CHEMICAL AND ENERGY ENGINEERING

SIMULATIONS OF MECHANICAL PROCESSES

PROBLEM 4.5: SIEVING KINETICS

Submitted by:

1. Angshuman Buragohain
   Matriculation Nr. 221552
2. Devidas Khatri
   Matriculation Nr. 221549

Chemical and Energy Engineering
Otto-von-Guericke Universität, Magdeburg

Submitted to:

Prof. Berend van Wachem
Universitätsplatz, 39106,
Otto-von-Guericke Universität,
Magdeburg, G10-236

Authors: Angshuman Buragohain, Devidas Khatri

Date: 24th June 2019

PROBLEM 4.5: SIEVING KINETICS

A given sample of sand is sieved for 50 passes. The density of sand is 1450 kg/m3
The specification for the distribution of particle size is normal distribution
with a mean of d=100μm standard deviation of 0.2d and sieve pore size are given
as w = 92 μm and w = 108 μm. The objectives are:

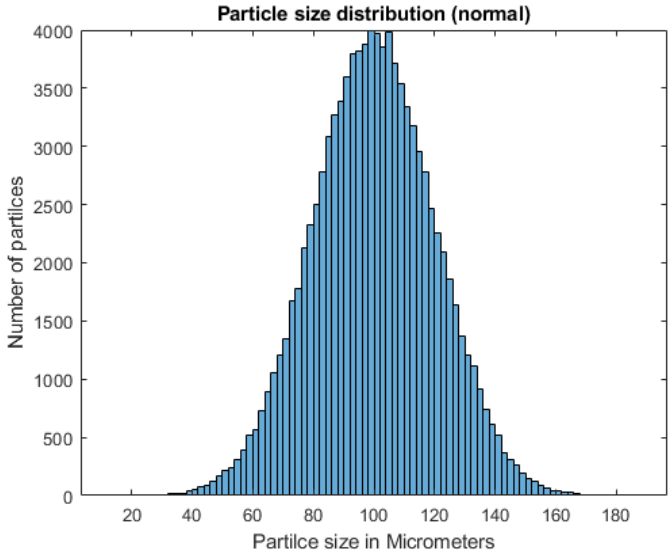
1. Create a particle size distribution of 10^5 particles and sieve size distribution of 10^4 sieve openings
2. Sieve the sample of sand into fine and coarse particles for 50 passes of sieves. Condition: One sieve opening encounters a particle atmost once. The coarse particles and remaining particles each pass go back to be feed for the next pass.
3. Plot the evolution of mass with each pass.
4. Plot the cumulative size distributions Q3(d) and the associated particle size distributions q3(d) for the feed, coarse and fine materials after P = 50 sieving passes.
5. Plot the separation function and determine efficiency of separation

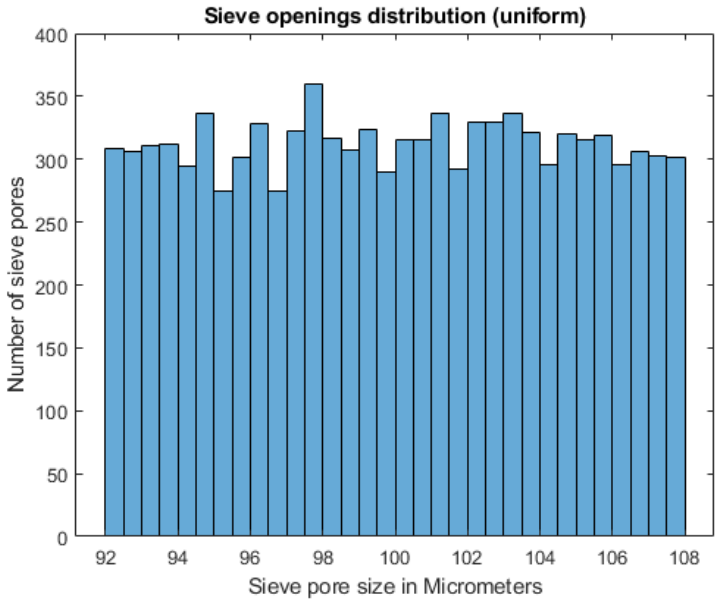
```
clear all;
close all;
```

Objective 1: Particle size and sieve opening distribution

```
    %Particle sample size
    Np=10^5;

    %Mean of particle size
    d=100;

    %Distribution parameters
    sigma = 0.2*d;
    mean = d;

    %Normal distribution of particle sizes with the given sigma and mean
    %value
    p_sz = sigma.*randn(Np,1) + mean;

    %Plotting the distribution
    figure(1);
    histogram(p_sz);
    title('Particle size distribution (normal)');
    xlabel('Partilce size in Micrometers');
    ylabel('Number of partilces')

%   Sieve openings distribution
```

```
Ns=10^4;
pmin=92;
pmax=108;
s_sz = pmin + (pmax-pmin).*rand(Ns,1);

figure(2);
histogram(s_sz,"BinWidth",0.5);
title('Sieve openings distribution (uniform)');
xlabel('Sieve pore size in Micrometers');
ylabel('Number of sieve pores')
```





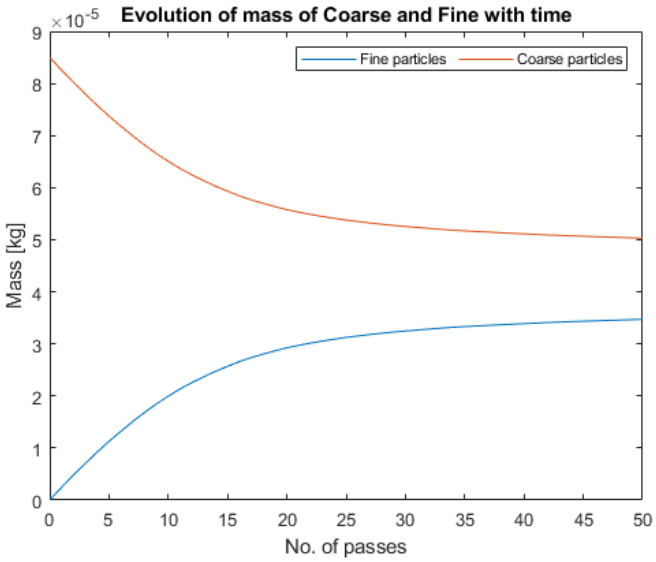
Objective 2: Separation of particles into fines and coarse

The particles are inspected in a for loop and with if conditions the number of particles encountering per sieve opening is restricted to 1. if a particle does not encounter a pore, it goes back into the feed for the next pass. After each pass the coarse particles and the remaining particles which do not encounter a sieve opening becomes the feed for the next pass. Thus, at the end of the for loop, the initial particles variable is equated with the coarse particles.

```
   for i=1:P
       b=1;
       p_coarse = [];

       %Generating 10^5 non-repeating random numbers to be used as random
       %indices for particle sizes. A different array of non-repeating
       %random indices for the feed particles is generated for each pass
       r=randperm(length(p_ini),length(p_ini));

% Running for loop to consider all the particles for sieving. The if loop randomly
% considers the particles randomly which passes through the sieve pores so that the else
% statement later considers the remaining particles which donot encounter a
% pore as a coarse particle which finally goes into the feed for the
% subsequent pass. The nested if loop separates the particles which
% encounter a pore into fine or coarse.
       for j=1:length(p_ini)
           if(j<=length(s_sz))
                if(s_sz(j)>p_ini(r(j)))
                    p_fine(a)=p_ini(r(j));
                    a=a+1;
                else
                    p_coarse(b)=p_ini(r(j));
                    b=b+1;
                end
           else
                p_coarse(b)=p_ini(r(j));
                b=b+1;
           end
       end

       % Creating cells for storing the fines and coarse particles after
       % each pass for observing later.
       f{i}=p_fine;
       c{i}=p_coarse;
       p_ini=p_coarse;
   end
```

Objective 3: Evolution of mass of coarse and fine particles with each pass

```
%RHS for calculating the mass of individual particles in kg
func_mass= @(y) (1/6)*rho*pi*(y*y*y)*10^(-18);

%Mass of fines before sieving is fm(1) which is zero
fm(1)=0;

%Mass of coarse before sieving cm(1) is the sum of masses of all the feed
%particles.
for j=1:length(p_sz)
    cm(1)=cm(1)+func_mass(p_sz(j));
end

% i=1 means after the first pass. since the mass for fm(1) is before the
% sieving started, the mass after sieving is starting to store from
% fm(2). The same goes for coarse particles at line 113.
for i=1:P
    for j=1:length(f{i})
        mass=func_mass(f{i}(j));
        fm(i+1)=fm(i+1)+mass;
    end
end

for i=1:P
    for j=1:length(c{i})
        mass=func_mass(c{i}(j));
        cm(i+1)=cm(i+1)+mass;
    end
end
```

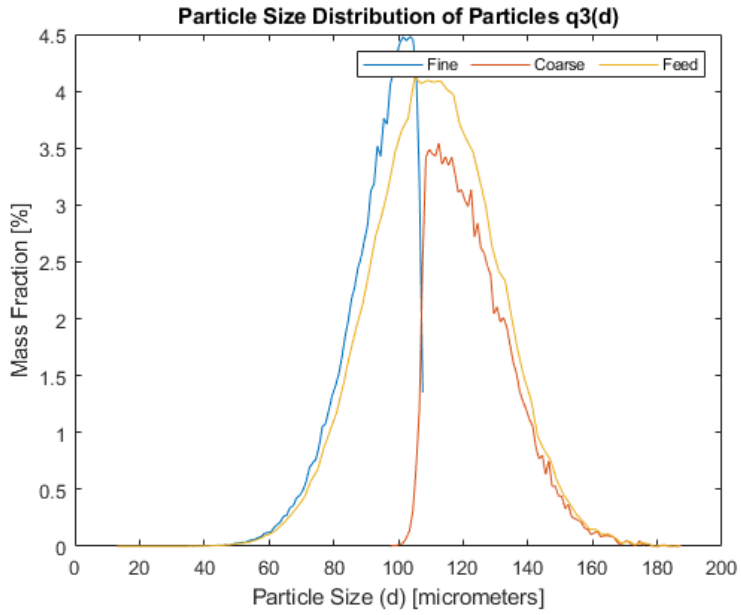


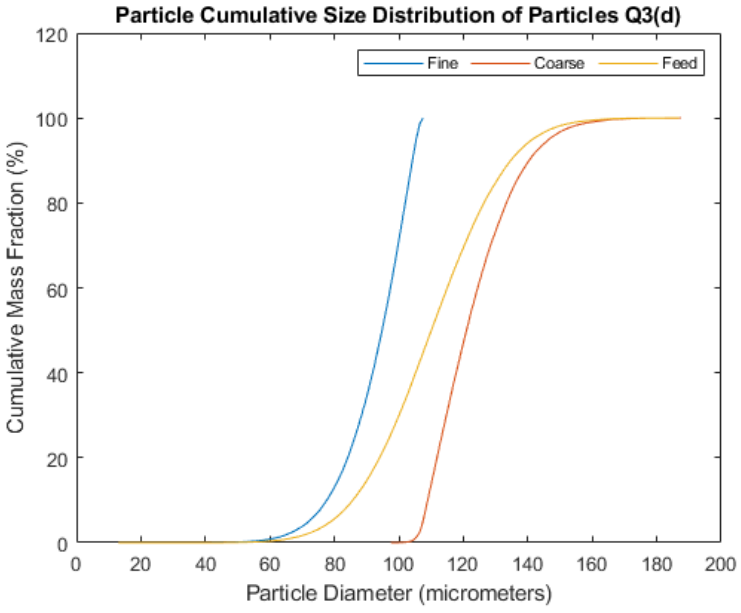
Objective 4: Cumulative size distribution Q3(d) and particle size distribution q3(d)

The for loop scans through each value of the fine particle sizes and sorts the sizes into the classes created while creating the histogram of p_fine and saving it to fine_distribution. The interval value is not always an integer due to which ceil function is used. As each particle size is sorted into intervals, the mass corresponding to that particle size is subsequently added to that particular interval. The cumulative distribution is later calculated by subsequently adding the mass fractions of each size interval. The same procedure is repeated for coarse and feed particles.

```
%********************** FINE PARTICLES ***************************************

    % The data for fine is sorted in ascending order for ease of
    % computation and the distribution is saved in a histogram form so that
    % the edges of each bar of the histogram can be used as intervals
    % to plot the size distribution and the cummulative distribution
    p_fine=sort(p_fine,"ascend");
    figure('Visible','off');
    fine_distribution=histogram(p_fine);

    % Assigning variables for the histogram bar properties for later use
    f_size_interval=fine_distribution.BinEdges;
    f_no_interval=fine_distribution.NumBins;
    f_width_interval=fine_distribution.BinWidth;

    f_mass=zeros(1,f_no_interval);
    finterval=zeros(1,f_no_interval);
    f_size_meaninterval=zeros(1,f_no_interval-1);

    % This for loop scans through each value of the fine particle sizes and
    % sorts the sizes into the classes created while creating the histogram
    % of p_fine and saving it to fine_distribution. The interval value is
    % not always an integer due to which ceil function is used. As each
    % particle size is sorted into intervals, the mass corresponding to that
    % particle size is subsequently added to that particular interval
    for i=1:length(p_fine)
        finterval(i)=(p_fine(i)-f_size_interval(1)+eps)/f_width_interval;
        f_mass(ceil(finterval(i)))=f_mass(ceil(finterval(i)))+func_mass(p_fine(i));
    end

    % The mass fraction and the cummulative mass fraction are calculated
    f_massfrac=f_mass.*100/sum(f_mass);
    f_cummulative_massfrac=cumsum(f_massfrac);

    % The mass fractions and the cummulative mass fractions are supposed to
    % be plotted against the mean value of each each interval. This for
    % loop calculates the mean values of each interval and saves it in an
    % array
```

```
    for i=1:length(f_size_interval)-1
        f_size_meaninterval(i)=(f_size_interval(i+1)+f_size_interval(i))/2;
    end
```



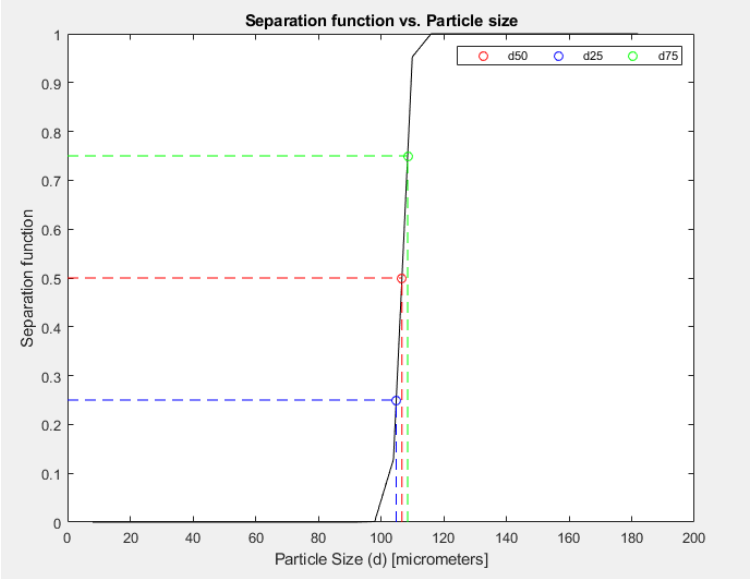


Objective 5: Separation function, cut diameter and sieving efficiency.

The same procedure for sorting the particle sizes into respective intervals and calculating the mass of each interval is followed as in Objective 4.

```
T=(Sep_coarse./(Sep_coarse+Sep_fine));
Sep_meaninterval=zeros(1,length(Sep_intervals)-1);
for i=1:length(Sep_intervals)-1
    Sep_meaninterval(i)=(Sep_intervals(i+1)+Sep_intervals(i))/2;
end
```

```
% Polyxpoly finds the point at which the poly lines intersect. The
% diameter of the particle corresponding to the value of T=0.5, 0.25
% and 0.75 i.e d50, d25 and d75 are extracted using polyxpoly
[d50,T50]=polyxpoly(Sep_meaninterval,T,[min(p_feed) max(p_feed)],[0.5 0.5],'unique');
[d25,T25]=polyxpoly(Sep_meaninterval,T,[min(p_feed) max(p_feed)],[0.25 0.25],'unique');
[d75,T75]=polyxpoly(Sep_meaninterval,T,[min(p_feed) max(p_feed)],[0.75 0.75],'unique');

% Separation efficiency
k=d25/d75;
```



The separation efficiency obtained for the process is 96.7% which is very good. The program gives a message at the end as shown below.