School of Mathematics and Statistics
MAST30028 Numerical Methods & Scientific Computing
2020

# Assignment 1: Simulation & errors
# Due:17:00 p.m. September 10

Note: This assignment is worth 20% of the total assessment in MAST30028. When you submit the assignment, you should submit two files: one pdf file and one zip file. The pdf file contains the answer you write, the numerical results you generated including figures or tables(0 mark if numerical results are not included in the pdf file), the comment or the explanation of the results, and the printout of your code (you may use Matlab `publish` or Matlab live scripts). The zip file should only contains all the m-files.

*To aid marking, please start Questions 3 and 4 on a new page in your write-up.*



## 1 Traffic [20 marks]



Here is a simple stochastic traffic model that incorporates a few simple rules for drivers to follow, yet predicts features seen in real traffic situations.

The model is discrete in time and space and the car velocities are also discrete. We have $N$ cars distributed along a circular single-lane road $L$ spatial units in length. Each car takes up 1 spatial unit. Cars are allowed to travel a maximum speed $v_{max}$. In what follows, take $v_{max} = 5$.



a. Initially, distribute the cars randomly on the road by sampling the $L$ possible positions without replacement (there is a command in MATLAB to do this). Set the velocity of each car to zero. Record the distances between each car and the car in front [remember the road is circular].

b. At each subsequent timestep, perform the following updates on all cars in this order:

   (i) accelerate: each car increases its velocity by 1 unit if it is not travelling at the maximum speed.

   (ii) avoid collisions: each car checks the distance $d$ to the next car. If its velocity $v \geq d$ then it reduces its speed to $d - 1$ to avoid a possible collision in the next timestep.

   (iii) random braking: if the car is moving, it reduces its speed by 1 unit with probability $p$. This is the only stochastic part of the model.

   You should seed the random number generator with the integer 42 if the logical variable `seed` is `true`. This is to aid marking.

   (iv) update positions (and distances): move all positions forward by $v$ units [remember the road is circular] and update all distances.

c. The model is run for a number $T_r$ timesteps to relax from the initial state, then for a further $T_e$ timesteps in which the positions of each car are recorded at each timestep. It is convenient to record the position of the cars as rows of a matrix `trafficflow` (initially full of zeroes) using a command such as

```
trafficflow(time,position) = 1;
```

where `position` is a row vector recording the position of all the cars at $t =$ `time`.

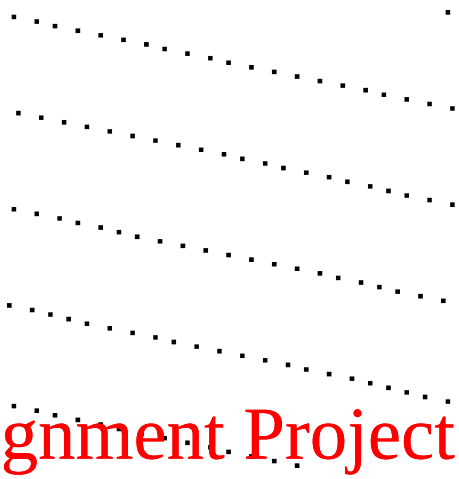
d. Write a MATLAB function to simulate this model, with the first line

```
function trafficflow = trafficSim(L,N,p,vmax,Tr,Te,seed)
```

You can't avoid loops over time but I was able to write the updates in section b above without any loops over cars, by using MATLAB commands operating on vectors and logical indexing. More marks will be given if you can avoid update loops (and the code is correct).

e. Test your code by:

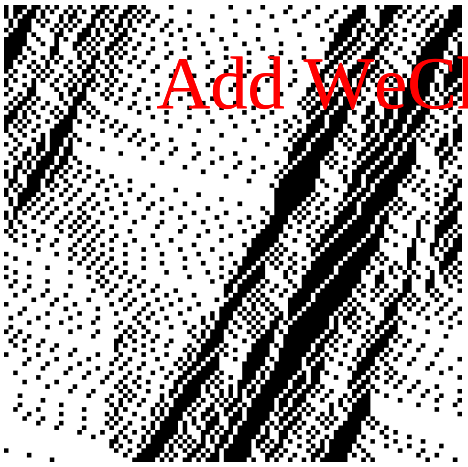
(i) writing a driver to simulate 1 car on a road of length 100, with $p = 0.3$, and `seed = true`. Set $T_r = 250$ and $T_e = 100$. You should get clear traffic





(ii) edit your driver to simulate 85 cars under the same conditions. You should obtain, using `imshow(trafficflow)`, the image below.







f. Finally run your code for a road of length 1000 with $p = 1/3$, and `seed = false`. Set $T_r = 2500$ and $T_e = 1000$. Perform simulations for various values of traffic density $\rho = N/L$, including $\rho = 0.1, 0.15, 0.2, 0.25$. Describe what you see and interpret it in terms of common traffic patterns.

## 2 Hyperspheres [12 marks]

In lectures I mentioned we could estimate $\pi$ by using hit-and-miss Monte Carlo estimation on a unit circle. Here we study the n-dimensional analogue — unit hyperspheres i.e. unit balls in $\mathbb{R}^n$. This allows us to test the abilities of MC methods on higher-dimensional integrals.

We want to estimate the volume of unit hyperspheres in dimensions $n = 2, 5, 8, 11, 14$. To do this, we distribute points in the hypercube $[-1, 1]^n$ randomly, and estimate the probability of landing inside the unit hypersphere, $p$. Then the volume of a hypersphere $S_n$ is given by

$$S_n = 2^n p$$

since $2^n$ is the volume of the hypercube with side length 2.

a. Write a MATLAB function with calling sequence

```
function [estimate, halfWidth] = ballMC(Nreps,n)
```

that uses hit-and-miss Monte Carlo with `Nreps` random points in $\mathbb{R}^n$ to estimate $p$ using the sample proportion $\hat{P}$. It returns the estimate $\hat{p}$ and the half-width of the 95% confidence interval for $p$.

b. Write a driver that, for each dimension in $n = 2, 5, 8, 11, 14$ calls `ballMC` with `Nreps` taking values distributed logarithmically between $10^1$ and $10^6$. The driver should plot

(i) the estimated $S_n$ versus `Nreps` on a suitable plot

(ii) the CI halfwidth (as a measure of statistical error) **for $S_n$** versus `Nreps` on a suitable plot

i.e. 10 plots in all. Include suitable labels, and titles including the value of $n$.



c. The volume $S_n$ is known exactly:

$$S_n = \frac{\pi^{n/2}}{\Gamma(1 + n/2)}$$



Modify your driver so that the plots of statistical error also include the actual error of each estimate. I suggest you use at least 100 values of `Nreps` for each $n$.



d. Based on your plots, answer the following:

- do the Monte Carlo estimates stabilize as `Nreps` increases?
- what happens to the estimated $S_n$ as $n$ increases?
- does the CI halfwidth capture the actual error behaviour?
- what happens to the CI halfwidth for $S_n$ as $n$ increases?
- is there anything unusual in your plots for high $n$ and low `Nreps`?

This example illustrates why one may need to use *variance reduction* techniques in high dimensions.

## 3 A bound on roundoff error [4 marks]

To analyse the roundoff error from $n$ floating point multiplies, the following quantity arises:

$$\Pi_{i=1}^{n}(1+\delta_i) \equiv 1+\theta_n$$

where $|\delta_i| \leq u$ and $u$ is unit roundoff i.e. $\theta_n$ is the relative error after $n$ floating point multiplies.

Use mathematical induction to prove the bound

$$|\theta_n| < \frac{nu}{1-nu}$$

assuming $nu < 1$. The same bound holds (with $\leq$) if we include divides as well, but you do not need to prove this.

## 4 A better derivative. [14 marks]

a. Use Taylor series to derive the truncation error of the approximation

$$f'(x) \approx \frac{-3f(x)+4f(x+h)-f(x+2h)}{2h}$$

assuming $f \in C^3$.

b. Explain why dividing by $h$ produces no roundoff error if $h = 2^{-k}, k \in \mathbb{N}$.

c. Assuming that built-in functions return exact answers but that addition/subtraction produce roundoff errors and that $h = 2^{-k}, k \in \mathbb{Z}$, show that the roundoff error $RE$ in computing this expression satisfies the bound

$$|RE| \leq (K_1 + K_2 h)u/h$$

where $K_1, K_2$ depend on $f$ and $x$. You may use the bound from Question 3.

d. Hence estimate, in terms of $u$, the optimal choice for h.

e. Write a script file called `BetterDerivative.m` to test your estimates for $f(x) = \exp x$ and $x = 1$, $h = 0.5, 0.25, \ldots 2^{-52}$ and plot the errors, as in `ForwardDerivative.m`. Comment on your results.