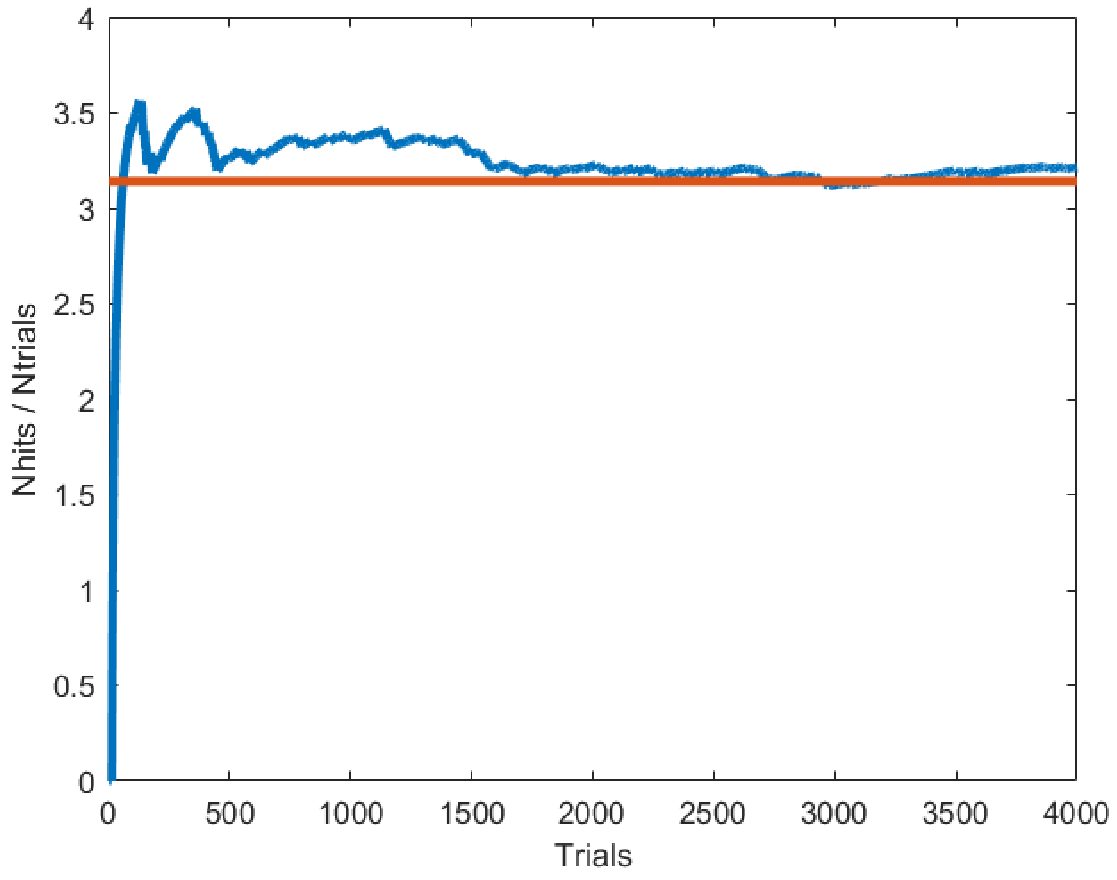
Homework-2
Umang Garg (Perm: 6787683)

Q1

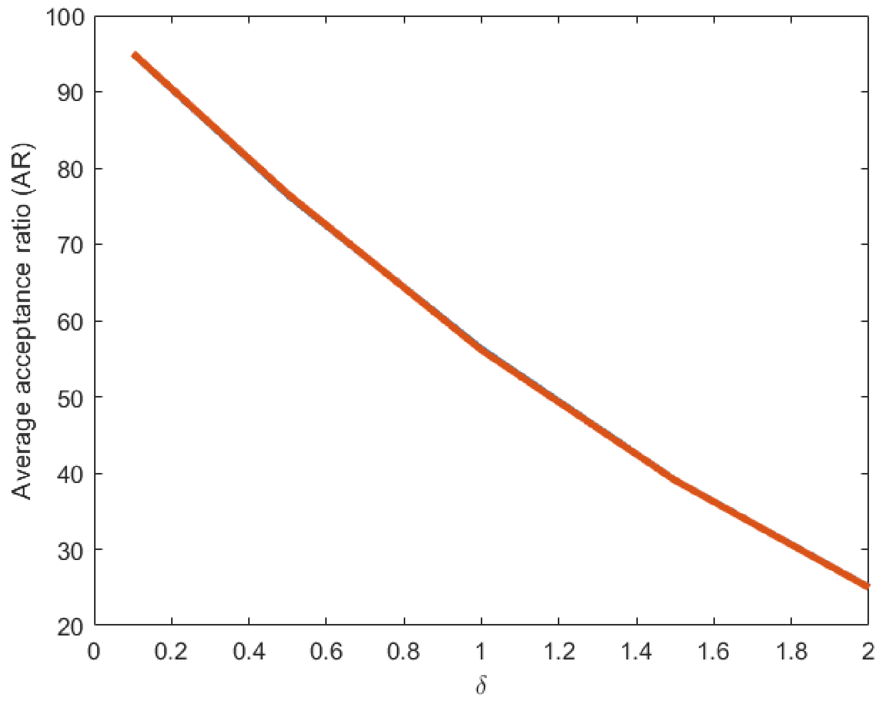
(a) Below is the required plot.



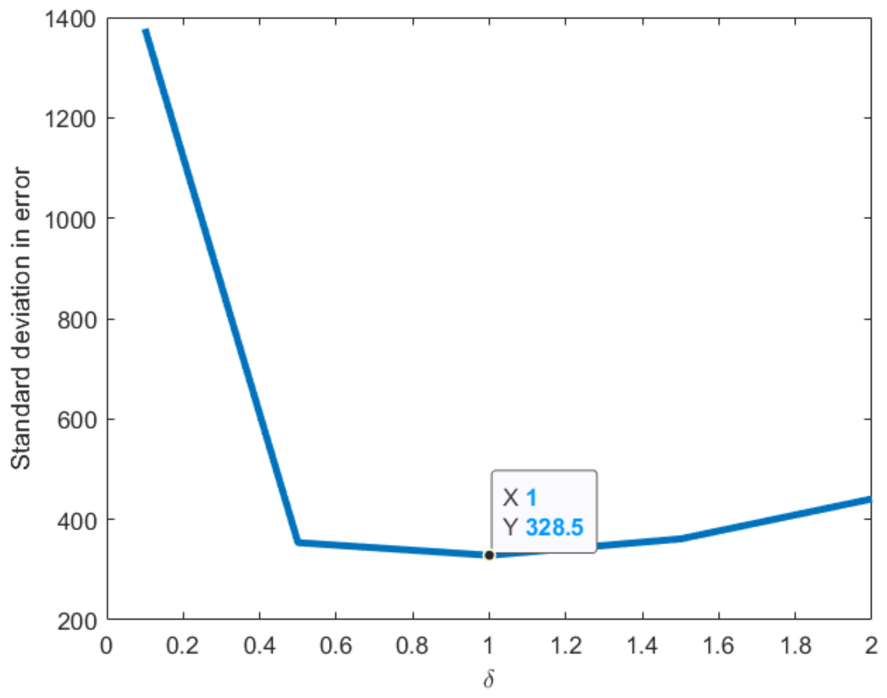
(b) The average acceptance rate is 0.8788 for N= 10000

(c) Average acceptance rate is 0.8785 for N =20000. No, we do not observe any significant change.

(d) Average acceptance ratio as percentage wrt step size is plotted below.



(e) Standard deviation of error with varying step sizes is plotted below.



We also note from 1(d) that the AR corresponding to least standard deviation is step size of 1. This rule hence is proven numerically, and hence agree to this rule of thumb, i.e. AR of 50 percent yields least error standard deviation. If we choose a small step size, then although most of our moves are highly probable to be accepted, it becomes imperative difficult to traverse the entire graph at end and the search might become highly local. On the other hand, if we have a large step size, we end up making large random walks in the search space, but most of the samples are rejected - not satisfying boundary conditions. Therefore, a trade-off is necessary between AR and search.

Q2 (a)

$$N(0,1) = \frac{1}{\sqrt{2\pi}} e^{-(x^2/2)}$$

$$\underline{\underline{\text{CDF}}} \quad y = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \, dx$$

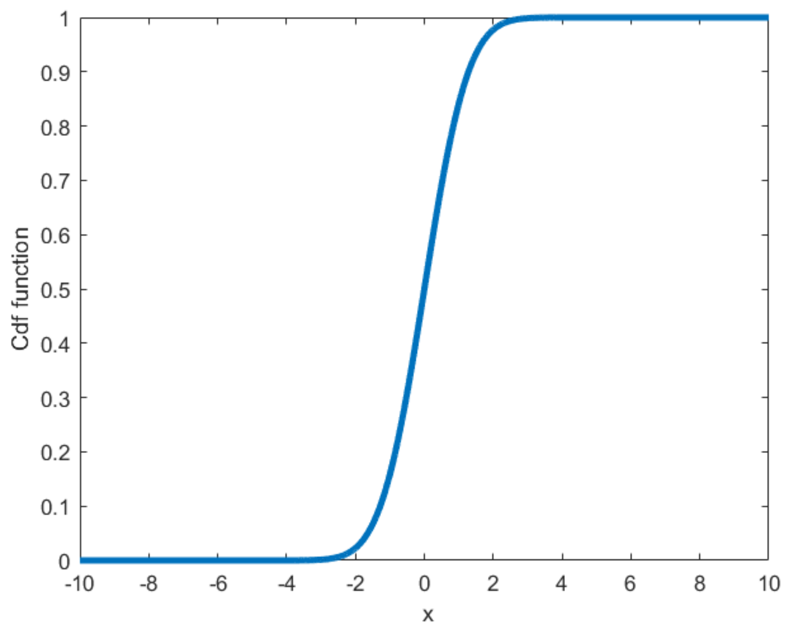
$$F(x) = y = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

$$\text{here } \mu = 0; \quad \sigma = 1$$

$$(a) \therefore \quad F(x) = y = \frac{1}{2}\left(1 + \operatorname{erf}(x/\sqrt{2})\right)$$

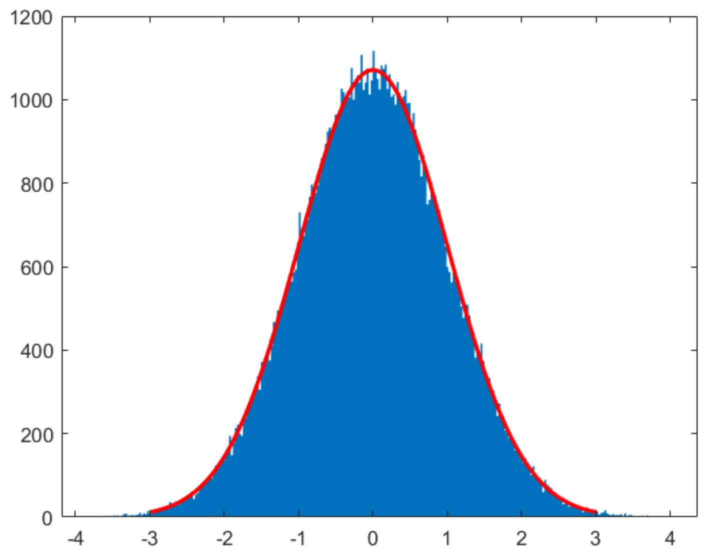
$$(b) \quad x = F^{-1}(y)$$

$$\operatorname{erf}^{-1}(2y-1) = x/\sqrt{2} \Rightarrow \boxed{x = \sqrt{2}\operatorname{erf}^{-1}(2y-1)}$$



Cdf is plotted above.

© The results taken from MATLAB code are: mean = 0.0024, Standard deviation = 0.9988

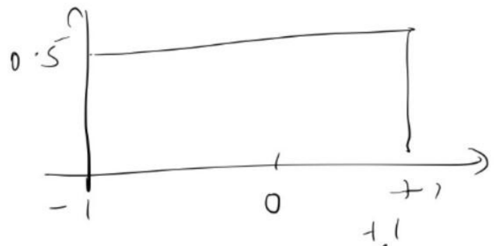


(d) The main idea of this approach is to assemble all the values of the probability distribution function together and cumulatively find its sum from one end to the other. This allows us to reduce the dimensionality of the problem, where instead of throwing samples in areas where the pdf is always zero, we end up using our samples efficiently and using them to find the regions of demarcation. Thus we strip the search space of no interest, making this sampling technique more efficient.

Q3

(a)

Variance, standard deviation of Uniform distribution



$$\text{Variance} = \int_{-1}^{+1} x^2 (0.5)\, dx - \mu^2 \hookrightarrow \boxed{0}$$

$$= \frac{x^3}{6}\Big]_{-1}^{+1}$$

$$= 2/6 = 1/3 \qquad S.D. = \underline{\underline{1/\sqrt{3}}}$$

(b)

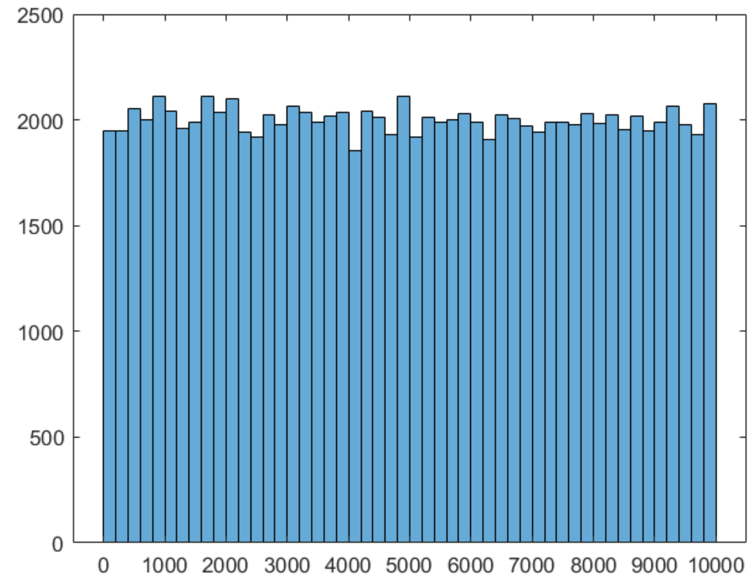


Figure: Sampling uniformly for 1 distribution

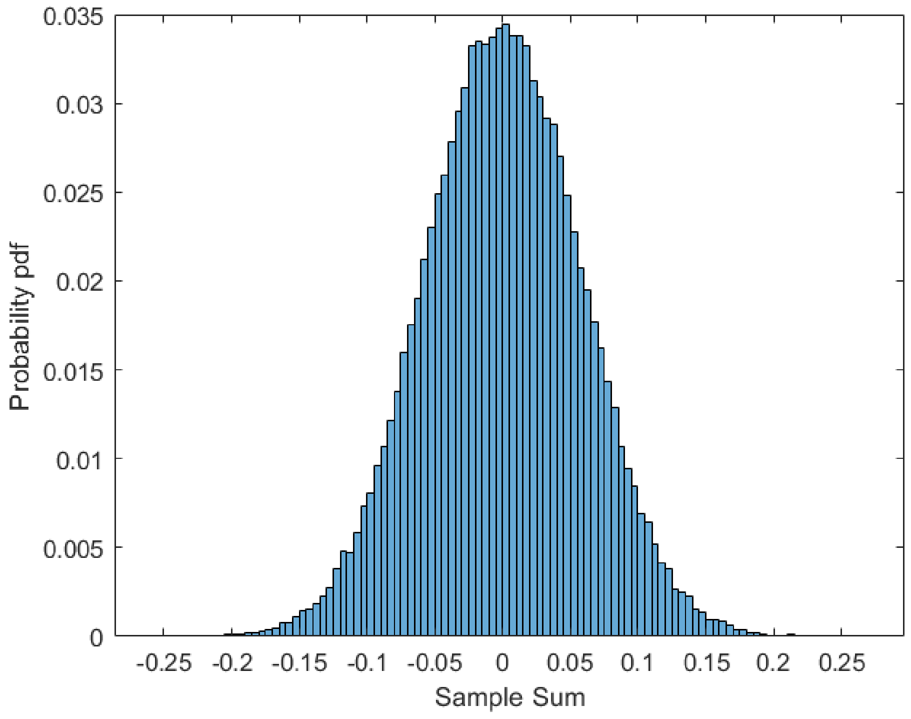


Figure: Probability distribution for 100 independent uniformly sampled distributions added

© and (d) Analytic variance calculation:

Standard deviation analytically = $\frac{\sigma}{\sqrt{N}}$ = 0.5770/10 = 0.057. This is proved numerically and analytically in the provided code.

Variance analytically = $\frac{Var}{N}$ = 0.0033; This is proven to match both numerically and analytically in the provided code.

The main idea of this approach is based on central limit theorem. Central limit theorem states that when N independently sampled distributions of any distribution are added together, the sum tends to be a gaussian as N tends to infinity. This tends to work because for each sample we take, the tendency to sample from edges of the distribution w.r.t to bulk is small, regardless of how the distribution looks. Hence, cumulatively if this effect is compounded over multiple samples, we understand that this in turn becomes the prime characteristic of the population of samples, i.e. the population of samples approaches normal distribution.

CODE APPENDED BELOW

```
clear all;
clc;


%% Problem 1

% Ntrials = 10000;
% Nruns = 1;
% delta = 0.25;
%
% for k =1:1:length(delta)
%
%     x = zeros(1, Ntrials);
%     y = zeros(1, Ntrials);
%
%     %   x(1) = -1 + 2*rand();
%     %   y(1) = -1 + 2*rand();
%
%     for j=1:1:Nruns
%
%         x(1) = 1;
%         y(1) = 1;
%         Nhit(1) = 0;
%         accept_moves(j) = 0;
%         reject_moves(j) = 0;
%
%       for i=1:1:Ntrials
%
%           delx = -delta(k) + 2*delta(k)*rand();
%           dely = -delta(k) + 2*delta(k)*rand();
%           N(i) = i;
%
%               x(i+1) = x(i);
%               y(i+1) = y(i);
%
%           if  (abs(x(i)+ delx) < 1) && (abs(y(i)+ dely) < 1)
%               x(i+1) = x(i)+ delx;
%               y(i+1) = y(i)+ dely;
%               accept_moves(j) = accept_moves(j) + 1;
%
%           else
%               reject_moves(j) = reject_moves(j) + 1;
%           end
%
%           if ((x(i+1))^2 + (y(i+1))^2 ) < 1
%              Nhit(i) = Nhit(i) + 1 ;
%           end
%              Nhit(i+1) = Nhit(i);
%
%              temp(i) = 4*Nhit(i)/N(i);
%
%        end
%
%    Run(j) = Nhit(Ntrials);
%    AR(j) = accept_moves(j)/ N ;
%
%    end
%
% avg_AR(k) = mean(AR)
% end
%
%
% M2= zeros(1,Ntrials);
% M2(:)= pi;
%
% Nhit = Nhit(1:Ntrials);
% figure(1);% Will give trials for only the last run
% p1= plot(N, temp);
% ylabel('Nhits / Ntrials');
% xlabel(' Trials ' );
% p1. LineWidth = 3;
% hold on;
% p2= plot(N, M2,'-');
% p2. LineWidth = 3;
%
% figure(2);
% p3 = plot(delta, avg_AR);
% ylabel('Average acceptance ratio (AR)');
% xlabel(' \delta ' );
% p3. LineWidth = 3;
%
% var(avg_AR);
% sigma = sqrt(var(avg_AR));
%
%
```

```
%% Problem 1 part e

% Ntrials = 10000;
% Nruns = 100;
% delta = [0.1, 0.5, 1, 1.5, 2]
%
% for k =1:1:length(delta)
%
%     x = zeros(1, Ntrials);
%     y = zeros(1, Ntrials);
%
%     for j=1:1:Nruns
%
%          x(1) = 1;
%          y(1) = 1;
%          Nhit(1) = 0;
%          accept_moves(j) = 0;
%          reject_moves(j) = 0;
%
%        for i=1:1:Ntrials
%
%            delx = -delta(k) + 2*delta(k)*rand();
%            dely = -delta(k) + 2*delta(k)*rand();
%            N(i) = i;
%
%                x(i+1) = x(i);
%                y(i+1) = y(i);
%
%            if  (abs(x(i)+ delx) < 1) && (abs(y(i)+ dely) < 1)
%                x(i+1) = x(i)+ delx;
%                y(i+1) = y(i)+ dely;
%                accept_moves(j) = accept_moves(j) + 1;
%
%            else
%                reject_moves(j) = reject_moves(j) + 1;
%            end
%
%            if ((x(i+1))^2 + (y(i+1))^2 ) < 1
%               Nhit(i) = Nhit(i) + 1 ;
%            end
%               Nhit(i+1) = Nhit(i);
%
%               temp(i) = 4*Nhit(i)/N(i);
```

```
%
%        end
%
%    AR(j) = accept_moves(j)/ Ntrials ;      % Acceptance ratio is accept/ Total
%    err_in_run(j) = (4*Nhit(Ntrials) - floor(Ntrials * pi));
%    Run(j) = Nhit(Ntrials);
%
%    end
%
% avg_AR(k) = mean(AR)
% sigma(k) = sqrt(var(err_in_run))
% end
%
%
% M2= zeros(1,Ntrials);
% M2(:)= pi;
%
% Nhit = Nhit(1:Ntrials);
% figure(1);% Will give trials for only the last run
% p1= plot(N, temp);
% ylabel('Nhits / Ntrials');
% xlabel(' Trials ' );
% p1. LineWidth = 3;
% hold on;
% p2= plot(N, M2,'-');
% p2. LineWidth = 3;
%
% figure(2);
% p3 = plot(delta, avg_AR*100);
% ylabel('Average acceptance ratio (AR)');
% xlabel(' \delta ' );
% hold on;
%
% p3. LineWidth = 3;
%
% figure(3);
% p4 = plot(delta, sigma);
% ylabel('Standard deviation in error');
% xlabel(' \delta ' );
% hold on;
%
% p4. LineWidth = 3;
```

```
%% Problem 2

% finding Cdf for normal distribution -- taken from Wikipedia

% x = -10:0.01:10 ;
% y = (1 + erf(x/sqrt(2)) )/2 ;
%
% figure(5);
% p5 = plot (x,y);
% ylabel('Cdf function');
% xlabel('x' );
% p5.LineWidth = 3;

% Nsamples = 100000;
%
% for i=1:1:Nsamples
%
%  y(i)= rand();
%  x(i) = (sqrt(2)) * erfinv(2*y(i) -1) ;
%
% end
%
% h = histfit(x);
% avg = mean(x)
% sigma = sqrt(var(x))

%% Problem 3

% N = 100000;
%
% x = -1:2/N:1;
% f = (1/(N+1))*ones(1,N+1);
% avg = 0;
% expectation_x_squared  =0;
% for i =1:1:N+1
%
%     avg = avg + (x(i)*f(i)) ;
%     expectation_x_squared = expectation_x_squared + f(i)*(x(i))^2 ;
% end
%
% % Using Var(X) = E(X2) – [E(X)]2
%
%
% variance = expectation_x_squared - avg^2 ;
% SD = sqrt(variance);


%% Problem 3 part b

clc;
clear all;

N = 100;
a=-1;b =1;

x = -1:2/N:1;
f = (1/(N+1))*ones(1,N+1);
avg = 0;
expectation_x_squared  = 0;
% for i =1:1:N+1
%
%     avg = avg + (x(i)*f(i)) ;
%     expectation_x_squared = expectation_x_squared + f(i)*(x(i))^2 ;
% end
%
% % Using Var(X) = E(X2) – [E(X)]2
%
%
% variance = expectation_x_squared - avg^2 ;
% SD = sqrt(variance);

% N_distributions = 100;
% N_samples = 1e5;
%
% for j= 1:1: N_distributions
%
%        spectrum(j,:) = a + (b-a)*rand(1,N_samples) ;
% end
%
% for i = 1:N_samples
%     spectrum_final(i) = sum(spectrum(:,i));
% end
%
% % Normalize the spectrum
%     spectrum_final = spectrum_final/N_distributions;
%
%     % Theoritical and Actual comparison
%
```

```
%     variance = var(spectrum(1,:));
%     sd = sqrt(var(spectrum(1,:)));
%     theory_sd = sd*(1/sqrt(N_distributions));
%     theory_var = (1/N_distributions)*variance;
%
%     sd_Actual = sqrt(var(spectrum_final));
%     var_Actual = var(spectrum_final);
%
%     figure();
%     histogram(spectrum_final,'Normalization','probability');
%     xlabel('Sample Sum');
%     ylabel('Probability pdf');
%
%
```