Mittuniversitetet

MID SWEDEN UNIVERSITY

Institution of Information Systems and -Technology (IST)

Laboration 2
Monte Carlo Simulations

(MA069G, Mathematical Modelling 6hp)

Eric Johansson (erjo2002@student.miun.se)
Can Kupeli (caku2002@student.miun.se)
Samuel Greenberg (sagr1908@student.miun.se)

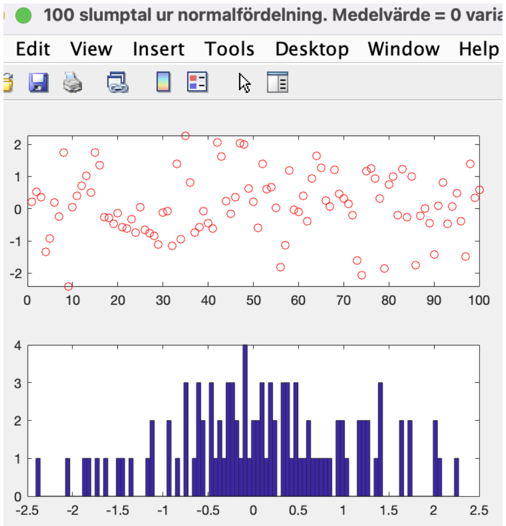
Wednesday 21st September, 2022

Supervisor:
Cornelia Schiebold
Magnus Eriksson
Suprokash Hazra

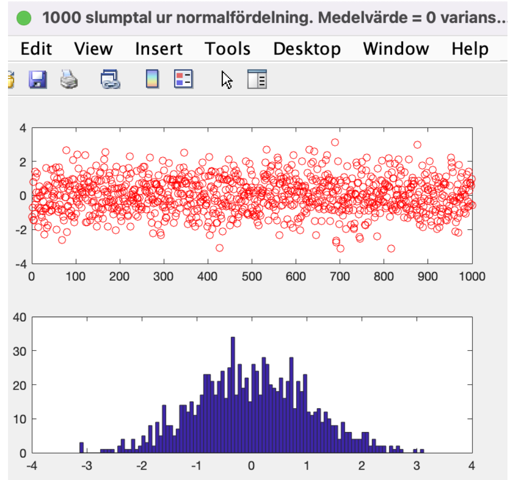
Contents

Question 1

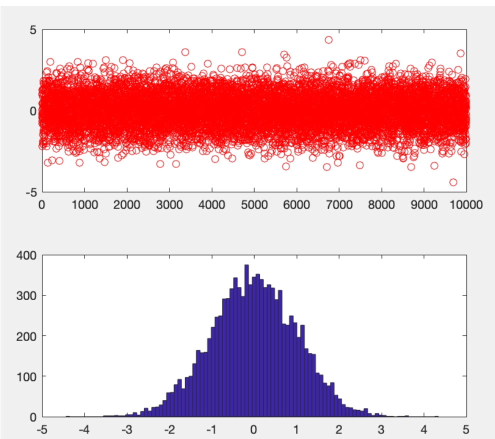
(a)



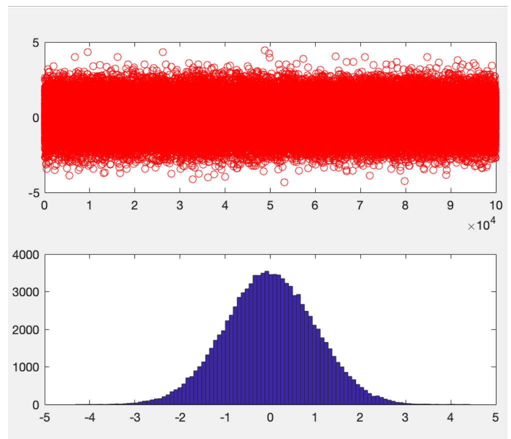
(a) 100 Randomly generated numbers



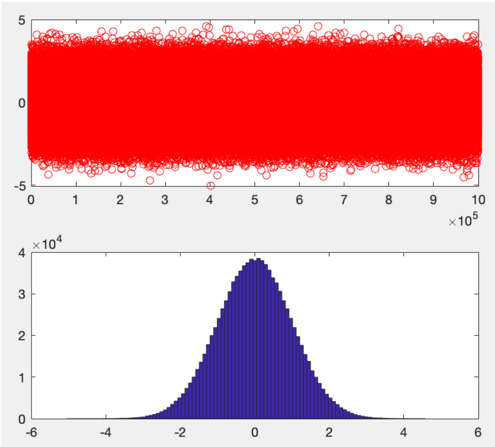
(b) 1000 Randomly generated numbers



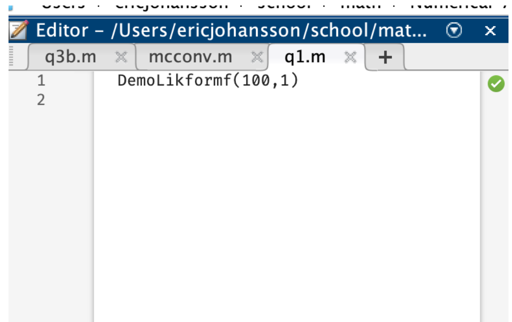
(a) 10000 Randomly generated numbers



(b) 100000 Randomly generated numbers



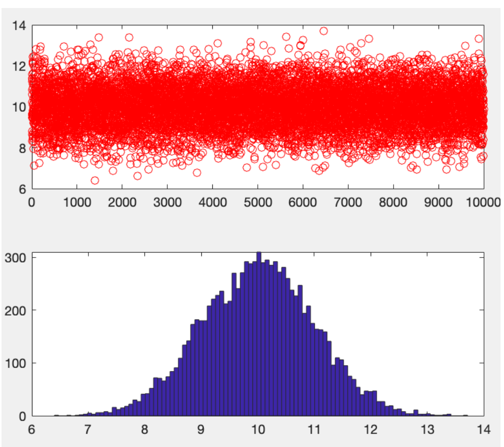
(a) 10000 Randomly generated numbers



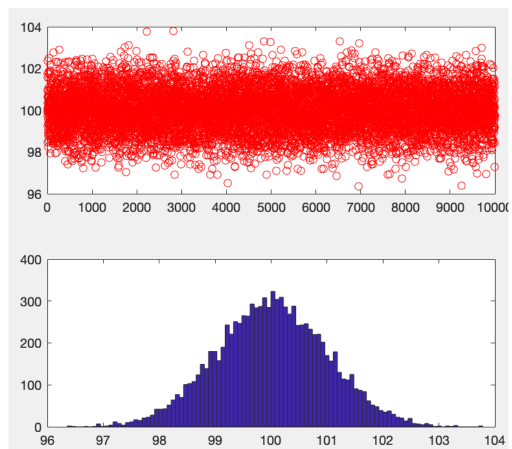
(b) Code for question 1

As N icreases the distubution becomes more alike a Normal distribution.

(b)



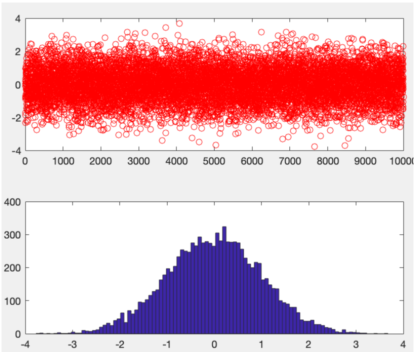
(a) $\mu = 10$



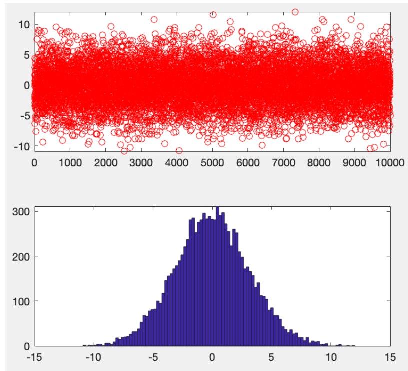
(b) $\mu = 100$

We can see that the distribution is about the same and the main difference is that they are centered around different values.

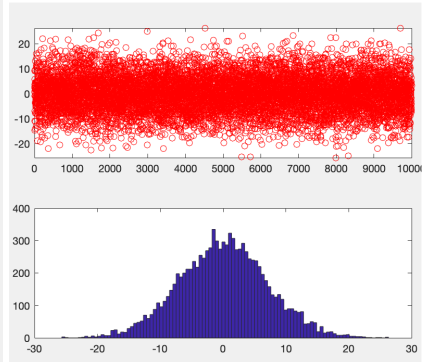
(c)



(a) $\sigma^2 = 1$



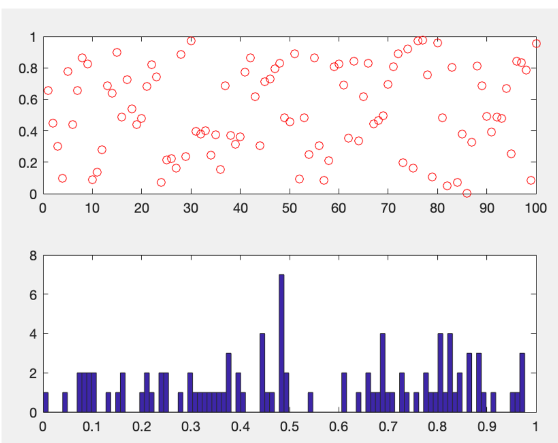
(b) $\sigma^2 = 10$



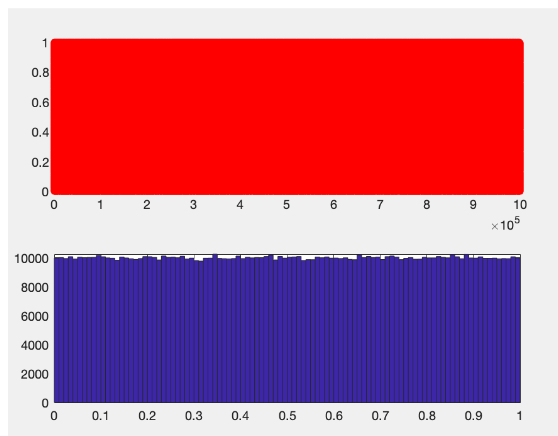
(c) $\sigma^2 = 50$

Once again the form of the distribution is mostly the same, however we can see now that the values extend more to the side as the variance increases. It doesn't increase by five times when comapring 10 to 50 because the standard deviation is $\sqrt{\text{Variance}}$. Since 50 is $5 \cdot 10$ it means that the standard deviation will incerase by $\sqrt{5}$ times which is pretty accurate with the results.

(d)



(a) $N = 10$



(b) $N = 1000000$

We can see that as $\lim_{N\to\infty}$ the distrubution becomes more like a uniform distobution.

Question 2

(a)

```
header

mean_y1 = mean(floor(1+6*rand(1,10.^3)));
mean_y2 = mean(floor(1+6*rand(1,10.^4)));
mean_y3 = mean(floor(1+6*rand(1,10.^5)));
```

(a) Code for 2a

Name	Value
mean_y1	3.5240
mean_y2	3.4854
mean_y3	3.5045

(b) Results for 2a

Yes as N increases the resulting average converges to 3.5

(b)

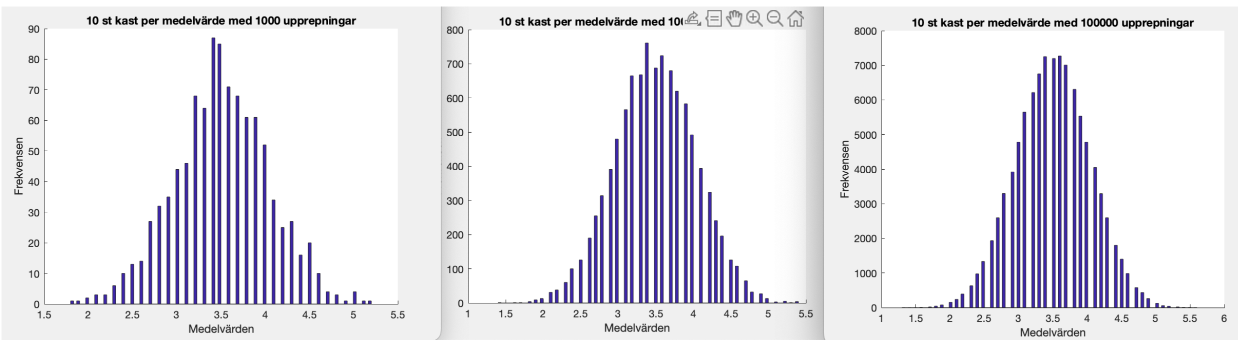


Figure 8: Results for 2b

We can see that it becomes more like a normal distribution as we repeat the test more times

(c)

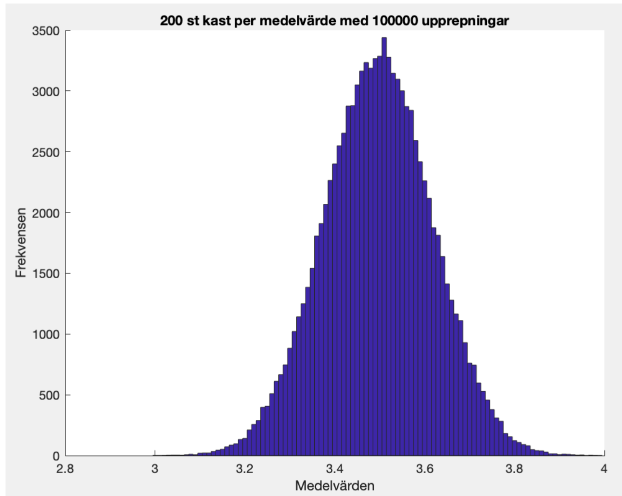


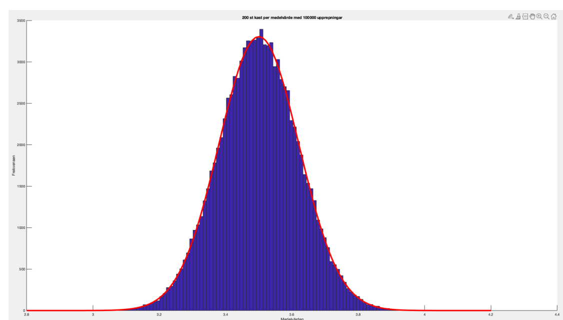
Figure 9: Results for 2c

It's possible to observe that the variance of this result is lower than that of question 2b. This is logical since we take the mean of more throws before adding them. This should create fewer outliers and thus lower the variance.

(d)

```
clear
clc
format long

n = 200; %antal kast per medelvärde
N = 10^5; %antal upprepningar
for i=1:length(N)
    figure;
    hold on
    r = zeros(1,N(i));
    for j=1:N(i)
        y = floor(1+6*rand(1,n));
        r(j) = mean(y);
    end
    hist(r,100); % Rita ett histogram med 100 intervall
    title([num2str(n),' st kast per medelvärde med ', num2str(N(i)), ' upprepningar'])
    xlabel('Medelvärden')
    ylabel('Frekvensen')
    shg

end
sigma = sqrt(35/(12*n));
mu = 3.5;
x =2.8:0.01:4.2;
f = 1/(sigma*sqrt(2*pi))*exp(-((x-mu).^2)/(2*sigma.^2));
plot(x,1000*f, 'Color','red','LineWidth',4);
```

(a) Code for 2d



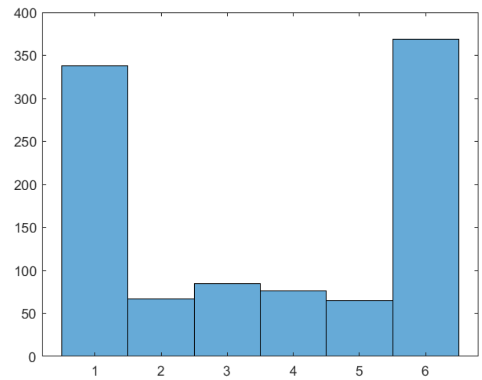
(b) Results for 2d

The magnitude of the function f is dependent on n so we need to multiply the function to make it fit our histogram.

(e)

```
header
loops = 1000;
p = [5, 1, 1, 1, 1, 5];

p = p / sum(p);
F = cumsum(p);

counter = 1;
inverse_sampler = zeros(1, loops);
while(counter <= loops)
    u = rand(1);
    inverse_sampler(counter) = sum(u > F) + 1;
    counter = counter + 1;
end

histogram(inverse_sampler)
p
```

(a) Code for 2e



(b) Histogram for 2e

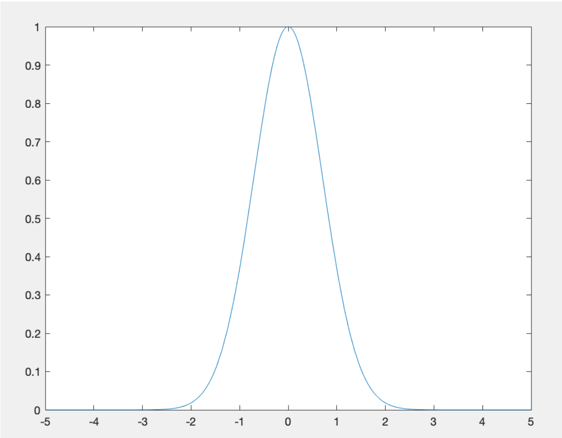
```
p =

   0.357142857142857   0.071428571428571   0.071428571428571   0.071428571428571   0.071428571428571   0.357142857142857
```

Figure 12: Probability of p

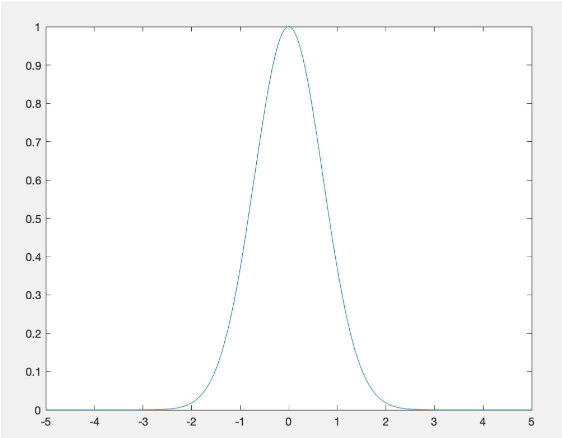
Comparing the probabilities for the different outcomes of p with the histogram we can see that they match up well. Where p(1) and p(6) is roughly 5 times greater than p(2), p(3), p(4), p(5). Looking at their y-values in the histogram they are about a fifth of the height at 1 and 6.

Question 3

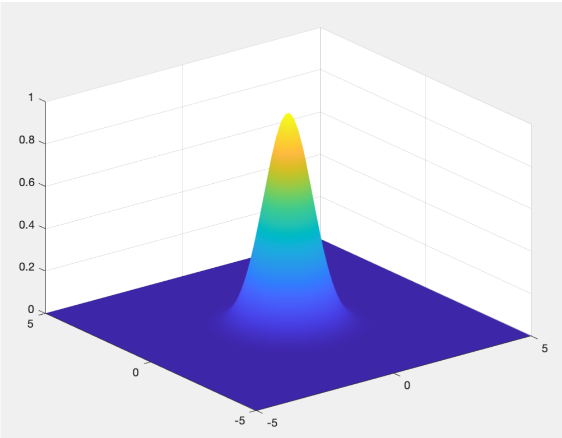
(a)



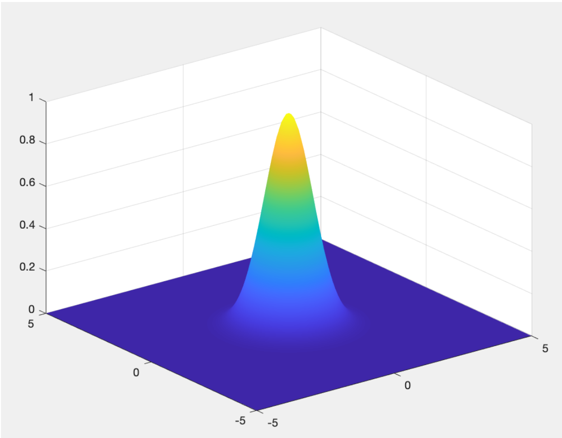
(a) 1D 100



(b) 1D 1000000



(a) 2D 100



(b) 2D 1000000

```
header

[N1d100,mu1d100,err1d100]=mcconv([-5 5],100);
waitforbuttonpress
[N1d1000000,mu1d1000000,err1d1000000]=mcconv([-5 5],1000000);
waitforbuttonpress
[N2d100,mu2d100,err2d100]=mcconv([-5 5; -5 5],100);
waitforbuttonpress
[N2d1000000,mu2d1000000,err2d1000000]=mcconv([-5 5; -5 5],1000000);
```

Figure 15: Code for question 3a

(b)

```
q3b.m  tarning_upprepa.m  q4.m  q3a.m  +
header

[N,mu, err] = mcconv([-5 5]);
a = polyfit(log(N),log(err),1);

waitforbuttonpress
fprintf("p = %.15f \t C = %.15f\n", a(1), exp(a(2)))

[N,mu, err] = mcconv([-5 5; -5 5]);
a = polyfit(log(N),log(err),1);
fprintf("p = %.15f \t C = %.15f\n", a(1), exp(a(2)))
```

```
ommand Window
A 1-dimensional integral is solved
for random numbers: 100     500    1000    5000    10000   100000
p = -0.492826899535472   C = 2.799991718641048
A 2-dimensional integral is solved
for random numbers: 100     500    1000    5000    10000   100000
p = -0.461378981347707   C = 8.546773223857077
>>
```

Figure 16: Question 3b

We can see that p does not change very much depending on the dimension. However C increases a lot when increasing dimention as you can see in Figure 16.

(c)

Monte Carlo is more efficient in higher dimensions than the Trapezoid method. This is because of the *Curse of Dimensionality* which states that when dimensionality.

Question 4

```
header
cumwinloss = 0;
day = 1;

while day < 366
    bs = blackjacksim(100);
    cumwinloss = cumwinloss + bs(end);
    day = day+1;
end
if cumwinloss > 0
    fprintf("Total yearly earnings is %d\n", cumwinloss)
else
    fprintf("Total yearly losses is %d\n", cumwinloss)
end
```

Figure 17: Code for question 4

Running this script usually yields a cumulative return of around -8000 to +2000. However it more often than not ends up being a net loss for the year.