MSE = SSE / (L - numberOfParameters);

% calculation of covariance matrix

% covariance matrix ~= MSE \* inverse( J^(T)\* J)

covp = MSE\*inv(transpose(J)\*J);

%OFWEL

%covp = MSE\*inv(transpose(J)\*J);

%fisher=(1/MSE)\*J.'\*J % Fisher informatiematrix

%covp=inv(fisher) % variantie-covariantiematrix

% Calculation of standard deviations of the 4 parameters

% standard deviation = sqrt(diagonal elements)

SE\_N\_0=sqrt(covp(1,1));

SE\_Q\_0=sqrt(covp(2,2));

SE\_mu\_max=sqrt(covp(3,3));

SE\_N\_max=sqrt(covp(4,4));

SEpar=[N\_0 SE\_N\_0; Q\_0 SE\_Q\_0; mu\_max SE\_mu\_max; N\_max SE\_N\_max];

SEpar=full(SEpar);

save('parest\_SE.txt', 'SEpar','-ascii','-double')

%plot maken

figure(1)

subplot(1,2,1)

plot(data(:,1),data(:,2),'o') % experimentele data

hold on

plot(t,y(:,1),'-') %geoptimaliseerd model

hold on

title ('Baranyi & Roberts model')

xlabel('tijd (h)')

ylabel('ln (N) (ln(cfu/mL))')

subplot(1,2,2)

bar(data(:,1),residual(1:length(data))) %residuals plotten

xlabel('tijd (h)')

title('Residuals')

saveas(gcf,'figure4.eps','psc2')

**Bar\_rob**

function BR=bar\_rob(t,y,mu\_max,N\_max)

BR=zeros(2,1);

N=y(1);

Q=y(2);

BR(1)=(Q./(1+Q))\*mu\_max\*(1-exp(N-N\_max));

BR(2)=mu\_max\*Q;

End

**CostD**

function C=costD(x,data)

N\_0=x(1);

Q\_0=x(2);

mu\_max=x(3);

N\_max=x(4);

tspan=data(:,1);

values=[N\_0 Q\_0];

[t,y] = ode23s(@bar\_rob,tspan,values,foptions,mu\_max,N\_max);

C=[(y(:,1)-data(:,2))];

end

Matlabsessie 3

# Opgave A

Fit de parameters van het Michealis Menten model voor enzymkinetiek.

Vergelijking: V=Vmax.S/(Km+S)

Input:

* DataMichaelisMenten1 met voor verschillende substraatconcentraties (kolom 1) de enzymactiviteit (kolom 2).

Gevraagd:

* Gebruik de informatie van in Matlabsessie 1 om de intiële waarden van de parameters te kiezen.
* Je schrijft een ‘cost’-functie die het verschil tussen de experimentele waarden en de gesimuleerde output berekent.
* Je gebruikt de lsqnonlinfunctie voor het bekomen van de gefitte parameters.

Output:

* Geschatte parameters met standaardafwijking.
* Plot met data & modelsimulatie van V in functie van S.
* Upload de programma's, de outputfile met de parameters en de figuur.

**Hoofdprogramma A (kostfunctie, zie C)**

close all

clear all

data=load('DataMichaelisMenten1.txt'); %S en V

S\_data=data(:,1);

V\_data=data(:,2);

V\_max=3; %mM/min

K\_m=0.2; %mM

x0 = [V\_max K\_m];

lb=[0 0]; %lower bounds

ub=[5 5]; %upper bounds

% input lsqnonlin

options=optimset('MaxFunEvals',200000,'Tolfun',1e-14,'MaxIter',30000);

% optimisation met lsqnonlin

[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(@costA,x0, lb, ub,options,data);

%output lsqnonlin

V\_max=x(1);

K\_m=x(2);

save('geoptimaliseerde parameters A', 'x');

%modelsimulatie

V=V\_max\*S\_data./(K\_m+S\_data);

%enzymactiviteit plotten (V in functie van S)

plot(S\_data,V\_data, 'o') % experimentele data

hold on

plot(S\_data,V,'-') %geoptimaliseerd model

hold on

title('Michaelis Menten kinetiek')

xlabel('substraatconcentratie (M)')

ylabel('enzymactiviteit (M/min)')

legend('experimentele data', 'geoptimaliseerd model')

saveas(gcf,'figureA.eps','psc2')

# Opgave C

Deze opgave is een uitbreiding op Opgave 3A.

Gevraagd:

* Evalueer de estimator SSE voor een range van verschillende parameterwaarden Km en Vmax rond de eerder geschatte optimale parameters in Opgave 3A door een rooster (een grid) te creëren van mogelijke combinaties van parameterwaarden.

Output:

* Plot een contourplot van de SSE-waarden met gepaste ‘hoogtelijnen’ (met bijschrijven van de SSE-waarden) in functie van de verschillende waarden van de twee parameters. Zoek bij de Matlabdocumentatie naar “contour plot of a matrix” voor de syntax.
* Plot met dezelfde informatie een response surface plot van de SSE i.f.v. de twee parameters. Zoek bij de Matlabdocumentatie naar ‘surf’ voor de syntax.
* Upload het programma, de contourplot en de surf plot.

**Hoofdprogramma C**

close all

clear all

data=load('DataMichaelisMenten1.txt'); %S en V

S\_data=data(:,1);

V\_data=data(:,2);

V\_max=3; %mM/min

K\_m=0.2; %mM

x0 = [V\_max K\_m];

lb=[0 0]; %lower bounds

ub=[10 10]; %upper bounds

% input lsqnonlin

options=optimset('MaxFunEvals',200000,'Tolfun',1e-14,'MaxIter',30000);

% optimisation met lsqnonlin

[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(@costC,x0, lb, ub,options,data);

%output lsqnonlin

V\_max=x(1);

K\_m=x(2);

save('geoptimaliseerde parameters C', 'x');

SSE=resnorm;

J=jacobian;

% calculation of mean square error (MSE) = SLE/(nt-np)

% nt is number of datapoints = L

% np is number of parameters = 5

numberOfParameters = length(x);

L=length(data);

MSE = SSE / (L - numberOfParameters);

% calculation of covariance matrix

% covariance matrix ~= MSE \* inverse( J^(T)\* J)

covp = MSE\*inv(transpose(J)\*J);

% Calculation of standard deviations of the 5 parameters

% standard deviation = sqrt(diagonal elements)

SE\_V\_max=sqrt(covp(1,1));

SE\_K\_m=sqrt(covp(2,2));

SEpar=[V\_max SE\_V\_max; K\_m SE\_K\_m];

SEpar=full(SEpar);

save('parest\_SE.txt', 'SEpar','-ascii','-double')

%function with estimated parameters

S=0:0.1:data(end,1);

v=V\_max\*S./(K\_m+S);

%Mogelijke waarden voor Vmax en Km

X=(V\_max-10\*SE\_V\_max:SE\_V\_max:V\_max+10\*SE\_V\_max);

Y=(K\_m-10\*SE\_K\_m:SE\_K\_m:K\_m+10\*SE\_K\_m);

%berekening SSE voor elke combinatie van Vmax en Km

Z=zeros(length(Y),length(X));

for k = 1:length(X)

for l = 1:length(Y)

S=data(:,1);

v=X(k)\*S./(Y(l)+S);

for m = 1:length(S)

I(m,1)=(data(m,2)-v(m,1))^2;

end

Z(l,k)=sum(I(:,1));

% m=1;

end

%l=1;

end

%Contour plot

figure(1)

[C,h]=contour(X,Y,Z,20);

set(h,'ShowText','on','TextStep',get(h,'LevelStep')\*4)

xlabel('Vmax')

ylabel('Km')

title('Contourplot Michaelis Menten')

%Response surface plot

figure(2)

surf(X,Y,Z)

xlabel('Vmax')

ylabel('Km')

title('Response surface plot Michaelis Menten')

**CostC**

function C=costC(x,data)

V\_max=x(1);

K\_m=x(2);

S\_data=data(:,1);

V\_data=data(:,2);

v=V\_max\*S\_data./(K\_m+S\_data);

C=V\_data-v;

end

# Opgave B

Fit de parameters van de Lineweaver-Burk vergelijking voor enzymkinetiek.

Vergelijking: 1/V=1/Vmax+(Km/Vmax).1/S

Input: DataMichaelisMenten1 met voor verschillende substraatconcentraties (kolom 1) de enzymactiviteit (kolom 2). Zet dus eerst S en V om naar 1/S en 1/V.

Gevraagd: Gebruik dezelfde intiële waarden van de parameters als in vorige opgave.

Je schrijft een ‘cost’-functie die het verschil tussen de experimentele waarden en de gesimuleerde output berekent. Je gebruikt de lsqnonlinfunctie voor het bekomen van de gefitte parameters.

Output:

* Geschatte parameters met standaardafwijking.
* Plot met data & modelsimulatie van V in functie van S.
* Upload de programma's, de outputfile met de parameters en de figuur.
* Vergelijk de bekomen parameters met die bekomen in opgave 3A.

**Hoofdprogramma B**

close all

clear all

data=load('DataMichaelisMenten1.txt');

S\_data=1./data(:,1);

V\_data=1./data(:,2);

V\_max=3; %mM/min

K\_m=0.2; %mM

x0 = [V\_max K\_m];

lb=[0 0]; %lower bounds

ub=[5 5]; %upper bounds

% input lsqnonlin

options=optimset('MaxFunEvals',200000,'Tolfun',1e-14,'MaxIter',30000);

% optimisation met lsqnonlin

[x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(@costB,x0, lb, ub,options,data);

%output lsqnonlin

V\_max=x(1);

K\_m=x(2);

save('geoptimaliseerde parameters B', 'x');

%modelsimulatie

v=1/V\_max+(K\_m/V\_max)\*S\_data;

V=1./v;

%enzymactiviteit plotten (V in functie van S)

plot(S\_data,V\_data, 'o') % experimentele data

hold on

plot(S\_data,V,'-') %geoptimaliseerd model

hold on

title('Lineweaver-Burk')

xlabel('substraatconcentratie (M) [1/s]')

ylabel('enzymactiviteit (M/min) [1/v]')

legend('experimentele data', 'geoptimaliseerd model')

saveas(gcf,'figureB.eps','psc2')

**CostB**

function C=costB(x,data)

V\_max=x(1);

K\_m=x(2);

S\_data=1./data(:,1);

V\_data=1./data(:,2);

v=(1/V\_max)+(K\_m/V\_max)\*S\_data;

V=1./v;

C=V-V\_data;

end

# Opgave D

Zoals in opgave 3A, fit je de parameters van het Michaelis Menten model voor enzymkinetiek.

Input: Ditmaal gebruik je echter twee datasets, namelijk DataMichaelisMenten1 en DataMichaelisMenten2 voor het schatten van de beste parameters voor alle meetwaarden samen. Gevraagd: Schrijft een ‘cost’-functie die het verschil tussen de experimentele waarden en de gesimuleerde output voor alle data berekent. Output: Geschatte parameters met standaardafwijking. Plot met data & modelsimulatie van V in functie van S. Upload de programma's, de outputfile met de parameters en de figuur.

**Hoofdprogramma D**

data1=load('DataMichaelisMenten1.txt');

Smax1=data1(end,1);

L1=length(data1);

data2=load('DataMichaelisMenten2.txt');

Smax2=data2(end,1);

L2=length(data2);

% bepalen van de hoogste substraatconcentratie voor het plotten van de data

Smax=max(Smax1,Smax2);

L12=max(L1,L2);

%parameterschatting met gebruik van data1 en data2

x0=[0.5 0.5];

lb=[0 0];

ub=[data1(end,2)\*2 100];

[x,resnorm,residual,~,~,~,jacobian]=lsqnonlin(@(x)Matlab3D\_MMcost(x,data1,data2),x0);

%opslaan van de parameters

Vmax12=x(1);

Km12=x(2);

%berekenen van de standaardafwijking op de parameters

SLE = resnorm;

J = jacobian;

NumberOfParameters = length(x);

NumberOfData=length(data1)+length(data2);

MSE = SLE/(NumberOfData - NumberOfParameters);

covp = MSE\*inv(transpose(J) \* J);

SE\_Vmax12 = sqrt(covp(1,1));

SE\_Km12 = sqrt(covp(2,2));

%plotten van figuur met data, simulatie met geschatte parameters en boven

%en ondergrenzen van 95% betrouwbaarheidsinterval

figure(1)

S=0:0.01:Smax;

V=Vmax12\*S./(Km12+S);

plot(S,V,data1(:,1),data1(:,2),'bo',data2(:,1),data2(:,2),'b^')

**Cost D**

function f=costMM12(x,data1,data2)

Vmax=x(1);

Km=x(2);

S=data1(:,1);

V1=Vmax\*S./(Km+S);

S=data2(:,1);

V2=Vmax\*S./(Km+S);

f=[data1(:,2)-V1(:);data2(:,2)-V2(:)];

end

Matlabsessie 4

# Opgave A

In een voorgaande Matlabsessie heb je DataMichaelisMenten1 en DataMichaelisMenten2 samen gebruikt om de parameters van het Michaelis Menten model te schatten.

Creëer een plot van deze data, het gesimuleerde model en het 95% betrouwbaarheidsinterval. Hiervoor gebruik je een Monte Carlo simulatie met 1000 iteraties.

Valideer nu het model voor de meetgegevens **DataMichaelisMenten3** door de nieuwe experimentele data weer te geven in de voorgaande plot. Vallen al deze data binnen de grenzen van het interval, dan is het model gevalideerd.

Upload in Blackboard:

* Programma
* Figuur met duidelijke weergave van de drie sets van meetresultaten, het gesimuleerde model en het betrouwbaarheidsinterval. Zorg ervoor dat het op de figuur duidelijk is welke data werden gebruikt voor modelleren en welke voor valideren.

**Hoofdprogramma A**

clear all

close all

clc

data1=load('DataMichaelisMenten1.txt');

Smax1=data1(end,1);

L1=length(data1);

data2=load('DataMichaelisMenten2.txt');

Smax2=data2(end,1);

L2=length(data2);

data3=load('DataMichaelisMenten3.txt');

Smax3=data3(end,1);

L3=length(data3);

% bepalen van de hoogste substraatconcentratie voor het plotten van de data

Smax=max([Smax1 Smax2 Smax3]);

L12=max(L1,L2);

%parameterschatting met gebruik van data1 en data2

x0=[0.5 0.5];

lb=[0 0];

ub=[data1(end,2)\*2 100];

[x,resnorm,residual,~,~,~,jacobian]=lsqnonlin(@(x)Matlab4A\_MMcost(x,data1,data2),x0);

%opslaan van de parameters

Vmax12=x(1);

Km12=x(2);

%berekenen van de standaardafwijking op de parameters

SLE = resnorm;

J = jacobian;

NumberOfParameters = length(x);

NumberOfData=length(data1)+length(data2);

MSE = SLE/(NumberOfData - NumberOfParameters);

covp = MSE\*inv(transpose(J) \* J);

SE\_Vmax12 = sqrt(covp(1,1));

SE\_Km12 = sqrt(covp(2,2));

%Monte Carlo simulatie

iteratie=1000;

for i=1:1:iteratie

%1000 combinaties van de 2 parameters binnen de normaalverdeling

%selectie parameter uit normaalverdeling

% Vmax12

Vmax=normrnd(Vmax12,SE\_Vmax12);

while Vmax<(Vmax12-2\*SE\_Vmax12) | Vmax>(Vmax12+2\*SE\_Vmax12)

Vmax=normrnd(Vmax12,SE\_Vmax12);

end

Vmax\_iteratie(i)=Vmax;

% Km12

Km=normrnd(Km12,SE\_Km12);

while Km<(Km12-2\*SE\_Km12) | Km>(Km12+2\*SE\_Km12)

Km=normrnd(Km12,SE\_Km12);

end

Km\_iteratie(i)=Km;

% met de twee geselecteerde parameters wordt een curve gesimuleerd

S=0:0.01:Smax;

V(:)=Vmax\_iteratie(i)\*S./(Km\_iteratie(i)+S);

% een matrix MC met i=1000 rijen en een kolom voor elke S-waarde in [0 Smax]

% wordt gecreÍerd.

MC12(i,:)=V(:);

end

% sorteren van rijen met V-waarden

S=0:0.01:Smax;

LS=length(S);

Y=sort(MC12);%sorteert elke kolom van de matrix in stijgende volgorde

MCmin12=Y(1,:);%minimum waarden op rij 1 in matrix

MCmax12=Y(iteratie,:);%maximum waarden op rij 1000

%plotten van figuur met data, simulatie met geschatte parameters en boven

%en ondergrenzen van 95% gezamenlijk betrouwbaarheidsinterval

figure(1)

S=0:0.01:Smax;

V=Vmax12\*S./(Km12+S);

plot(data1(:,1),data1(:,2),'ko',data2(:,1),data2(:,2),'ks',...

S,V,S,MCmin12,'k--',data3(:,1),data3(:,2),'r\*',S,MCmax12,'k--')

hold on

legend('data1','data2','simulatie op basis van data1 en data2',...

'gezamenlijk betrouwbaarheidsinterval voor data1 en data2','Location','southeast','data3')

**CostA**

function f=costMM12(x,data1,data2)

Vmax=x(1);

Km=x(2);

S=data1(:,1)

V1=Vmax\*S./(Km+S)

S=data2(:,1)

V2=Vmax\*S./(Km+S)

f=[data1(:,2)-V1(:);data2(:,2)-V2(:)]

end

# Opgave B

Gegeven de experimentele data **datafileB** voor een fermentatie waarbij de celconcentratie gemeten werd in functie van de tijd. Zoals eerder kunnen deze data beschreven worden met behulp van het model van Baranyi en Roberts (1994) dat een lagfase, exponentiële fase en stationaire fase beschrijft.

Een vereenvoudiging van dit model kan gebruikt worden indien de lagfase verwaarloosbaar is. Ga na of het over genestelde of niet-genestelde modellen gaat.

Gebruik de formules voor discriminatie van modellen op basis van SSE om na te gaan welke van de twee modellen de meetgegevens het best beschrijft.

TIP1: ook het gebruik van 'finv' voor de F-test zal nodig zijn.

TIP2: gebruik de eerder geleerde methode om SSE (ook SLE genoemd) te bepalen. Er is maar een manier waarop de som van de **kleinste** kwadraten bepaald werd.

Upload in Blackboard

* Programma
* Figuur met meetgegevens en simulatie van volledig model en vereenvoudigd model.
* Een bestand met de informatie en het resultaat van de modeldiscriminatie. Dit bestand mag in Word, dit mag een printscreen zijn van jouw output of het mag een matlabbestand zijn.
* Als inhoud wordt T, F en de conclusie verwacht.

Model van Baranyi en Roberts (1994) dat een lagfase, exponentiële fase en stationaire fase beschrijft.





Een vereenvoudiging van dit model kan gebruikt worden indien de lagfase verwaarloosbaar is.



**functie bar\_dyn**

function dydt=bar\_dyn(t,y,mumax, nmax)

dydt=zeros(2,1);

dydt(1)= mumax\*(1-exp(y(1)-nmax))\*(y(2)./(y(2)+1));

dydt(2)= mumax\*y(2);

end

**functie bar\_dynminlag**

function dydt=bar\_dynminlag(t,y,mumax, nmax)

dydt=zeros(1,1);

dydt(1)= mumax\*(1-exp(y(1)-nmax));

end

**functie cost**

function f=cost(x,data)

%input

n0 = x(1);

q0 = x(2);

mumax = x(3);

nmax = x(4);

%solving differential equation for simulation with given parameters

%starting values for integration

y0=[n0 q0];

%time scale

tspan = data(:,1);

%call integration function

[t,y]=ode45(@Matlab4B\_bar\_dyn,tspan,y0,foptions,mumax,nmax);

%analyse goodness of fit by calculation of error between simulated values

%and measurements

f=y(:,1)-data(:,2);

end

**functie costminlag**

function f=costminlag(x,data)

%input

n0 = x(1);

q0 = x(2);

mumax = x(2);

nmax = x(3);

%solving differential equation for simulation with given parameters

%starting values for integration

y0=n0;

%time scale

tspan = data(:,1);

%call integration function

[t,y]=ode45(@Matlab4B\_bar\_dynminlag,tspan,y0,foptions,mumax,nmax);

%analyse goodness of fit by calculation of error between simulated values

%and measurements

f=y(:,1)-data(:,2);

end

**Hoofdprogramma B**

clear all

close all

format long e

load datafileB.txt;

datafileB(:,3)=log(datafileB(:,2));

data=[datafileB(:,1) datafileB(:,3)];

L=length(data);

%%%%%%%%%%%%%%%%%%%%%

% MODEL G

%%%%%%%%%%%%%%%%%%%%%

% add initial estimates for the parameters x0

n0=data(1,2);

q0=1.0;

mumax=1;

nmax=1;

x0 = [n0 q0 mumax nmax];

par\_G=length(x0);

% lower bounds on the parameter values lb

% upper bounds on the parameter values ub

lb = [1 0 0 0];

ub = [20 25 10 22];

[x,resnorm]=lsqnonlin(@(x)Matlab4B\_cost(x,data),x0,lb,ub);

%save fitted parameters

n0=x(1);

q0=x(2);

mumax=x(3);

nmax=x(4);

SSE\_G=resnorm;

%simulate optimized model with estimated parameters

%add starting parameters for integration

y0=[n0 q0];

tspan=data(:,1);

%integration

[t1,y1]=ode45(@Matlab4B\_bar\_dyn,tspan,y0,foptions,mumax,nmax);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%MODEL F (zonder lag)= subhypothese G

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x0 = [n0 mumax nmax];

par\_F=length(x0);

lb = [1 0 0];

ub = [20 10 22];

[x,resnorm]=lsqnonlin(@(x)Matlab4B\_costminlag(x,data),x0,lb,ub);

n0=x(1);

mumax=x(2);

nmax=x(3);

SSE\_F=resnorm;

y0=[n0];

tspan=data(:,1);

[t2,y2]=ode45(@Matlab4B\_bar\_dynminlag,tspan,y0,foptions,mumax,nmax);

%discriminatie tussen model F, subhypothese van model G.

T=((SSE\_F-SSE\_G)/SSE\_G)\*((L-par\_G)/par\_F);

F = finv(1-0.05,par\_F,L-par\_G); % eenzijdige F-test voor alfa=0.05

if T<F

input('T<F, Model F is een subhypothese van Model G. Druk hier enter voor de figuur')

else

input('T>=F, Model F is geen subhypothese van Model G. Druk hier enter voor de figuur.')

end

% plot experimental data

plot(data(:,1),data(:,2),'o','MarkerSize',8);

hold on

plot(t1,y1(:,1),'-','linewidth',1.5);

hold on

plot(t2,y2(:,1),'-','linewidth',1.5);

hold on

set(gca,'LineWidth',1.5,'fontsize',18)

xlabel('time [h]','FontSize',18)

ylabel('ln (N) [ln(cfu/mL)]','FontSize',18)

grid on

hold on

Matlabsessie 5

# Opgave A

Aerobe degradatie van een organische component door een gemengde cultuur van organismen in afvalwater kan door volgende reactie voorgesteld worden:

C3H6O3 + a O2 + b NH3 → c C5H7NO2 + d H2O + e CO2

Schrijf de vergelijkingen voor bepalen van a, b, c, d en e als Yx/s = 0,4 gX/gS

Schrijf een matlabprogramma voor het bepalen van de stoichiometrische coëfficiënten door oplossen van het stelsel van vergelijkingen bekomen in de oefening.

HINT: Symbolen ingeven doe je met ‘syms’. Maak gebruik van een matrix met behulp van ‘linsolve’. Door gebruik te maken van EquationsToMatrix kan je rechtstreeks de vergelijkingen ingeven.

close all

clear all

syms a b c d e;

eqn(1)=5\*c+e==3;

eqn(2)=6+3\*b==7\*c+2\*d;

eqn(3)=3+2\*a==2\*c+d+2\*e;

eqn(4)=b==c;

eqn(5)=c==0.4;

%Use equationsToMatrix to convert the equations into the form AX = B

%The second input to equationsToMatrix specifies the independent variables in the equations.

[A,B]=equationsToMatrix(eqn, [a, b, c, d, e])

%Use linsolve to solve AX = B for the vector of unknowns X.

X=linsolve(A,B)

# Opgave B

In steady state is de fractie van de componenten C1-5 constant. De opname van de substraten S1-2 en de concentratie P wordt gemeten (details in Provost en Bastin 2004). De vergelijkingen zijn:

dC1/dt = v1-v4-v8==0

dC2/dt = v2-v6==0

dC3/dt = v3-v8==0

dC4/dt = v4-v5==0

dC5/dt = v4+v5+v6-v7==0

S1: v1==0.2;%specifieke opnamesnelheid substraat1

S2: v2+v3==0.2;%specifieke opnamesnelheid substraat2

P: v7==0.3; % specifieke productvormingssnelheid

Matlab kan dit op een relatief eenvoudige manier oplossen.

Definieer v1; v2… als variabelen, gebruik het commando ‘syms’. Breng de variabelen in een vector v.

Vervolgens definieer je de vergelijkingen met ‘eqn’. Creëer een reactiematrix P en de vector met de reactiesnelheden r door gebruik te maken van het commando ‘equationsToMatrix’.  Los de vergelijkingen op met gebruik van ’linsolve’ van P en r. Breng de oplossing, P en r onder matrixvorm met gebruik van het commando ’double’.

close all

clear all

syms v1 v2 v3 v4 v5 v6 v7 v8;

v=[v1 v2 v3 v4 v5 v6 v7 v8];

eqn(1)=v1-v4-v8==0;

eqn(2)=v2-v6==0;

eqn(3)=v3-v8==0;

eqn(4)=v4-v5==0;

eqn(5)=v4+v5+v6-v7==0;

eqn(6)=v1==0.2; %specifieke opnamesnelheid substraat 1

eqn(7)=v2+v3==0.2; %specifieke opnamesnelheid substraat 2

eqn(8)=v7==0.3; %specifieke productvormingsssnelheid

[A,B]=equationsToMatrix(eqn,v);

X=linsolve(A,B);

P=double(A) %reatiematrix P

r=double(B) %reactiesnelheden r

oplossing=double(X)