Modelling with MATLAB: Assignment 3 2019

Interpreting, running, commenting and extending existing code: the Gillespie algorithm to simulate a simple birth-death process

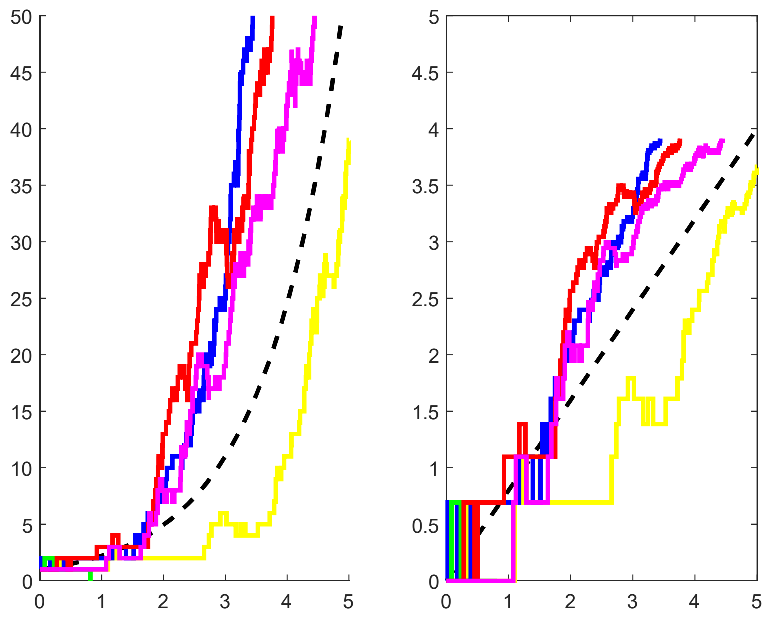
Your solutions should be uploaded to Moodle as a single written document, which may contain graphics and mathematics (but which does not just list your code), and which makes clear links to well-labelled MATLAB files which should be uploaded to Moodle (as .m files) at the same time. Your solutions may be in PDF (e.g. generated via LaTeX), Word, or any other appropriate format, but they must NOT depend on the marker having access to any additional software beyond a PDF reader, Microsoft Word, and a copy of MATLAB. Credit will be given for providing working code, and for providing suitable comments within the code to allow it to be used accurately.

Simply submitting a collection of MATLAB files is not enough.

There are three questions in this assignment, carrying 35, 35 and 20 marks, respectively. Your answers need to be uploaded on Moodle by **1200 on Friday 6th December**, please.

Background: There are two files on Moodle **Week 8**: A3_1_2019.m and A3_2_2019.m. Just like real-world situations, the files contain elements of coding which you may not be familiar with (e.g. the `stairs` command and `while` loops). The files can be used to simulate a Gillespie algorithm implementation of a simple birth-death process.

Download copies of these A3_1_2019.m and A3_2_2019.m, load them into MATLAB, and run A3_1_2019.m. You will see an output similar to this:



1. (a) Rename A3_1_2019.m and A3_2_2019.m appropriately, and add comments to them so that a user can understand what the code is doing. You should rename parameters and/or variables for clarity.

(b) Adapt the code so that the axes in the output graphs are labelled clearly and appropriately.

(c) Provide a brief "user guide" (max. 1 side of A4) for a mathematically literate user to use these files, explaining the inputs they require, the meaning of the key parameters and how these might be changed, and the outputs the code produces.

2. (a) Write down the ODE which is the deterministic counterpart to the simple birth-death process model in the code in Q1. Explain how the parameters and variables in your ODE model relate to the parameters and variables in the code.

(b) Adapt your code so that it can simulate a large number of independent realisations of the model in Q1, with each model running for a fixed time which the user can control, and which records the state of each independent realisation of the model at that fixed time. There is no need for graphical output to be displayed for these simulations.

(c) Compare the means and variances of your outputs from your independent simulations in (b) with the ODE in (a), and comment on any similarities or differences.

(d) Use simulations to estimate the probability that the population modelled in Q1 eventually becomes extinct.

3. Adapt the code in Q1 so that it can simulate a simple birth-death process with the additional constraint that the births are constrained by a logistic limitation:

$$\text{(per capita birth rate when population is } n\text{)} \quad = \quad B \; * \; (1 - (n / C))$$

where B and C are constants and n is any non-negative integer; B is the maximum per capita birth rate and C represents the logistic limitation to birth.

Illustrate the behaviour of this revised model, and calculate its long-term mean numerically.

Explain briefly why this long term mean should be described as a "quasi equilibrium state" (rather than an "equilibrium state").

```matlab
% A3_1_2019.m

clear all
init=1;
aa=1.5; bb=0.7;
%aa=1.2; bb=0.4;
%aa=0.8; bb=0.0;

t=[0:.1:5];
pp=init*exp((aa-bb).*t);
subplot(1,2,1)
hold on
plot(t,pp,'k--','Linewidth',2);
axis([0,5,0,50]);
subplot(1,2,2)
plot(t,log(pp),'k--','Linewidth',2);
axis([0,5,0,5]);

hold on

for k=1:5
clear t y
y(1)=init;
t(1)= 0;
j=1;
while y(j)>0 & y(j)<50
u=rand(1,2);
t(j+1)=-log(u(1))/(aa*y(j)+bb*y(j))+t(j);
if u(2)<aa/(aa+bb)
y(j+1)=y(j)+1;
else
y(j+1)=y(j)-1;
end
j=j+1;
end

A3_2_2019(t,k,y);

end
```

```matlab
% A3_2_2019.m

function F = A3_2_2019(A,B,C)

subplot(1,2,1)
hold on
if B==1
stairs(A,C,'y-','Linewidth',2);
end
if B==2
stairs(A,C,'b-','Linewidth',2);
end
if B==3
stairs(A,C,'g-','Linewidth',2);
end
if B==4
stairs(A,C,'r-','Linewidth',2);
end
if B==5
stairs(A,C,'m-','Linewidth',2);
end

subplot(1,2,2)
hold on
if B==1
stairs(A,log(C),'y-','Linewidth',2);
end
if B==2
stairs(A,log(C),'b-','Linewidth',2);
end
if B==3
stairs(A,log(C),'g-','Linewidth',2);
end
if B==4
stairs(A,log(C),'r-','Linewidth',2);
end
if B==5
stairs(A,log(C),'m-','Linewidth',2);
end
```