Supporting information

Nanocluster aerosol emissions of a 3D printer

Mikko Poikkimäki,*,† Ville Koljonen,† Niko Leskinen,† Mikko Närhi,‡ Oskari Kangasniemi,† Oskari Kausiala,† and Miikka Dal Maso†

†*Aerosol Physics Laboratory, Physics Unit, Tampere University, FI-33720, Tampere, Finland*

‡*Photonics Laboratory, Physics Unit, Tampere University, FI-33720, Tampere, Finland*

E-mail: mikko.poikkimaki@tuni.fi

Number of Pages: 30

Number of Figures: 7

Number of Tables: 2

SI I - 3D printer properties

Table S1: The properties of the printer, the printing process and the printed frog object.

Property	**Value**
Model size (mm^3)	54.85 x 46.24 x 24.60
Total filament used (mm)	793.16
Total filament weight used (g)	4.2 to 5.3
Original filament diameter (mm)	2.85
Estimated print time (min)	36.05
Layer height (mm)	0.20
Layer count	124
Extrude speed (mm/s)	20.00
Nozzle move speeds (mm/s)	100.00; 20.00
Nozzle size (mm)	0.4

SI II - Material-temperature combinations printed

Table S2: All printing events including both successful and malfunction situations as well as total SMPS and PSM concentrations, N_{SMPS} and N_{tot} (cm^{-3}).

No.	Start time	Material	T_N/T_B*	T_N-setting*	Status	N_{SMPS}	N_{tot}
1	24-Mar 13:48	ABS	260/90	Recommended	Nozzle clog	$1.8{\cdot}10^4$	$4.0{\cdot}10^5$
2	24-Mar 14:50	ABS	260/90	Recommended	Nozzle clog	$5.5{\cdot}10^4$	$7.2{\cdot}10^4$
3	24-Mar 15:05	ABS	250/90	Recommended	Nozzle clog	$9.3{\cdot}10^3$	$2.1{\cdot}10^4$
4	24-Mar 15:21	ABS	250/90	Recommended	Success	$3.0{\cdot}10^3$	$1.0{\cdot}10^4$
5	27-Mar 10:51	ABS	250/90	Recommended	Aborted, wrong object	$8.0{\cdot}10^3$	$9.1{\cdot}10^5$
6	27-Mar 11:23	ABS	250/90	Recommended	Aborted, wrong object	$7.7{\cdot}10^3$	$3.5{\cdot}10^4$
7	27-Mar 12:31	ABS	250/90	Recommended	Success	$4.0{\cdot}10^4$	$3.7{\cdot}10^4$
8	27-Mar 13:13	ABS	230/90	Recommended	Nozzle clog	$9.4{\cdot}10^3$	$6.4{\cdot}10^3$
9	27-Mar 13:35	ABS	230/90	Recommended	Nozzle clog	$2.3{\cdot}10^3$	$9.9{\cdot}10^3$
10	27-Mar 13:42	ABS	230/90	Recommended	Nozzle clog	$2.3{\cdot}10^3$	$8.3{\cdot}10^4$
11	27-Mar 13:49	ABS	260/90	Recommended	Cleaning nozzle with high temp.	$5.2{\cdot}10^3$	$1.1{\cdot}10^5$
12	27-Mar 14:04	ABS	260/90	Recommended	Cleaning nozzle with high temp.	$2.7{\cdot}10^3$	$4.5{\cdot}10^4$
13	27-Mar 14:12	ABS	260/90	Recommended	Cleaning nozzle with high temp.	$2.9{\cdot}10^3$	$2.1{\cdot}10^4$
14	27-Mar 14:20	ABS	240/90	Recommended	Nozzle clog	$2.6{\cdot}10^3$	$2.1{\cdot}10^4$
15	27-Mar 14:48	ABS	240/90	Recommended	Success	$1.0{\cdot}10^3$	$7.1{\cdot}10^3$
16	27-Mar 16:27	PLA	210/60	Recommended	Nozzle clog, Data log fail	-	$4.0{\cdot}10^4$
17	27-Mar 16:39	PLA	210/60	Recommended	Nozzle clog, Data log fail	-	$4.3{\cdot}10^4$
18	27-Mar 16:51	PLA	210/60	Recommended	Data log fail	-	$1.1{\cdot}10^4$
19	28-Mar 09:51	PLA	210/60	Recommended	Aborted	$1.0{\cdot}10^3$	$1.2{\cdot}10^4$
20	28-Mar 09:57	PLA	210/60	Recommended	Success	$6.5{\cdot}10^2$	$1.9{\cdot}10^3$
21	28-Mar 10:40	PLA	210/60	Recommended	Success	$3.9{\cdot}10^2$	$1.6{\cdot}10^3$
22	28-Mar 11:17	PLA	220/60	Recommended	Data log fail	-	$3.8{\cdot}10^3$
23	28-Mar 11:37	PLA	220/60	Recommended	Success	$1.5{\cdot}10^3$	$5.7{\cdot}10^3$
24	28-Mar 12:19	PLA	220/60	Recommended	Success	$1.3{\cdot}10^3$	$4.4{\cdot}10^3$
25	28-Mar 12:58	PLA	190/60	Recommended	Nozzle clog	$4.8{\cdot}10^2$	$2.4{\cdot}10^3$
26	28-Mar 13:28	nGEN	240/80	Recommended	Data log fail	-	$5.3{\cdot}10^4$
27	28-Mar 13:38	nGEN	240/80	Recommended	Success	$2.2{\cdot}10^3$	$1.2{\cdot}10^4$
28	28-Mar 14:32	nGEN	240/80	Recommended	Success	$5.9{\cdot}10^3$	$2.4{\cdot}10^4$
29	28-Mar 15:18	nGEN	220/80	Recommended	Data log fail	-	$2.9{\cdot}10^3$
30	28-Mar 15:34	nGEN	220/80	Recommended	Success	$1.7{\cdot}10^3$	$3.0{\cdot}10^3$
31	28-Mar 16:23	nGEN	220/80	Recommended	Success	$1.6{\cdot}10^3$	$2.5{\cdot}10^3$
32	28-Mar 17:17	nGEN	250/80	Over recomm.	Success	$6.9{\cdot}10^5$	$2.0{\cdot}10^6$
33	28-Mar 18:01	nGEN	250/80	Over recomm.	Success	$7.9{\cdot}10^5$	$2.4{\cdot}10^6$
34	29-Mar 10:56	PLA+W	220/60	Over recomm.	Nozzle clog	$1.2{\cdot}10^3$	$1.0{\cdot}10^5$
35	29-Mar 11:04	PLA+W	220/60	Over recomm.	Nozzle clog	$5.1{\cdot}10^4$	$6.3{\cdot}10^4$
36	29-Mar 11:26	PLA+W	220/60	Over recomm.	Success	$9.4{\cdot}10^4$	$3.0{\cdot}10^5$
37	29-Mar 12:09	PLA+W	220/60	Over recomm.	Success	$1.0{\cdot}10^5$	$3.1{\cdot}10^5$
38	29-Mar 12:55	PLA+W	190/60	Recommended	Nozzle clog	$1.2{\cdot}10^3$	$5.5{\cdot}10^3$
39	29-Mar 13:00	PLA+W	190/60	Recommended	Nozzle clog	$8.3{\cdot}10^2$	$3.6{\cdot}10^3$
40	29-Mar 13:03	PLA+W	200/60	Recommended	Nozzle clog	$7.9{\cdot}10^2$	$7.5{\cdot}10^3$
41	29-Mar 13:07	PLA+W	200/60	Recommended	Nozzle clog	$2.5{\cdot}10^3$	$1.4{\cdot}10^4$
42	29-Mar 13:11	PLA+W	260/60	Over recomm.	Cleaning nozzle with high temp.	$1.1{\cdot}10^5$	$3.4{\cdot}10^5$
43	29-Mar 13:18	PLA+W	220/60	Over recomm.	Nozzle clog	$6.5{\cdot}10^4$	$7.4{\cdot}10^4$
44	29-Mar 13:23	PLA+W	210/60	Recommended	Nozzle clog removed	$1.8{\cdot}10^4$	$5.6{\cdot}10^4$
45	29-Mar 13:56	PLA+W	210/60	Recommended	Success	$8.7{\cdot}10^4$	$2.7{\cdot}10^5$
46	29-Mar 14:57	PLA+Cu	220/60	Over recomm.	Nozzle clog, chamber dilution off	$1.9{\cdot}10^5$	$4.0{\cdot}10^5$
47	29-Mar 15:30	PLA+Cu	210/60	Recommended	Nozzle clog	$2.7{\cdot}10^3$	$1.1{\cdot}10^4$
48	29-Mar 15:45	PLA+Cu	220/60	Over recomm.	Nozzle clog	$1.2{\cdot}10^5$	$2.8{\cdot}10^5$
49	29-Mar 16:27	PLA+Cu	230/60	Over recomm.	Nozzle clog	$2.4{\cdot}10^5$	$8.6{\cdot}10^5$
50	30-Mar 10:37	nGEN	250/80	Over recomm.	Success	$2.3{\cdot}10^4$	$1.1{\cdot}10^5$
51	30-Mar 12:59	ABS	250/90	Recommended	Success	$1.8{\cdot}10^4$	$6.7{\cdot}10^4$
52	30-Mar 15:39	ABS	250/90	Recommended	Success	$1.3{\cdot}10^4$	$5.5{\cdot}10^4$
53	31-Mar 08:54	ABS	250/90	Recommended	Success	$1.8{\cdot}10^4$	$5.7{\cdot}10^4$
54	31-Mar 09:55	ABS	250/60	Recommended	Aborted, chamber open	$5.2{\cdot}10^3$	$3.8{\cdot}10^4$
55	31-Mar 15:14	PLA+Cu	220/60	Over recomm.	Success	$2.0{\cdot}10^5$	$5.9{\cdot}10^5$
56	31-Mar 15:53	PLA+Cu	220/60	Over recomm.	Success	$1.4{\cdot}10^5$	$4.5{\cdot}10^5$

*The nozzle temperature T_N (°C) varied from manufacturer recommended to higher than recommended values, while the bed temperature T_B (°C) was kept at the recommended value.

SI III - CPC comparison

Two different CPCs were used in the SMPS system to test if the aerosol particles have different condensation growth inside the CPC using butanol compared to water. This comparison between the CPCs showed in average 5 % higher concentration for the butanol based CPC. In fact, the butanol-CPC in SMPS system allows better comparability to the butanol based A20 CPC employed in the PSM system. Thus, the concentration data from the butanol-CPC was used for the calculation of the SMPS distribution. However, in the cases, when the butanol-CPC was reporting faulty concentration values, the concentration was taken from the water-CPC.

SI IV - PSM calibration data

Figure 1 presents the PSM calibration data, measured by Airmodus Ltd, for four PSM saturator flows applied in this study. We can approximate the calibration data as step functions by setting a cutoff to 50 % of the maximum PSM efficiency.[1] In this way, we get a corresponding cutoff diameter for each saturator flow. Moreover, as PSM measures cumulative distribution, we can calculate a concentration between two cutoff diameters as a subtraction of two cumulative distributions. We assume that all cutoff curves reach the same maximum efficiency of 77.3 ± 1.5 percent. As an additional note, the maximal efficiency, in Fig. S2, seems to be lower for the two smaller saturator flows (larger cutoff sizes) due to the limited particle size range only up to 4.5 nm. However, in reality, the maximal detection efficiency is similar for each saturator flow since the Airmodus nCNC A11, used for counting the particles, has maximum efficiency of circa 80 % for larger particles up to 1 µm.

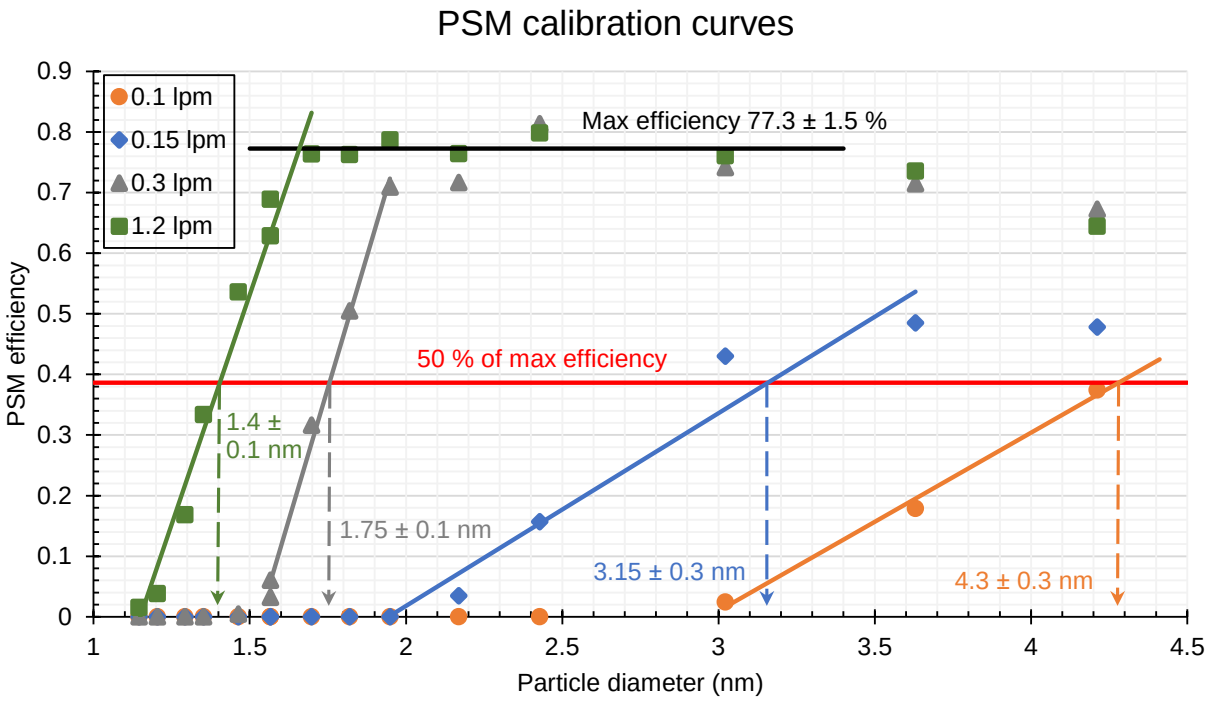

Figure S1: The PSM efficiency as a function of particle diameter for different saturator flows along with the cutoff diameters for each curve.

SI V -The exposure model code

We modeled the room concentrations by a indoor dispersion model presented in Drivas et al. [2] The model was implemented as MATLAB code and the correctness of the implementation verified by comparing to the results of the same paper.

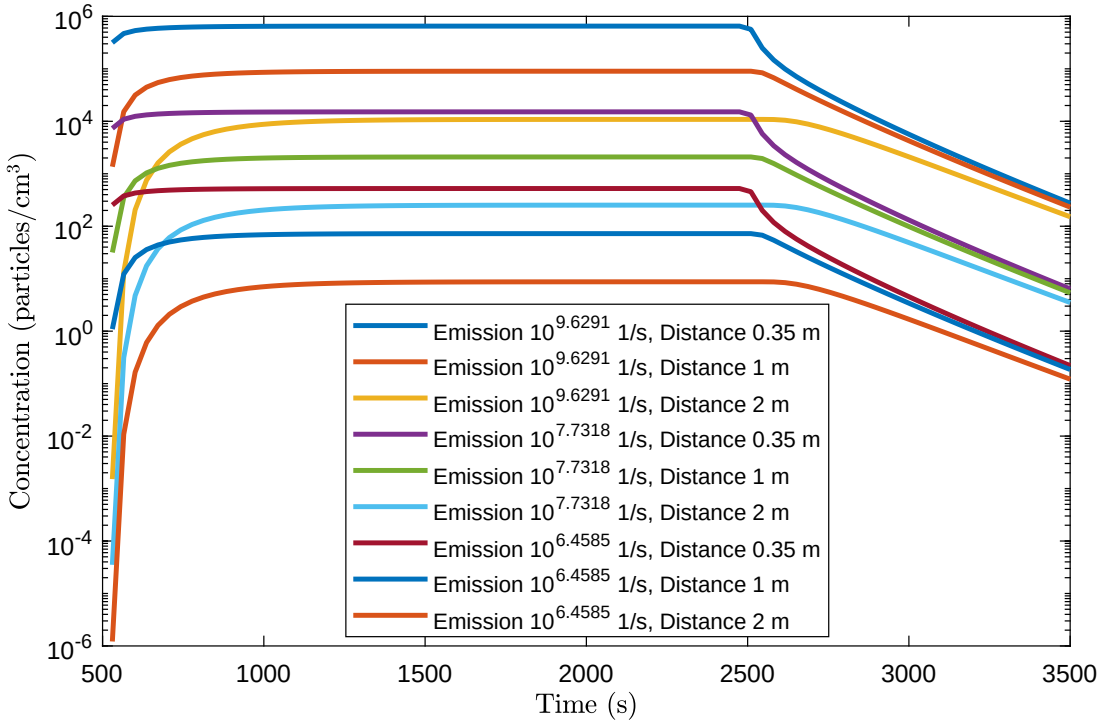

Figure S2: Modeled exposure concentrations over time at different distances in the laboratory room for the three emission scenarios.

The main program

A MATLAB function taking the input values, running the model and plotting the results.

```
function dispersionNCA3D()
%DISPERSIONNCA3D Aerosol concentration from 3D printing.
%   Dispersion of aerosol particles in a room
```

```
%   from a 3D printing source based on the measurements
%   by Poikkimaki et al. ES&T (2019).
%   System requirement: MATLAB R2017b or newer
%                       Functions cPuff.m and cPuffInstant.m
%   Licence:  Artworks Creative Commons CC BY-SA
%             Code     Boost Software License -
%                      Version 1.0 - August 17th, 2003
%   Authors:  Niko Leskinen and Mikko Poikkimaki

% input
p = struct;
% total and emission times
p.t = linspace(0,3500,100); % s
emissionStart = 500; % s
emissionEnd = 2500; % s
p.emissionPeriod = [emissionStart, emissionEnd];
% room dimensions - laboratory room
p.xRoom = 9; % m
p.yRoom = 6;
p.zRoom = 3;
% source position
p.xSource = 0.7; % m
p.ySource = 3.2;
p.zSource = 0.9;
% measurement/observer position
distances = [0.35, 1, 2]; % m
x = 0.7 + distances;
```

```
p.y = 3.2;
p.z = 0.9;
% turbulent diffusion coefficient
p.K = 0.002; % m^2/s
% ventilation rate - laboratory room
p.a = 18/3600; % 1/s
% deposition rate
p.wd = 0; % m/s
% emission rate NCA size 1-3 nm
Q = [4.25745e9, 5.39233e7, 2.87422e6]; % 1/s
% total emission rate
Qtot = Q./[.44605, .24331, .37524]; % 1/s
% emission rate larger particles than NCA
Qlarge = Qtot - Q;

% other environment (p2) representing a normal home room
p2 = p;
p2.xRoom = 4; % m
p2.yRoom = 4;
p2.zRoom = 2.5;
p2.a = 0.5/3600; % 1/s

% output
% model concentration for different emission rates and
% distances
[~, cNCAMax, ctot, ctotMax, ~, clargeMax] = ncaConc(p, x,
   distances, Q, Qtot, Qlarge);
```

```
[~, ~, ~, ctotMax2, ~, clargeMax2] = ncaConc(p2, x, distances,
   Q, Qtot, Qlarge);


% rough concentration calculation assuming uniform mixing
volume = p.xRoom*p.yRoom*p.zRoom;
cUni = Q*(emissionEnd-emissionStart)/volume*1e-6; % 1/cm3
disp(['Concentrations assuming uniform mixing are ', ...
    num2str(cUni),' 1/cm^3.'])

% plot concentration over time in the laboratory room
set(0,'defaulttextinterpreter','latex')
set(groot,{'DefaultAxesXColor','DefaultAxesYColor',...
    'DefaultAxesZColor'},{'k','k','k'})

figure
set(gcf,'Unit','inches','Position',[0 0 7 4.33])
set(gcf, 'PaperSize', [7 4.33]);
legendInfo = cell(1,length(Q)*length(distances)); ileg = 1;
for iq = 1:length(Q)
    for idist = 1:length(distances)
        semilogy(p.t,ctot{iq,idist}*1e-6,'-','Linewidth',2)
        hold on
        legendInfo{ileg} = ['Emission 10^{', ...
            num2str(log10(Q(iq))), ...
            '} 1/s, Distance ', ...
            num2str(distances(idist)),' m' ];
```

```
        ileg = ileg +1;
    end
end
xlabel('Time (s)')
ylabel('Concentration (particles/cm$^3$)')
legend(legendInfo, 'Location', 'Best')

% barplot of steady-state concentrations
figure;
set(gcf,'Units','Inch','Position',[0 0 6.8 3.25])
set(gcf,'Units','Inch','PaperSize',[6.8 3.25]);

% Laboratory room
setappdata(gcf, 'SubplotDefaultAxesLocation', [0.1, 0.14, 0.95,
    0.8]);
sp1 = subplot(121);
h1 = bar(ctotMax'*1e-6, 'FaceAlpha',0.4, 'Linewidth', 0.5);
h1(1).FaceColor = 'k';
h1(2).FaceColor = 'r';
h1(3).FaceColor = 'b';
set(gca,'nextplot','add')
h2 = bar(clargeMax'*1e-6, 'FaceAlpha',0.6, 'Linewidth', 0.5);
h2(1).FaceColor = 'k';
h2(2).FaceColor = 'r';
h2(3).FaceColor = 'b';
% set ticks and labels
set(gca,'YScale','log')
```

```
axis([0.5 3.5 1e2 1e7])
set(gca,'xticklabel',{'0.35 m','1 m','2 m'})
ax = gca; ax.XAxis.TickLength = [0,0];
% OEL horizontal lines
hlower = refline([0,20000]);% OEL lower
hupper = refline([0,40000]);% OEL upper
highway = refline([0,100000]);% highway
urbanBg = refline([0,5000]);% bg
hlower.Color = 'k';
hupper.Color = 'k';
highway.Color = 'k';
urbanBg.Color = 'k';
hlower.LineStyle = '--';
hupper.LineStyle = '--';
highway.LineStyle = ':';
urbanBg.LineStyle = '-.';
% text boxes
text(3.3, 5.5e6,'a)');
text(2.9, 150000,'Highway');
text(2.9, 27000,'OELs');
text(2.9, 7000,'Urb. BG');
annotation('textarrow',[0.18 0.14],[0.85 0.74],...
    'String','NCA ','Interpreter','latex')
text(0.7, 2.5e6, [num2str(round(cNCAMax(1,1)/ctotMax(1,1)*100)
   ,2), ' \%'])
annotation('textarrow',[0.20 0.17],[0.84 0.48],...
    'String','','Interpreter','latex')
```

```
text(0.9, 6e4, [num2str(round(cNCAMax(2,1)/ctotMax(2,1)*100),2)
   , ' \%'])
annotation('textarrow',[0.22 0.20],[0.84 0.24],...
    'String','','Interpreter','latex')
text(1.27, 1e3, [num2str(round(cNCAMax(3,1)/ctotMax(3,1)*100)
   ,2), ' \%'])
set(gca,'TickLabelInterpreter','latex')
set(gca,'fontsize', 10)

% Home room
setappdata(gcf, 'SubplotDefaultAxesLocation', [0.00, 0.14,
   0.95, 0.8]);
sp2 = subplot(122);
h1 = bar(ctotMax2'*1e-6, 'FaceAlpha',0.4, 'Linewidth', 0.5);
h1(1).FaceColor = 'k';
h1(2).FaceColor = 'r';
h1(3).FaceColor = 'b';
set(gca,'nextplot','add')
h2 = bar(clargeMax2'*1e-6, 'FaceAlpha',0.6, 'Linewidth', 0.5);
h2(1).FaceColor = 'k';
h2(2).FaceColor = 'r';
h2(3).FaceColor = 'b';
% set ticks and labels
set(gca,'YScale','log')
axis([0.5 3.5 1e2 1e7])
set(gca,'xticklabel',{'0.35 m','1 m','2 m'})
set(gca,'yTickLabel',[]);
```

```
ax = gca; ax.XAxis.TickLength = [0,0];
% OEL horizontal lines
hlower = refline([0,20000]);% OEL lower
hupper = refline([0,40000]);% OEL upper
highway = refline([0,100000]);% highway
urbanBg = refline([0,5000]);% bg
hlower.Color = 'k';
hupper.Color = 'k';
highway.Color = 'k';
urbanBg.Color = 'k';
hlower.LineStyle = '--';
hupper.LineStyle = '--';
highway.LineStyle = ':';
urbanBg.LineStyle = '-.';
% text boxes
text(3.3, 5.5e6,'b)');
text(1.69, 1.8e6,'nGEN');
text(1.69, 1.1e6,'250$^\circ$C');
text(1.91, 5e4,'ABS');
text(1.91, 2.8e4,'250$^\circ$C');
text(2.14, 1.2e3,'PLA');
text(2.14, 7.5e2,'210$^\circ$C');
set(gca,'TickLabelInterpreter','latex')
set(gca,'fontsize', 10)

% Axis labels
pos1=get(sp1,'position');
```

```
pos2=get(sp2,'position');
height=pos1(2)+pos1(4)-pos2(2);
width=pos2(1)+pos2(3)-pos1(1);
h5=axes('position',[pos1(1) pos2(2) width height],'visible','
   off');
h5.XLabel.Visible='on';
h5.YLabel.Visible='on';
axes(h5)
ylabel('test')
xlabel('Distance from the printer')
ylabel('Concentration (cm$^{-3}$)')

end

function [cNCA, cNCAMax, ctot, ctotMax, clarge, clargeMax] =
   ncaConc(p, x, distances, Q, Qtot, Qlarge)
cNCA = cell(length(Q),length(distances));
ctot = cNCA; clarge = cNCA;
cNCAMax = zeros(length(Q),length(distances));
ctotMax = cNCAMax; clargeMax = cNCAMax;
for iq = 1:length(Q)
    for idist = 1:length(distances)
        p.x = x(idist);
        % NCA
        p.Q = Q(iq);
        conc = cPuff(p);
        cNCA{iq, idist} = conc;
```

```
            cNCAMax(iq, idist) = max(conc);
            % total
            p.Q = Qtot(iq);
            conc = cPuff(p);
            ctot{iq, idist} = conc;
            ctotMax(iq, idist) = max(conc);
            % large
            p.Q = Qlarge(iq);
            conc = cPuff(p);
            clarge{iq, idist} = conc;
            clargeMax(iq, idist) = max(conc);
        end
    end

end
```

The cPuff.m function

A sub-function needed in the main program to calculate the concentration at certain distance from continuous source.

```
function c = cPuff(p)
%CPUFF Concentration from puff source.
%   p is struct with input parameters. If emissionPeriod input
%   parameter is scalar, then concentration is calculated
%   assuming instant puff source. If emissionPeriod is array
%   with range, e.g., [2,5] or [2,inf], then concentration
```

```
%   is calculated assuming continuous source emitting during
%   given range. If source is instantanious, Q parameter is
%   interpreted as amount, and if source is continuous,
%   Q is interpreted as emisssion flux (amount/time).
%   Coordinates:
%       p.x (m)
%       p.y (m)
%       p.z (m)
%       p.t (s)
%   Source:
%       p.xSource, location of source in x dimension (m)
%       p.ySource, location of source in y dimension (m)
%       p.zSource, location of source in z dimension (m)
%       p.emissionPeriod, time(range) when puff occurs (s)
%       p.Q, amount of emission (particles or mass, arbitrary
%            unit) or emission flux (particle or
%            mass per time, a.u./s)
%       p.K, turbulent diffusion coefficient (m^2/s)
%   Room:
%       p.xRoom, length of the room in x dimension (m)
%       p.yRoom, width  of the room in y dimension (m)
%       p.zRoom, height of the room in z dimension (m)
%       p.a, ventilation rate (1/s)
%       p.wd, deposition rate (m/s)
%   About input dimensions:
%       Input dimensions for for x, y, z and t should be:
%       * They all are equal and output is elementwise
```

```
%         calculation
%       OR
%       * x, y and z are equal, but t is scalar
%       OR
%       * x, y and z are scalar and t is arbitrary.
%         Output will be same dimensions as t
%   System requirement: MATLAB R2017b or newer
%                       Function cPuffInstant.m
%   Licence:  Artworks CC BY-SA
%             Code     Boost Software License -
%                      Version 1.0 - August 17th, 2003
%   Author:   Niko Leskinen

%checkInput(p)
if isscalar(p.emissionPeriod)
    % Instant source
    p.tEmission = p.emissionPeriod;
    c = cPuffInstant(p);
else
    % Continuous source
    if isscalar(p.t)
        c = cPuffContinuous(p);
    elseif isscalar(p.x) && isscalar(p.y) && isscalar(p.z)
        c = nan(size(p.t));
        t = p.t;
        for i = 1:numel(t)
            p.t = t(i);
```

```
            c(i) = cPuffContinuous(p);
        end
    else
        error('Error: Reduce dimensions')
    end
end
end


function checkInput(p)
assert(isfield(p,'emissionPeriod'), ...
    'Input p.emissionPeriod missing.')
assert(0 < numel(p.emissionPeriod) && ...
    numel(p.emissionPeriod) < 3,...
    'emissionPeriod should be scalar', ...
    'or array with two elements')
end

function c = cPuffInstantWrapper(p,t)
p.t = t;
c = cPuffInstant(p);
end

function c = cPuffContinuous(p)
assert(isscalar(p.t),...
    'p.t must be scalar to avoid ambiguousness')
t = p.t;
```

```
p.tEmission = 0;
emissionStarted = p.emissionPeriod(1);
emissionEnded   = p.emissionPeriod(2);
integrand = @(t) cPuffInstantWrapper(p,t);
if t < emissionStarted+eps
    % emission has not started yet
    c = zeros(size(p.x));
elseif emissionStarted < t && t < emissionEnded
    % during emission
    timeFromLastPuff = 0;
    timeFromEarliestPuff = t - emissionStarted;
    c = integral(integrand,...
        timeFromLastPuff,timeFromEarliestPuff,...
        'ArrayValued',true,'RelTol',1e-1);
else
    % after emission period
    timeFromLastPuff = t - emissionEnded;
    timeFromEarliestPuff = t - emissionStarted;
    c = integral(integrand,...
        timeFromLastPuff,timeFromEarliestPuff,...
        'ArrayValued',true,'RelTol',1e-1);
end

end
```

The cPuffInstant.m function

A sub-function calculating the concentration at certain distance resulting from an instantaneous source.

```
function c = cPuffInstant(p)
%CPUFF Concentration from puff source.
%   p is struct with input parameters.
%   Coordinates:
%       p.x (m)
%       p.y (m)
%       p.z (m)
%       p.t (s)
%   Source:
%       p.xSource, location of source in x dimension (m)
%       p.ySource, location of source in y dimension (m)
%       p.zSource, location of source in z dimension (m)
%       p.tEmission, time when puff occurs (s)
%       p.Q, amount of emission (particles or mass,
%            arbitrary unit) or emission flux
%           (particle or mass per time, a.u./s)
%       p.K, turbulent diffusion coefficient (m^2/s)
%   Room:
%       p.xRoom, length of the room in x dimension (m)
%       p.yRoom, width  of the room in y dimension (m)
%       p.zRoom, height of the room in z dimension (m)
%       p.a, ventilation rate (1/s)
%       p.wd, deposition rate (m/s)
```

```
%   About input dimensions:
%       Input dimensions for for x, y, z and t should be:
%       * They all are equal and output is elementwise
%         calculation
%       OR
%       * x, y and z are equal, but t is scalar
%       OR
%       * x, y and z are scalar and t is arbitrary.
%         Output will be same dimensions as t
%   System requirement: MATLAB R2017b or newer
%   Licence:  Artworks CC BY-SA
%             Code     Boost Software License -
%                      Version 1.0 - August 17th, 2003
%   Author:   Niko Leskinen

%checkInput(p);
% shift time such that emission starts at 0
p.t = p.t-p.tEmission;
c = firstTerm(p).*secondTerm(p).*...
    rTerm(p,'x').*rTerm(p,'y').*rTerm(p,'z');
% let c be zero if emission has not occurred
if isscalar(p.t) && p.t < 0+eps
    c = zeros(size(c));
else
    T = p.t;
    c(T<0+eps) = 0;
end
```

```
if any(isnan(c(:)))
    error('Error')
end
end

function checkInput(p)
inputFields = { 'x','y','z','t',...
    'xSource','ySource','zSource','tEmission','Q','K',...
    'xRoom','yRoom','zRoom','a','wd'};
for i = 1:numel(inputFields)
    inField = inputFields{i};
    assert(isfield(p,inField),sprintf(...
        'Input p.%s missing',inField));
end
assert(checkDimensions(p),'Check input dimensions')
end


function b = checkDimensions(p)
if isequal(size(p.x),size(p.y),size(p.z))
    if isequal(size(p.x),size(p.t))
        % All are equal, this is okay
        b = true;return;
    elseif isscalar(p.t)
        % time is scalar, this is okay too
        b = true;return;
    elseif isscalar(p.x) && isscalar(p.y) && isscalar(p.z)
```

```
        % spatial coordinates are scalar, this is okay too,
        % output will be same size as input time array
        b = true;return;
    else
        b = false;return
    end
else
    b = false;return
end
end


function value = firstTerm(p)
Q = p.Q;
K = p.K;
t = p.t;
value = Q./((4*pi*K*t).^1.5);
end


function value = secondTerm(p)
a = p.a;
wd = p.wd;
if a==0 && wd==0;value=1;return;end
xRoom = p.xRoom;
yRoom = p.yRoom;
zRoom = p.zRoom;
```

```
t = p.t;
% deposition surface
A = 2*(xRoom*yRoom + xRoom*zRoom + yRoom*zRoom);
% volume
V = xRoom*yRoom*zRoom;
% decayterm
value = exp(-a*t - wd*A*t/V);
end


function value = rTerm(p,dim)
switch dim
    case 'x'
        x = p.x;
        xRoom = p.xRoom;
        xSource = p.xSource;
    case 'y'
        x = p.y;
        xRoom = p.yRoom;
        xSource = p.ySource;
    case 'z'
        x = p.z;
        xRoom = p.zRoom;
        xSource = p.zSource;
    otherwise
        error('Incorrect dimension')
end
```

```
t = p.t;
K = p.K;
region = sqrt(2*K*t)/xRoom;
if isscalar(t)
    if true%region <= 0.64
        value = firstRegionApproximation(x,xRoom,xSource,K,t);
        return
    elseif 0.63 < region && region <= 1.08
        value = secondRegionApproximation(x,xRoom,xSource,K,t);
        return
    else
        value = ones(size(p.x)).* ...
            thirdRegionApproximation(xRoom,K,t);
        return
    end
else
    R1 = firstRegionApproximation(x,xRoom,xSource,K,t);
    R2 = secondRegionApproximation(x,xRoom,xSource,K,t);
    R3 = thirdRegionApproximation(xRoom,K,t);
    R = nan(size(t));
    R(region <= 0.64) = R1(region <= 0.64);
    R(0.63 < region & region <= 1.08) = ...
        R2(0.63 < region & region <= 1.08);
    R(1.08 < region) = R3(1.08 < region);
    value = R;
end
end
```

```
function value = firstRegionApproximation(x,xRoom,xSource,K,t)
R = 0;
for i = -10:10
    numeratorMinus = (x+2*i*xRoom-xSource).^2;
    numeratorPlus  = (x+2*i*xRoom+xSource).^2;
    denominator = 4*K*t;
    R = R + ...
        exp(- numeratorMinus./denominator) + ...
        exp(- numeratorPlus./denominator);
end
value = R;
end


function value = secondRegionApproximation(x,xRoom,xSource,K,t)
beta = exp(-pi^2*K*t/(xRoom^2));
termA = (2*sqrt(pi*K*t)/xRoom);
termB = (1-beta.^2);
termC = (1+beta.^2 + 2*beta.*...
    cos(pi*x/xRoom).*cos(pi*xSource/xRoom));
value = termA.*termB.*termC;
end

function value = thirdRegionApproximation(xRoom,K,t)
value = 2*sqrt(pi*K*t)/xRoom;
end
```

SI VI - Particle number size distribution data

Total particle number concentrations and size distributions for nGEN material printed in temperatures of 220, 240 and 250 °C. The comparison of PSM and SMPS data reveals that for size range over 3.15 nm, the PSM concentration is 1.4 to 3 times larger than that of SMPS. There are a few possible reasons for this difference. Firstly, the actual particle diameters (cutoff) might differ due to different measurement principles (PSM and SMPS), as shown by Olin et al.[3] Secondly, the PSM measures in a larger size range up to 1 µm. Thus, there might be another mode of particles between 60 nm to 1 µm. Thirdly, in some cases, the SMPS distribution has a tail that is out of the measurement range, thus it detects less particles. Lastly, for PSM, conducting sampling lines were used, while, for SMPS, the lines were low-conducting Tygon tubing, which might explain partly the difference if charged particles were present. However, an upper limit for the electric losses would be from 5 to 10 percent,[4,5] which is not enough to explain the concentration differences.

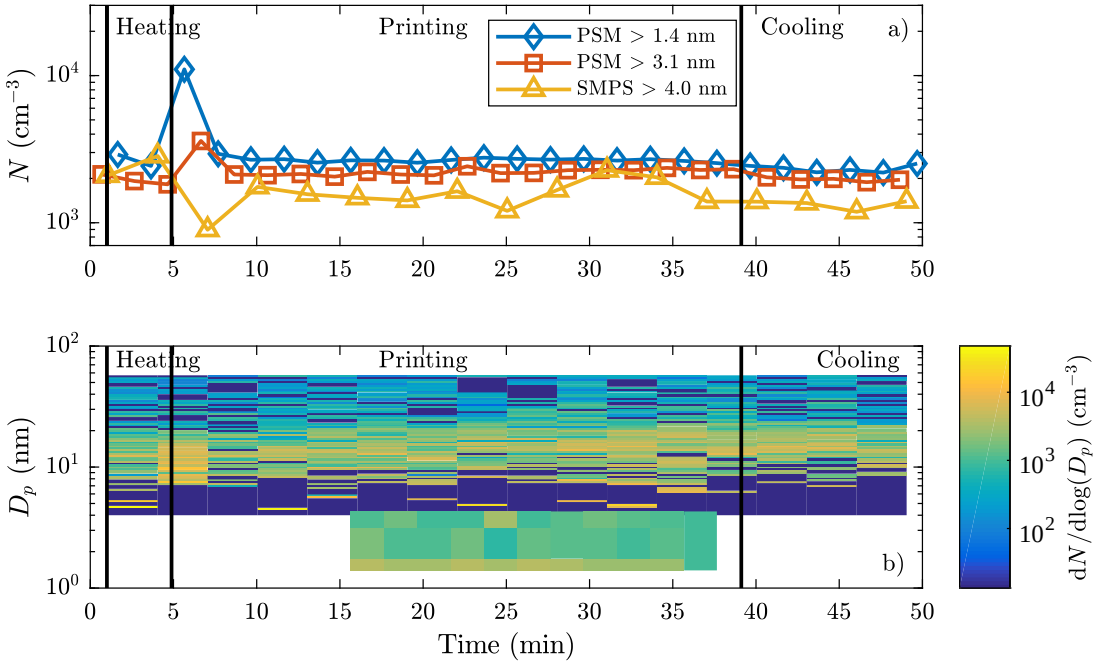

Figure S3: Total particle number concentration (a) and size distribution (b) in the chamber during a printing process of nGEN filament in temperature of 220 °C (print 30).

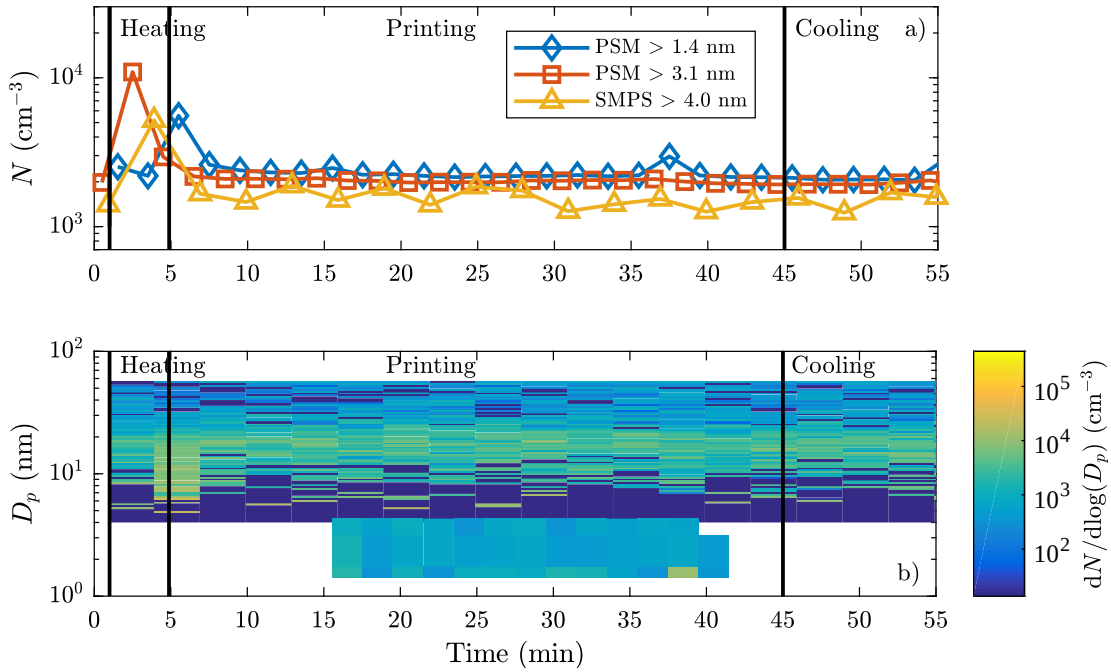

Figure S4: Total particle number concentration (a) and size distribution (b) in the chamber during a printing process of nGEN filament in temperature of 220 °C (print 31).

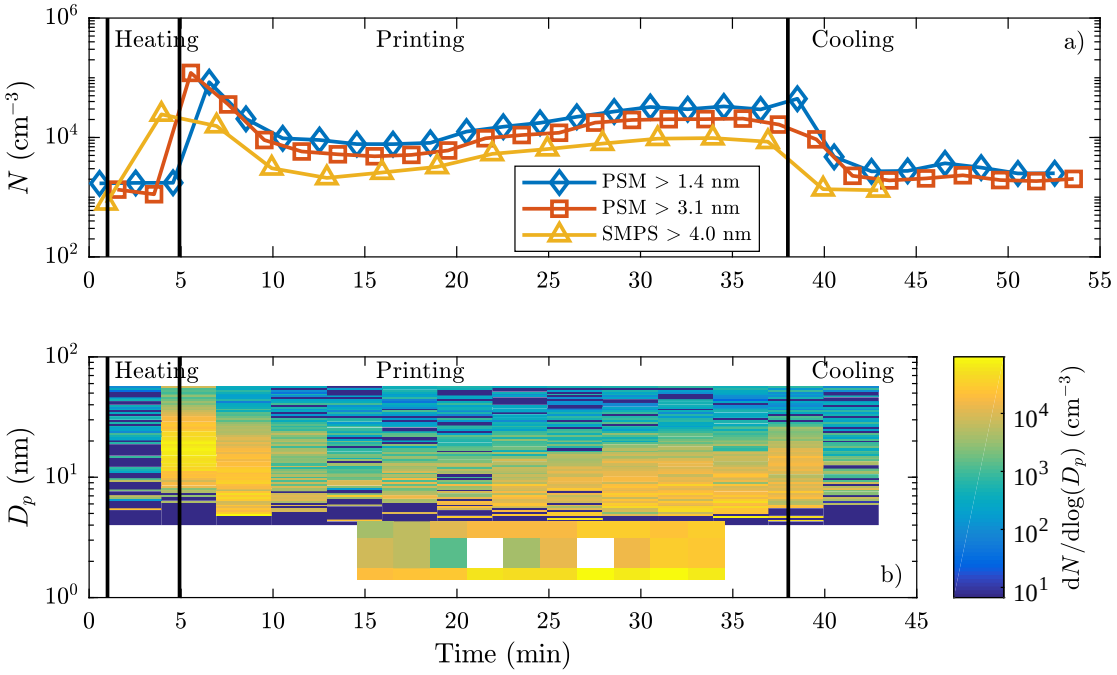

Figure S5: Total particle number concentration (a) and size distribution (b) in the chamber during a printing process of nGEN filament in temperature of 240 °C (print 28).

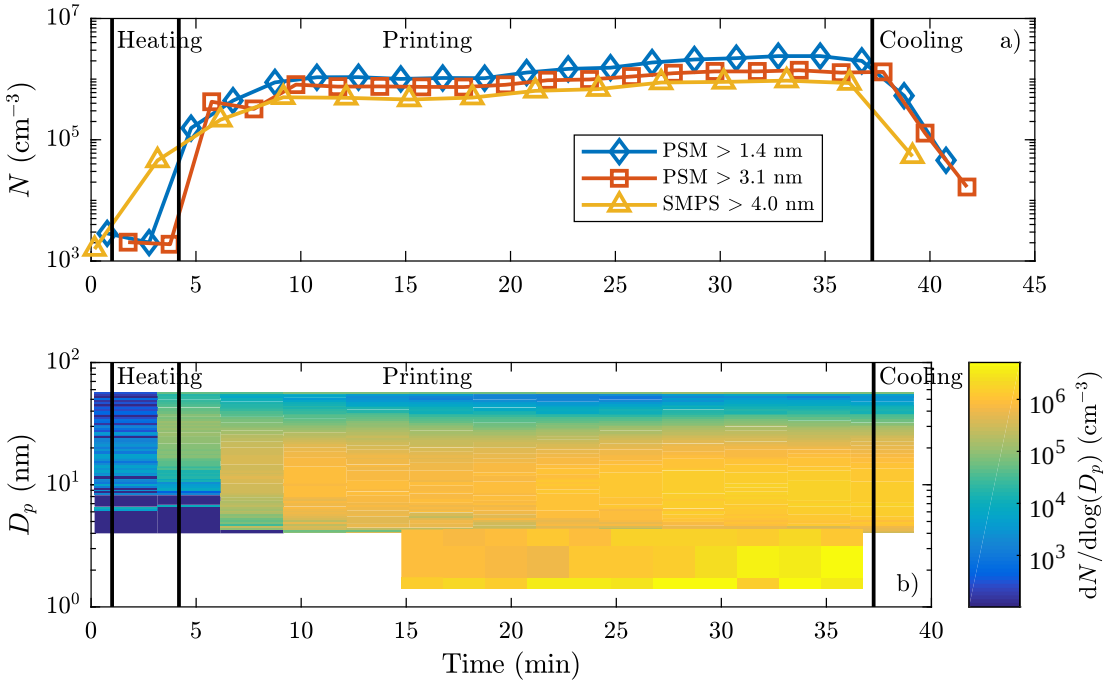

Figure S6: Total particle number concentration (a) and size distribution (b) in the chamber during a printing process of nGEN filament in temperature of 250 °C (print 32).

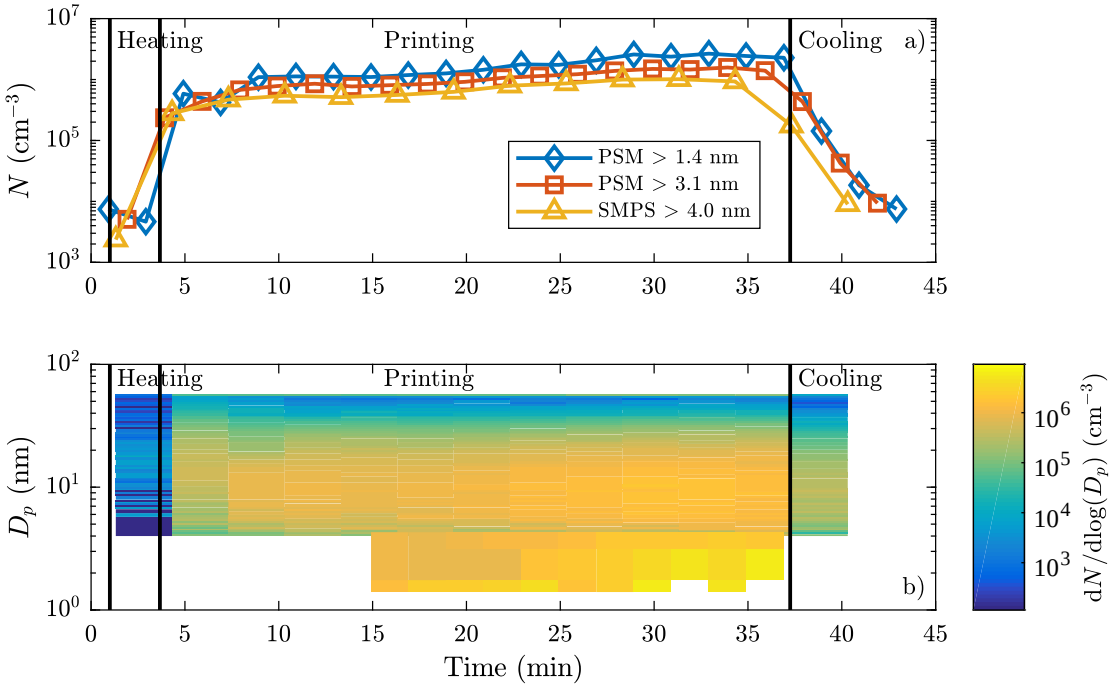

Figure S7: Total particle number concentration (a) and size distribution (b) in the chamber during a printing process of nGEN filament in temperature of 250 °C (print 33).

References

(1) Kangasluoma, J.; Junninen, H.; Lehtipalo, K.; Mikkilä, J.; Vanhanen, J.; Attoui, M.; Sipilä, M.; Worsnop, D.; Kulmala, M.; Petäjä, T. Remarks on Ion Generation for CPC Detection Efficiency Studies in Sub-3-nm Size Range. *Aerosol Science and Technology* **2013**, *47*, 556–563.

(2) Drivas, P. J.; Valberg, P. A.; Murphy, B. L.; Wilson, R. Modeling indoor air exposure from short-term point source releases. *Indoor Air* **1996**, *6*, 271–277.

(3) Olin, M.; Alanen, J.; Palmroth, M. R. T.; Rönkkö, T.; Dal Maso, M. Inversely modeling homogeneous H_2SO_4-H_2O nucleation rate in exhaust-related conditions. *Atmospheric Chemistry and Physics* **2019**, *19*, 6367–6388.

(4) Liu, B. Y. H.; Pui, D. Y. H.; Rubow, K. L.; Szymanski, W. W. Electrostatic effects in aerosol sampling and filtration. *The Annals of Occupational hygiene* **1985**, *29*, 251–269.

(5) Asbach, C.; Kaminski, H.; Lamboy, Y.; Schneiderwind, U.; Fierz, M.; Todea, A. M. Silicone sampling tubes can cause drastic artifacts in measurements with aerosol instrumentation based on unipolar diffusion charging. *Aerosol Science and Technology* **2016**, *50*, 1375–1384.