UPPSALA UNIVERSITET

Miniprojekt 2
Beräkningsvetenskap 2
Fredrik Gustafsson F2C
2017-05-05

May 5, 2017

1 Introduktion

De flesta problem som vi stöter på kan modeleras med ODE eller PDE eller andra differentialekvationer, dessa kan sedan lösas med matlabs funktioner som ode45. För vissa problem är dock de olika runge-kutta metoderna mindre bra. Exempel på sådanna problem är kemiska problem där man inte vet vilken reaktion som kommer ske eller när den kommer ske. För att hantera sådana problem kan en monte carlo metod vara att föredra.

2 Teori

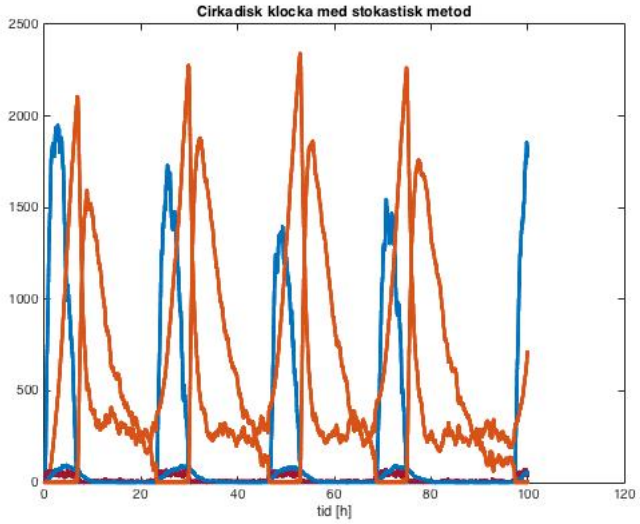
En stokastisk metod fungerar i principen att man slumpar fram ett stort antal punkter och undersöker vad funktionsvärdet är i dessa punkter för att sedan ta ett medelvärde av dessa diskreta punkter. I t.ex. biokemi kan detta vara att föredra då det är en slump om och vilka proteiner som reagearar med varandra. I ett sådant fall är kontinuerliga lösningsmetoder dåliga då dessa kan räkna med halva proteiner, vilket kan fungera i höga koncentrationer men ej i låga, vilket då inte kommer fungera.

3 Metod

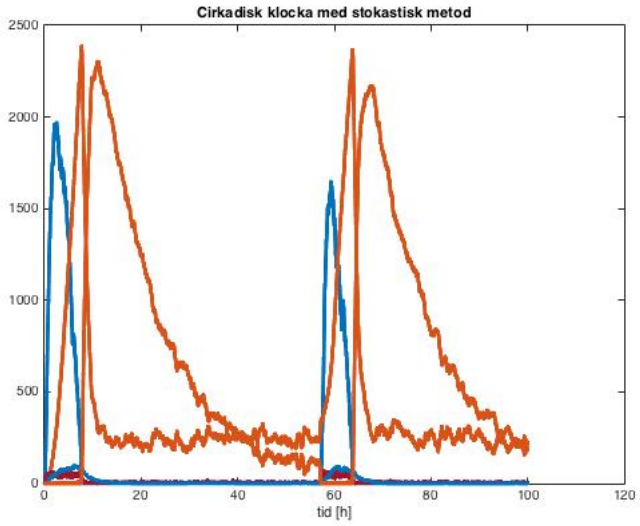
För att använda en stokastisk metod för att lösa ett kemiskt problem behövs en stökiometri matris. Den samt propensiteterna för problemet var givna i egna filer. Programet hämtar då startvärden för dessa från filerna och går sedan in i en while-loop. I denna loop slumpas två slimptal fram. Det första används för att bestämma Tao genom inverterad kumulativ densitetsfunktion som använder det första slumptalet. Detta ritas in i ett diagram tiden ändras med Tao och sedan upprepas processen med nya reaktioner som då stegvis ritas in i diagramet till önskad tid.

4 Resultat

Figur 1 visar den cirkadiska klockan med $\delta_R = 0.2$ beräknad med koden som bifogas i appendix. Figur 2 har samma beräkningar gjorts men med $\delta_R = 0.08$, återigen samma kod.



Figur 1 cirkadisk klocka med $\delta_R = 0.2$



Figur 2 cirkadisk klocka med $\delta_R = 0.08$

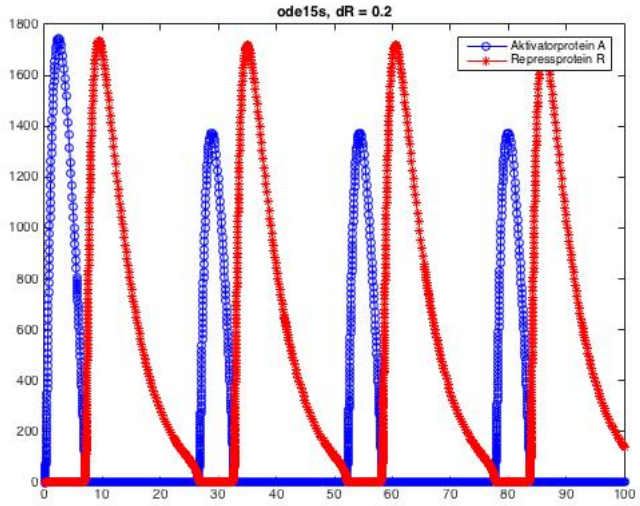
5 Diskussion

Om man jämför figur 1 med figur 3, som visar lösningen med matlabs ode15 så kan man se att dessa verkar stämma överens. Från detta kan man då dra slutsatsen att stokastiska metoder kan användas för att lösa ett sådant här problem.

Man kan även se att med $\delta_R = 0.08$, figur 2, fortsätter oscillationerna trots detta. För samma störning beräknad med ode15 kan man se att detta inte är fallet, figur 4. Detta i sin tur visar att den stokastiska metoden är mindre känslig för störningar än vad ode metoden är. Detta har att göra med hur metoderna gör beräkningarna. Om initialdatan störs för ode metoden så kommer felet påverka alla efterföljande tal. Den stokastiska metoden räknar fram koncetrationen med en slumpad reaktion. När nästföljande reaktion slumpas fram kommer kurvan börja falla in mot den kurvan som ges från det ickestörda problemet.

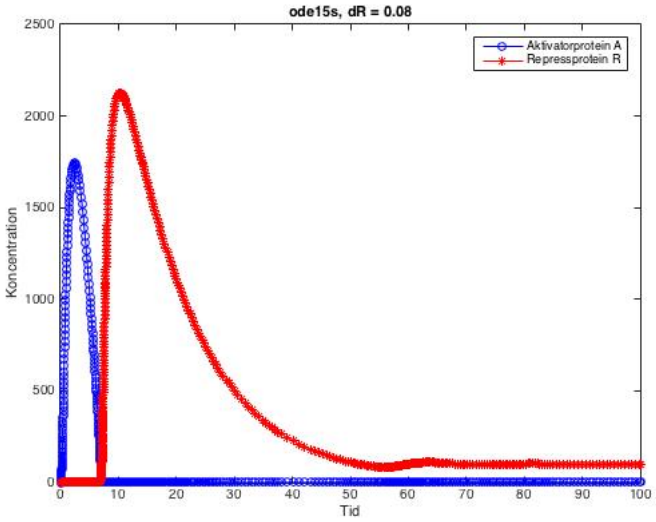
6 Appendix

Figur 3 simulering utan störning med diffekvationslösare.



Figur 3 simulering med $\delta_R = 0.2$ med diffekvationslösare.

Figur 4 simulering med $\delta_R = 0.08$ med diffekvationslösare



Figur 4 simulering med störning med diffekvationslösare

Matlab Kod:

```
function nr = nr_vilar( )% nr = nr_vilar()
%
% Stoichiometry matrix, nr, for the vilar oscillator.
% The variables (corresponding to the columns in nr) are ordered as:
% A C D_A D_A' D_R D_R' M_A M_R R
%
% See page 21 (formula 20) in Hellander: Stochastic Simulation and
% Monte Carlo Methods

nr = zeros(18,9);

nr(1,:) = [ 0 0 1 -1 0 0 0 0 0];
nr(2,:) = [-1 0 -1 1 0 0 0 0 0];
nr(3,:) = [1 0 0 0 1 -1 0 0 0];
nr(4,:) = [-1 0 0 0 -1 1 0 0 0];
nr(5,:) = [0 0 0 0 0 0 0 1 0];
nr(6,:) = [0 0 0 0 0 0 0 1 0];
nr(7,:) = [0 0 0 0 0 0 0 -1 0];
nr(8,:) = [0 0 0 0 0 0 1 0 0];
nr(9,:) = [0 0 0 0 0 0 1 0 0];
nr(10,:) = [0 0 0 0 0 0 -1 0 0];
nr(11,:) = [0 0 0 0 0 0 0 0 1];
nr(12,:) = [0 0 0 0 0 0 0 0 -1];
nr(13,:) = [0 -1 0 0 0 0 0 0 1];
nr(14,:) = [1 0 0 0 0 0 0 0 0];
nr(15,:) = [1 0 0 0 0 0 0 0 0];
nr(16,:) = [1 0 0 0 0 0 0 0 0];
nr(17,:) = [-1 0 0 0 0 0 0 0 0];
nr(18,:) = [-1 1 0 0 0 0 0 0 -1];
```

```
function w = prop_vilar(u, p) %
% w = prop_vilar(u, p)
% Propensities, w, for the Vilar oscillator.
%
% Input: u - the current state.
% p - list of parameters
%
% The current state variables (u) are ordered as:
% A C D_A D_A' D_R D_R' M_A M_R R
% The parameters (in p) are ordered as:
% alfa_A alfa'_A alfa_R alfa'_R beta_A beta_R teta_A teta_R ...
% gamma_A gamma_R gamma_C delta_M_R delta_M_A delta_A delta_R
%
```

```
alfaA = p(1); alfapA = p(2);
alfaR = p(3); alfapR = p(4);
betaA = p(5); betaR = p(6);
tetaA = p(7); tetaR = p(8);
gammaA = p(9); gammaR = p(10);
gammaC = p(11);
deltaMR = p(12); deltaMA = p(13);
deltaA = p(14);
deltaR = p(15);

w = zeros(18,1);
w(1 ) = tetaA*u(4);
w(2 ) = gammaA*u(1)*u(3);
w(3 ) = tetaR*u(6);
w(4 ) = gammaR*u(5)*u(1);
w(5 ) = alfapR*u(6);
w(6 ) = alfaR*u(5);
w(7 ) = deltaMR*u(8);
w(8 ) = alfapA*u(4);
w(9 ) = alfaA*u(3);
w(10) = deltaMA*u(7);
w(11) = betaR*u(8);
w(12) = deltaR*u(9);
w(13) = deltaA*u(2);
w(14) = betaA*u(7);
w(15) = tetaA*u(4);
w(16) = tetaR*u(6);
w(17) = deltaA*u(1);
w(18) = gammaC*u(1)*u(9);



    Huvudprogram
    clear all
x = [0 0 1 0 1 0 0 0 0 ]; % A C D_A D_A D_R D_R mRNA_A mRNA_R R
t = 0; %Starttid
Tf = 100; %sluttid n = nr_vilar();

parametrar = [50 500 0.01 50 50 5 50 100 1 1 2 0.5 10 1 0.08];

Tid =[t];
Result = [x];


    while t < Tf
w_r= pro p_vilar(x, parametrar);
```

```
a0 = sum(w_r);

u1 = rand;
u2 = rand;


Tao = log(1-u1)/(-a0);
p = w_r./a0;

    r = find(cumsum(p) > u2, 1);

    x = x + n(r,:);
t = t+Tao;

Result = [Result; x];
Tid = [Tid; t];
end


plot(Tid, Result, '.')
title('Cirkadisk klocka med stokastisk metod');
xlabel('tid [h]');
```