Computer labs

May 10, 2018

A list of matlab tutorials can be found under
http://snovit.math.umu.se/Personal/cohen_david/teachlinks.html

Task 1: The following MATLAB code generates (pseudo) uniform random variables

```
clear all
rand('state',100) % set the state of Matlab's function rand
U=rand
m=2;n=2;
UU=rand(m,n)
```

The above code returns a standard uniform random variable U (on the interval $(0,1)$) and an $m \times n$ matrix UU containing independent uniform random variables. Further information can be found under https://se.mathworks.com/help/matlab/ref/rand.html. Play a little bit with this command to get familiar with.

The goal of the first task is to simulate a toss of a coin with probability of heads p using uniform random variables generated by MATLAB.

Let U be a uniform random variable on $(0,1)$ and $0 < p < 1$. Consider the random variable X defined by

$$X = \begin{cases} 1 & \text{if} \quad U < p \\ 0 & \text{if} \quad U \geq p. \end{cases}$$

Show (by paper and pen) that $\mathbb{P}(H) := \mathbb{P}(\text{"Head"}) = \mathbb{P}(X = 1) = p$ (and also that $\mathbb{P}(T) := \mathbb{P}(\text{"Tail"}) = \mathbb{P}(X = 0) = 1 - p$). To do this, use the formula, seen in the lecture, that allows you to compute probabilities using a PDF.

Complete the following MATLAB code to simulate two toss of a fair coin ($p = 1/2$)

```
clear all
rand('state',100) % set the state of rand
p= ... ;
U= ...
X=( ... <p) % first simulation
U= ...
X=( ... <p) % second simulation
```

Task 2: The goal of this task is to simulate normal random variables using uniform random variables as seen in the previous task. To do so, we shall use the so-called Box-Muller transformation (https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform) which generates a pair of independent random variables (Z_1, Z_2) by transforming a pair of independent uniform random variables (U_1, U_2) using the formula

$$Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)$$
$$Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2).$$

To illustrate the fact that Z_1 and Z_2 are normally distributed, we first generate $n = 5000$ pairs of such normal random variables. We then plot the histograms of Z_1 and Z_2. The following can be used to start this task

```
clear all
rand('state',123) % set the state of rand
n=5000;
UU=rand(n,2); % n pairs of unif. RV
ZZ= ... ; % pairs of standard normal RV using Box-Muller
% plot the histograms
...
histogram( ... ,'Normalization','pdf')
legend(...)
...
```

Feel free to increase n and compare the histograms with the true probability density function of a standard normal random variable. This can be done using the definition of a Gaussian function or the MATLAB command PDF (if you have the *Statistics and Machine Learning Toolbox*).

What does the MATLAB function RANDN do? Can you use this function in your above code?

Task 3: The goal of this task is to simulate a Brownian motion/Wiener process $(W(t))_{t\in[0,1]}$.

(a) Write a Matlab code to simulate one realisation of a discretised Brownian motion on $[0,1]$ for different values of Δt. Consider grid points given by $t_m = m\Delta t$, where $\Delta t = 2^{-4}, 2^{-6}$, and 2^{-8}. Use the definition of a Brownian motion to compute $W(0)$, $W(\Delta t)$, $W(2\Delta t)$, etc. Plot your numerical results.

You may use the following

```
clear all
randn('state',100) % set the state of randn
% discretised BM for dt=2^(-4)
Tend= ... ;dt= ... ;N= ...;
W(1)= ...; % BM starts at 0 a.s.
for m=...
  dW=sqrt(dt)*randn(1,1); % Wiener increment/normal rand. var.
  W(m+1)= ... ; % iter. procedure using def. of BM
end
% plot W against time
figure(),
plot([0:dt:Tend],W,'b','LineWidth',3)
...
% repeat the above for another step size dt
...
```

(b) With the help of the above part, compute the mean of $W(t)$ over 20000 trajectories of $W(t)$ on $[0,1]$ with $\Delta t = 2^{-8}$. Plot your results.

```
...
dW=sqrt(dt)*randn(M,N+1); % gen. M samples of Wiener increments
...
Wmean=mean(...);
...
```

(c) Finally, in the same figure, display 5 sample paths of $W(t)$ and the mean over 50000 trajectories of $W(t)$ on $[0,1]$ for $\Delta t = 2^{-8}$.

Task 4: The goal of this task is to simulate a stochastic process.

(a) Write a Matlab code to simulate one realisation of the discretised stochastic process (gBM) $X(t) = X_0 \exp((\mu - \frac{1}{2}\sigma^2)t + \sigma W(t))$ for $t_m = m\Delta t$, where $\mu = 2, \sigma = 1, X_0 = 1$ on $[0,1]$ with $\Delta t = 2^{-8}$. Plot your numerical results. Observe that you may use the previous task to generate the Wiener process $W(t)$.

```
...
X(1)=X0; % the process X starts at X0
for m=1:N
  ...
  % iter. to compute the process X, where W is Wiener proc.
  X(m+1)=X0*exp((mu-0.5*sigma^2)*m*dt+sigma*W(m+1));
end
...
```

(b) With the help of the above part, compute the mean of $X(t)$ on $[0,1]$ over 20000 trajectories of $X(t)$ with $\Delta t = 2^{-8}$. Can you guess (more or less) the value of $\mathbb{E}[X(1)]$?

(c) Finally, in the same figure, display 5 sample paths of $X(t)$ on $[0,1]$ together with the mean over 50000 trajectories of $X(t)$ for $\Delta t = 2^{-8}$.

Task 5: The main motivation for this task is to write a fast code for the simulation of a path of a Wiener process/Brownian motion $W(t)$ for $t \in [0,1]$, avoiding a FOR loop. To do this, we shall use the MATLAB function CUMSUM. Study this function, understand and complete the following piece of code. Compare with your result obtained in **Task 3**.

```
% Simulation of path of WP/BM
clear all
randn(state, ... ); % set state pseudo random number generator
T=1;
dt=2^(-8);
...
t = 0:dt:T; % discrete time grid
dW = ... ; % generate Wiener increments
W = [0 cumsum(dW)]; % Brownian path
% plot the Brownian path
...
```

Task 6: This task illustrates the numerical approximation by Euler-Maruyama's scheme. Consider the SDE (for times $0 \le t \le T$)

$$\begin{aligned} \mathrm{d}X(t) &= \lambda X(t)\,\mathrm{d}t + \mu X(t)\,\mathrm{d}W(t) \\ X(0) &= X_0 \end{aligned}$$

with parameters $X_0 = 1, \lambda = 2, \mu = 1, T = 1$. For this particular problem, we know that the exact solution reads $X(t) = X_0 \exp\left((\lambda - \frac{1}{2}\mu^2)t + \mu W(t)\right)$. Compute a discretised Brownian motion over $[0,1]$ with a very small discretisation parameter $\delta t = 2^{-8}$. This is used to compute our "exact solution" **Xtrue**. Apply Euler-Maruyama's method with 3 different time steps $\Delta t = 2^4\delta t, 2^2\delta t$, and δt. Here, it is important to be on the same discretised Brownian path as the one used for generating the exact solution. In three different figures, display one realisation of the numerical solution together with the exact solution.

```
...
randn('state',100)
% parameters
lambda=2; mu=1; X0=1;
Tend=1; N=2^8; dt=T/N;
dW=sqrt(dt)*randn(1,N);     % Brownian increments
% discretised Brownian path
W=...
% exact solution
Xtrue=X0*exp((lambda-0.5*mu^2)*([dt:dt:T])+mu*W);

%% First EM approx.
R=2^4; Dt=R*dt; L=N/R;    % L EM steps of size Dt = R*dt
Xem=zeros(1,L);       % preallocate for efficiency
for j=1:L
  Winc=sum(dW(R*(j-1)+1:R*j)); % Winc for EM on the same BM as above
  Xem(j)= ... ; % iter. for EM scheme with step Dt
end
% first plot on first figure
...
figure()
plot([0:dt:Tend],[X0,Xtrue],'k-'), hold on
plot([0:Dt:Tend],[X0,Xem],'r--*'), hold off
...
```

The expression **Winc** in the code above is computed as follows

$$\text{Winc} = W(j\Delta t) - W((j-1)\Delta t) = W(jR\delta t) - W((j-1)R\delta t) = \sum_{k=jR-R+1}^{jR} \mathrm{d}W_k,$$

with $\mathrm{d}W_k = W(k\delta t) - W((k-1)\delta t)$ the original Wiener increments.

Task 7: This task asks you to confirm numerically the weak order of convergence of Euler-Maruyama's scheme. Consider the SDE from the previous exercise with parameters $\lambda = 2, \mu = 0.1, X_0 =$

$1, T = T_N = 1$. Verify numerically that

$$e_{\Delta t}^{\text{Weak}} := \left| \mathbb{E}[X_N] - \mathbb{E}[X(t_N)] \right| \leq C\Delta t,$$

where, for the exact solution, one has $\mathbb{E}[X(t_N)] = \mathbb{E}[X(1)] = \mathrm{e}^{\lambda \cdot 1}$. In order to approximate expectations, you may use a Monte-Carlo approximation with $M_s = 50000$ realisations of the numerical solution.

In order to display the results in a loglog plot, one computes (for example) 5 different approximations with Euler-Maruyama's method with step sizes $\Delta t = 2^p \delta t$, where $\delta t = 2^{-10}$ and $p = 1, \ldots, 5$.

```
...
Ms=50000; % number of paths sampled
Xem=zeros(5,1); % preallocate arrays containing num. sol.
for p=1:5 % take various Euler timesteps
  Dt=2^(p-10);L=T/Dt; % L Euler steps of size Dt
  Xtemp=X0*ones(Ms,1); % initial values
  for j=1:L
    Winc=sqrt(Dt)*randn(M,1);
    Xtemp= ... ; % Ms EM approximation samples
  end
  Xem(p)=mean(...) ; % mean over Ms samples of EM at time Tend
end
Xerr=abs( ... - ... ); % weak errors
% loglog plots
Dtvals=2.^([1:5]-10); % array of time steps
figure('units','normalized','outerposition',[0 0 1 1])
loglog(Dtvals, ... ,'ks-',Dtvals, ... ,'r--','LineWidth',2),
xlabel('\Delta t'),
ylabel('Weak Err.','Rotation',0,'HorizontalAlignment', ...
          'right','FontSize',14)
legend('EM','Slope 1','Location','SouthEast');
title('Weak error: EM for geometric BM','FontSize',10)
set(gca,'FontSize',15);
```

Task 8: Consider again the SDE (geometric Brownian motion on $0 \leq t \leq T$)

$$\mathrm{d}X(t) = \lambda X(t)\,\mathrm{d}t + \mu X(t)\,\mathrm{d}W(t)$$
$$X(0) = X_0$$

with parameters $\lambda = 2, \mu = 1, X_0 = 1, T = T_N = 1$. Verify numerically that

$$e_{\Delta t}^{\text{strong}} := \mathbb{E}[|X_N - X(T_N)|] \leq C\Delta t^{1/2},$$

where $X(T_N)$ denotes the exact solution at time $T_N = 1$ and X_N the last step of Euler-Maruyama's method. In order to approximate the expectations, you may use $M_s = 1000$ samples.

In order to display the results in a loglog plot, one computes (for example) 5 different approximations with Euler-Maruyama's method with step sizes $\Delta t = 2^p \delta t$, where $\delta t = 2^{-10}$ and $p = 1,\ldots,5$.

```
...
Xerr=zeros(Ms,5);  % preallocate array error
for s=1:Ms,  % sample over discrete BM
  ...
  dW=sqrt(dt)*randn(1,N); % Brownian increments
  W=cumsum(dW);     % discrete BM
  Xtrue= ...; % exact sol.
  for p=1:5
    R=2^(p);Dt=R*dt;L=N/R; % L Euler steps of size Dt=R*dt
    Xtemp=Xzero;
    for j=1:L
      Winc= ...; % Wiener increm.
      XEM= ...; % EM scheme
    end
    Xerr(s,p)=abs( ... - ... ) ; % error at T=1
  end
end
% compute strong errors+plots
Dtvals=dt*(2.^([0:4]));
figure('units','normalized','outerposition',[0 0 1 1])
loglog(Dtvals,mean( ... ),'ks-',Dtvals,(Dtvals.^( ... )),'r--', ...
          'LineWidth',2),
...
```

Some of the exercises are inspired by materials from E. Allen, D. Higham, H. Pishro-Nik, A. Rakhshan, A. Szepessy.