A statistical PM-calculation method for low-cost sensors avoiding costly histogram computations

Bernd Laquai, 07.10.17

In professional PM measurement instrumentation, the most frequently used approach to calculate PM-values is the use of a histogram on a sample set of particle size data obtained from measuring the scattered light approximating the probability density function for the particle sizes with a number of count bins. In the histogram, the differential bin quantities dN/N coarsely represent the probability of each size with a remaining quantization error that can be minimized with increasing the number of bins. The calculation of mass requires a subsequent weighting of the differential counts with the third power of the particle size and finally a summing up of the weighted counts per bin. In case the bin data are normalized to the total number of counts, the result represents the average mass per particle. With the ideal theory of spherical particles, the differential mass per particle in each bin can be expressed as:

$dM/N = dN/N * \pi/6 * d^3 * \rho$ (1)

and the total PM-value is calculated from summing up the individual bins with index i:

$PM_{tot} = sum (dM_i/N) = sum(dN_i/N* d_i^3) * \pi/6 * \rho$ (2)

For a PM-value limited to a certain size such as PM10 and neglecting the modelling of a specific separation characteristic, the summing would stop at a specific bin related to a respective maximum particle size, e.g. d=10µm.

Computationally however, the implementation of a so called multi-channel-analyzer (MCA) that sorts the particle size data into many bins requires a powerful processor. When PM measurement results have to be provided on a per second basis, and many particle events occur for very small particles, the sorting speed must be very high, minimizing any dead times. For low-cost sensors, the use of a high-performance microcontroller is often not possible due to cost reasons. Therefore, the calculation of PM-values without histogram data is very attractive.

In the following, an approach is described to calculate the total PM-mass from a simple average over particle sizes, provided the particle size distribution is given and does not change dramatically. If there are changes in the distribution while using this method, errors will occur. However, the resulting errors may still be acceptable with respect to the cost goal.

Without loss of generality, the method is described assuming a particle mass that is equally distributed across all sizes. In such a case, the particle size distribution approximated by dN/N needs to be proportional to the 3rd power of d:

$dN/N \sim d^{-3}$ (3)

For Monte-Carlo simulations, such a particle distribution can be generated with a mapping function of:

d=1/sqrt(2*rand(S)) (4)

with rand(S) generating a set of S random numbers equally distributed between 0 and 1. After this distribution mapping, d will theoretically be distributed proportionally to $\sim d^{-3}$. To avoid the problem if infinite counts for values of d close to 0 and zero counts for large d, the distribution can be limited to given bounds d_{min} and d_{max}. Providing d_{min} and d_{max} , an offset ofst and a gain g can be calculated for a modified random number generation:

$ofst= 1/(2*d_{max}^2)$ (5)

$g= 1/2*(1/d_{min}^2-1/d_{max}^2)$ (6)

rand2 = g*rand(S)+ofst (7)

This will create a bounded particle size distribution proportionally to ~ d^{-3} when using the mapping of equation (4) in an identical way for rand2 as for rand().

According to the conventional approach, the resulting distribution function dN/N calculated from a histogram is summed up for calculating total PM-mass. When we now pursue the goal to calculate PM-mass from an average, we must assume the particle size arrival process to be ergodic. This means that the probabilistic average is equal to the sequence (or time) average:

PM_{tot} = $sum(dN_i/N*d_i^3)$ = $average(y_i)=sum(y_i*dNy_i/N)$ (8)

with y(d) being a mapping function that transforms the individual particle sizes d_i to values y_i as the incoming particles are registered such that the resulting distribution dNy/y is shaped according to:

dNy/N ~ 1/y (9)

If we can define such a mapping function y(d), then the distribution obtained from:

$dM_i'/N = dy_i*dNy_i/N$ = constant for all i (10)

and thus also reflects the equally distributed mass distribution.

If the above criteria for y(d) are met, the expected value Ey of the distribution dNy/N will be:

$Ey = sum(y_i*dNy_i/N)$ (11)

As we can assume the particle arrival process to be ergodic, the sequence average calculated from the individual y_i is equal to the expected value from the probabilistic histogram calculation:

average (y_i) = Ey = PM_{tot} (12)

With methods described in /1/ the required mapping function can be constructed:

$y(d) = m*exp(1/d^2)$ (13)

and

$m=exp(-(1/d_{min}^2))$ (14)

where d_{min} is the minimum particle sizes assumed in the generation process.

The simulation results for a Monte-Carlo simulation are shown in fig. 1 and fig. 2 for a set of 1 million particles and a normalized, unit-less size d. In fig 1. the probability density is shown which is required to achieve a constant mass distribution, the simulation is shown in blue and the expected theoretical behavior in red. Since the particle size has to be limited to a reasonable lower bound to avoid number representation overflows, the mass distribution for the generated particle mass and the result of calculating the distribution from the determined ideal mapping function differ slightly in quantity but indeed show an equal distribution level for all sizes. By decreasing the minimum particle size d_{min} , it can be shown that the mass distribution obtained from using the mapping function y(d) really converges towards the mass distribution as determined traditionally. Calculating the average on all y_i

and comparing the results to the traditional histogram based PM_{tot} calculation finally proves that for the given reference particle distribution both methods are indeed equivalent.

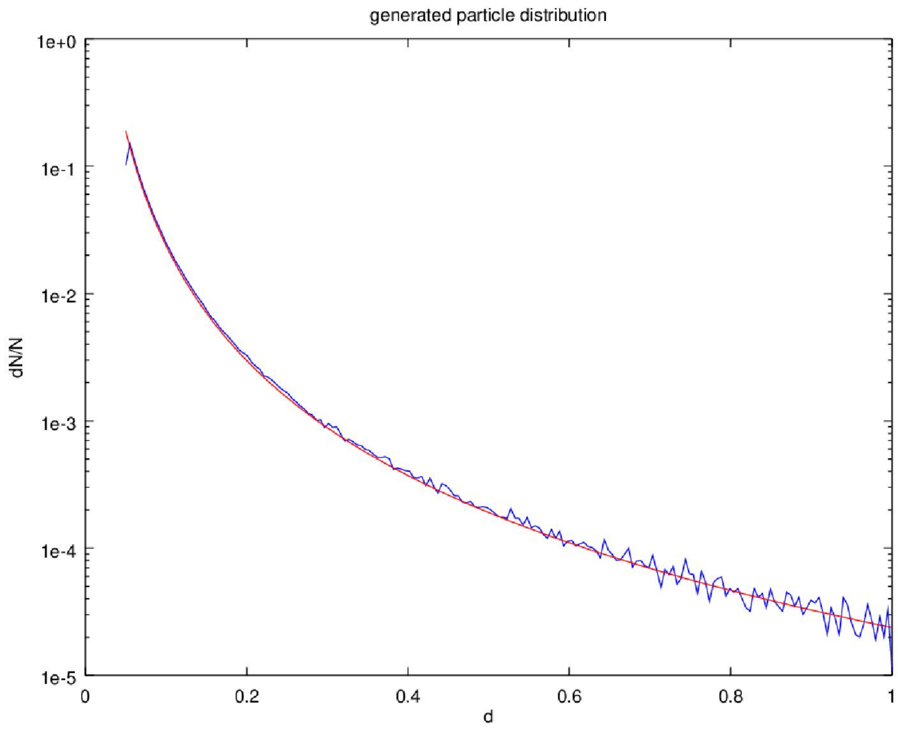

Fig. 1: Particle size distribution generated to obtain a constant mass distribution (10^6 particles), random simulation (blue), expected theoretical result (red)

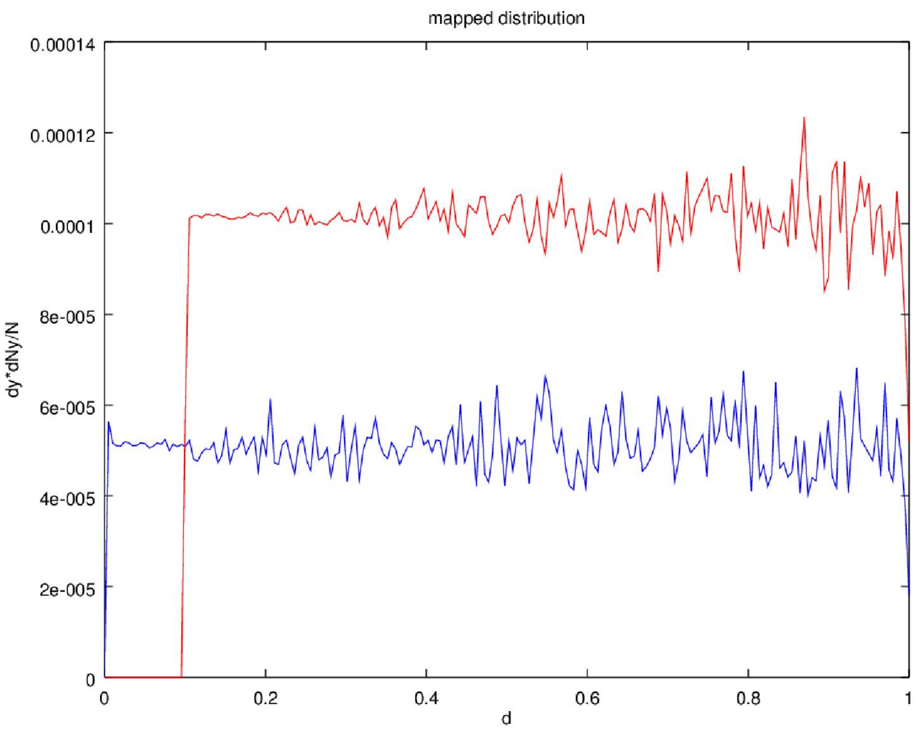

Fig 2a: Mass distribution from generation (red, with minimum size $d_{min} = 0.1$) and calculated with the mapping function (blue)

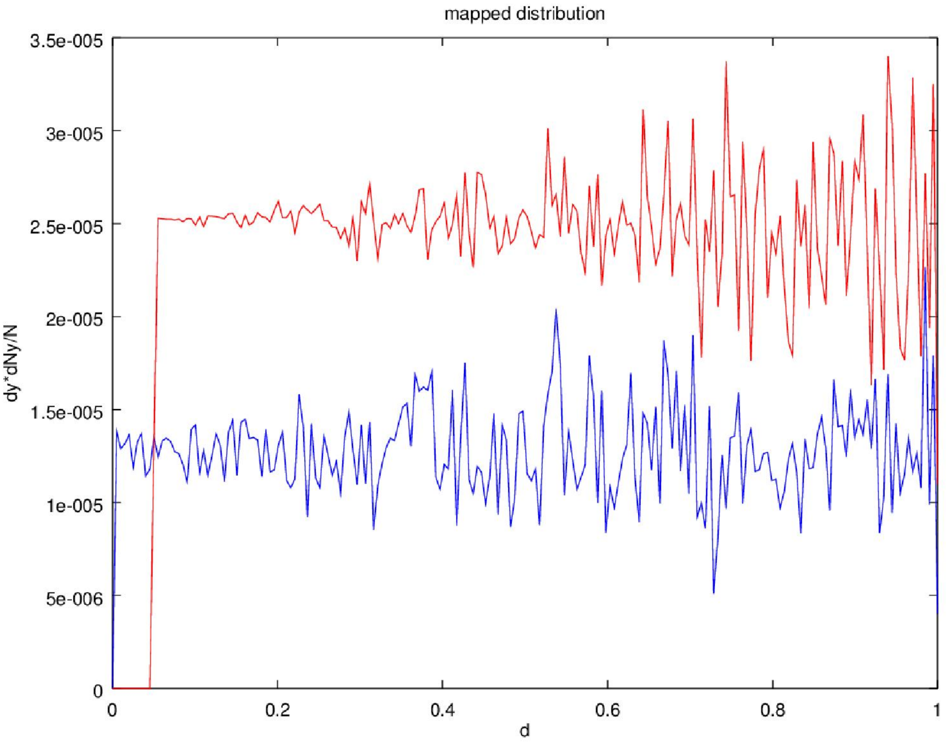

Fig. 2b: Mass distribution from generation (red, with minimum size d_{min} = 0.05) and calculated with the mapping function (blue)

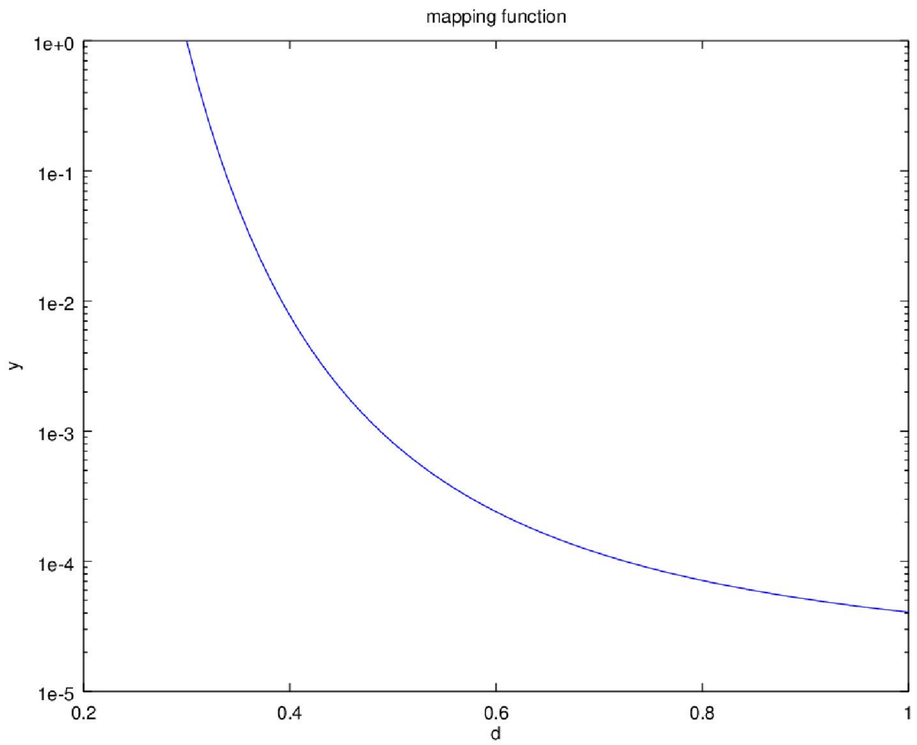

Fig. 3: Mapping function y(d) according to equation (13) and (14) for d_{min} = 0.1

For implementation in a low-cost sensor, the mapping function (13) with coefficient (14) can simply be stored as a lookup table in the microcontroller firmware. The mapping just requires comparison and fast index operations during program execution. Therefore, this approach can be implemented very easily in the firmware of a low-cost microcontroller avoiding any expensive histogram-based algorithm and thus requires much less computational power compared to the traditional approach. However, the disadvantage of this approach that has to be taken into account is the limited accuracy due to an error that will appear as soon as the sensor is exposed to a particle distribution that is not matching the reference spectrum that was assumed when generating the respective matching function.

According to the analysis of low-cost PM-sensors in laboratory, it is very likely that at least some of them (e.g. the SDS011 from Nova fitness) are using such type of PM-calculation method. For most of the low-cost PM-sensors the PM10-value is identical to PM_{tot} , since the maximum analog input of the A/D converter limits the scattering light signal automatically. The PM2.5-value seems to be a similar estimation while using a different mapping function.

Literature

/1/ Athanasios Papoulis: Probability, Random Variables, and Stochastic Processes; Mc Graw Hill 1965

/2/ Laser PM2.5Sensor specification Product model SDS011 V1.3; Nova Fitness Co., Ltd, China, 2015-10-9

Appendix

Script used for the Monte-Carlo simulation in GNU Octave (portable to Matlab)

```
clear;
nrSamp=1000000; %number of particles generated
dmin = 0.1; % min particle size [um] %needs to be > 0.05
dmax = 1; %max particle size
ofst= 1/(2*dmax^2);
g= 1/2*(1/dmin^2-1/dmax^2);
rnd2=(g*rand(1,nrSamp)+ofst);
d=1./sqrt(2*rnd2);

nrBins=200;
bins=linspace(0,1,nrBins);
[dN,di]=hist(d,bins);
dN_N = dN/nrSamp;
sum_dN_N=sum(dN_N);
di_t=(dmin:0.001:dmax);
dN_N_t=1./(g*di_t.^3)/nrBins*(dmax-dmin); %theoretical dN
if (1) %plot
  semilogy(di,dN_N,'-');hold on;
  semilogy(di_t,dN_N_t,'r-');
  title('generated particle distribution');
  xlabel ('d');
  ylabel ('dN/N');
  hold off;
end

m=exp(-(1/dmin^2));
y=m*exp(d.^(-2));
bins=linspace(min(y),max(y),nrBins);
[dNy,dy]=hist(y,bins);
dNy_N = dNy/nrSamp;
sum_dNy_N=sum(dNy_N);
figure;plot(dy,dy.*dNy_N);hold on;
plot(di,dN_N.*di.^3,'r'); hold off;
title('mapped distribution');
xlabel ('d');
ylabel ('dy*dNy/N');
sum1=sum(dy.*dNy_N);
sum2=sum(dN_N.*di.^3);
ave = mean(y);
fprintf(1,'sum1(mean): %.4f sum2(hist): %.4f ave(y): %.4f\n', sum1, sum2, ave);
```