Supplementary information

New Peak Recognition Algorithm for Detection of Ultra Small Nano Particles with Single Particle ICP-MS Using Rapid Time Resolved Data Acquisition on a Sector-Field Mass Spectrometer

Jani Tuoriniemi, Geert Cornelis, Martin Hassellöv[*]

Department of Chemistry and Molecular Biology, University of Gothenburg, SE-412 96 Gothenburg, Sweden

[*]CORRESPONDING AUTHOR: martin.hassellov@chem.gu.se, tel. +46-31-786 9050, fax +46-31-77227856.

Table of contents

Description and source code of particle detection algorithm

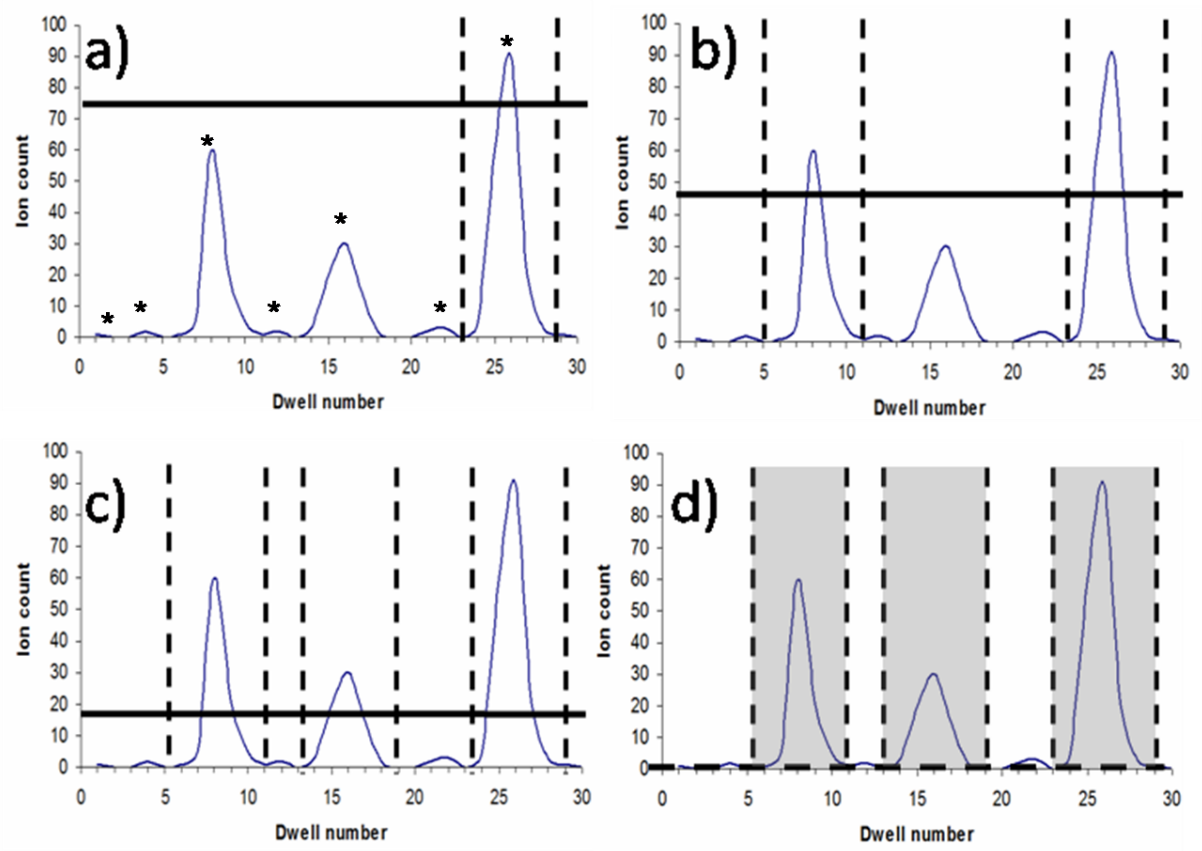

Figure ESI-1. The course of the particle detection algorithm. a) The potential particle events are identified by detecting zero crossings in derivative. They are marked by asterisks in the figure. The initial detection threshold, $I_{Threshi}$ (Horizontal line) is set, and it is exceeded by the rightmost particle event. The parameter w was set to 3, and the range consisting of the three data points surrounding the peak on each side (Marked by the vertical dashed line) are considered to be part of the particle event. b) The I_{Thresh} is set at the mean + n times the standard deviation of signal intensity the remaining dwells. The leftmost of the large particle events exceeds this threshold, and the $\pm w$ dwells surrounding the peak dwell are removed from consideration. c) The final I_{Thresh} is set when no additional particle events are detected with subsequent cycles of the algorithm. d) The particle events that pass the criteria set on the dwells adjacent to the peak dwell are integrated above the mean signal of those dwells that were deemed to contain dissolved signal only (Dashed horizontal line).

Cluster detection

The following sequences of dwells are smallest particle events that are recognized as such for the different values of the parameter M.

M=2 0:1:1:0, 0:2:0

M=3 0:1:1:1:0, 0:1:2:0, 0:3:0

M=4 0:1:1:1:1:0, 0:2:2:0, 0:1:2:1:0, 0:1:1:2:0, 0:1:3:0, 0:4:0

The detection efficiency as function of ion count and the parameter *M* There is always a risk for that a particle event containing *M* or more ions results in a sequence of dwells that is not recognized as such by the algorithm. The fraction of the particle events that are recognizable were calculated numerically, and are shown as function of I_{part} and *M* in Figure ESI-2.It was in these calculations assumed that the particle events are Gaussian bursts with a standard deviation 0.127 ms.

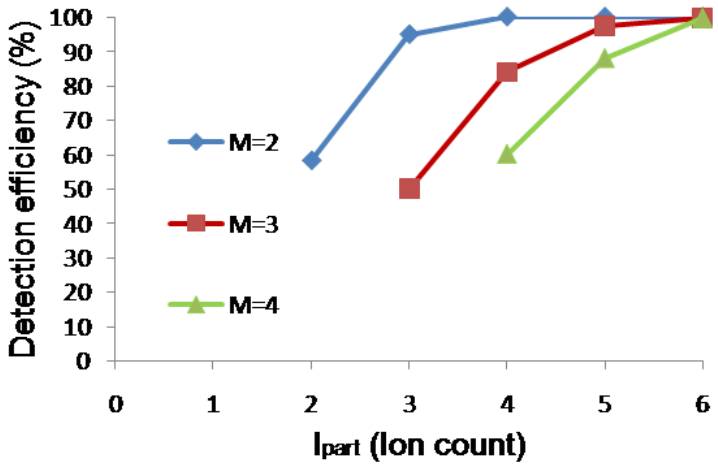

Figure SI-2. Detection efficiency due to the possibility of that particle events are not producing a sequence of dwells recognizable as a particle event.

There is a spread in the particle signals due to noise, and a particle with a size above the detection threshold may therefore also avoid detection if it is not producing the expected number of ions. For particle signals having an intensity of only a few ion counts, the noise is dominated by the shot noise. The magnitude of the shot noise is given by:

$$\sigma_{part}=\sqrt{\beta I_{part}} \qquad \text{(ESI-1)}$$

where β is a coefficient, that for the current experiments had value of 1.47. A more detailed discussion of noise in I_{part} has been given in previous work.[ESI-1]As an approximation, it can be assumed that the I_{part} produced by particles of a given size are Gaussian distributed around their mean value. In this case the overall detection efficiency can be calculated by multiplying the detection efficiencies in Figure ESI-2 with the fraction of the particles, having a certain expected I_{part} according to their size, that produce more than *M* ions. The overall detection efficiency as a function of *M* and I_{part} is shown in Figure ESI-3.

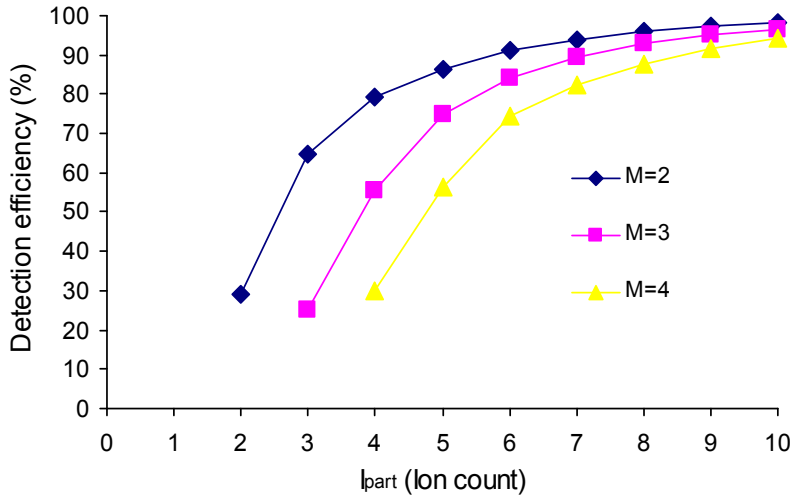

Figure SI-3 The overall detection efficiency as a function of the particle signal and parameter *M*.

Matlab source code with commentaries:

```
function [Intensities Threshold Pp Background]=Detect_peaks(A,w,t,s,n,M)
%  This is the main function. A is the vector with raw data. w, s, n, and M are
parameters explained in the main text. The parameter t is the number of consecutive
dwells acquired without any time spacing between them. The function returns
"Intensities" which is a list of the intensities of the particle signals."Threshold" is
the detection threshold, Pp is the list of positions of the particle events on the
vector "A". In order to obtain the particle count including those particle events that
were not integrated use the command ''length(Pp)'' after running the program.
"Background" is the mean intensity of the dissolved background signal.

[Threshold P]=Set_detectiontreshold(A,w,n); % This function iteratively determines the
detection threshold. 'P' is a preliminary list of particle positions on the vector A.
if Threshold>2  ;
[ Pp Background]=Inspect_particle_events_cluster_4(A,P,w,s); % If the detection
threshold determined previously exceeds 2 counts, the particles must fulfill the
criteria of M=4. The function returns the background signal intensity and list of peak
positions.
else % If the detection threshold is less than or equal to 2 ions then the particle
events need to fulfill the criteria of the M values set by the user.
if M==2
[ Pp Background]=Inspect_particle_events_cluster_2(A,P,w,s);
end
if M==3
[ Pp Background]=Inspect_particle_events_cluster_3(A,P,w,s);
end
if M==4
[ Pp Background]=Inspect_particle_events_cluster_4(A,P,w,s);
end
end
Intensities=peakintensities(A,Pp,w,t); % This function integrates the particle events
%*********************************************************
function [Threshold P]=Set_detectiontreshold(A,w,n) % This function sets the detection
threshold based on the parameter n. A preliminary set of particle events contained in
P(:,2) is detected and taken for further inspection.
As=sort(A); % The values in the raw data are sorted in ascending order.
Atrim=As((1):round(0.9*length(As))); % A dataset containing the 90 % of the dwells with
the lowest intensity is formed.
```

```
Threshold=median(A)+n*std(Atrim); % The initial threshold is calculated according to
description as the median of the whole dataset + n*std deviation of Atrim (n is chosen
by the user).
x=1:length(A); %The vector x=[ 1 2 3 …….Number of dwells] is formed.
f=0; % The initial value of the loop parameter f is set.
P=[0 0 0 0 0]; % The matrix for containing the positions of particle events is formed.
y=0; % The initial value of the parameter y is set.
while f==0 % The loop sets new detection thresholds until no more particle events are
found during subsequent runs.
P2=findpeaks_spICPMS(x,A,0.0,Threshold,1,1,1); % This function detects peaks based on
zero crossings in derivative.
[average stdbackground]=Stdbackgroundlevel(A,P2,w); % This function calculates the
mean, and standard deviations of the dissolved background.
if isnan(average)==1; % In the case there is no dissolved signal, its intensity is set
to zero.
average=0;
end
if isnan(stdbackground)==1; % In the case there is no dissolved signal, its standard
deviation is set to zero.
stdbackground=0;
end
Threshold=average+n*stdbackground;% The detection threshold is calculated as described
in the main text.
if length(P(:,2))>=length(P2(:,2))  % If number of particles is not larger than during
previous cycle of the while loop, it is terminated by setting f=1. The requirement y=1
ensures that the loop is not broken during its first cycle.
if y==1
    f=1;
end
end
P=P2; %The P is overwritten with the particle event positions detected at the current
cycle.
y=1; % The value of y is set to 1 to mark that it is not the first cycle of the loop.
end
%*********************************************************

%*********************************************************

function
P=findpeaks_spICPMS(x,y,SlopeThreshold,AmpThreshold,smoothwidth,peakgroup,smoothtype)
%This is a slightly modified version of the 'findpeaks' function by Tom O Haver. The
code and its commentaries are his, except for where indicated by**.
% function
P=findpeaks(x,y,SlopeThreshold,AmpThreshold,smoothwidth,peakgroup,smoothtype)
% Function to locate the positive peaks in a noisy x-y time series data
% set.  Detects peaks by looking for downward zero-crossings
% in the first derivative that exceed SlopeThreshold.
% Returns list (P) containing peak number and position,
% height, width, and area of each peak. Arguments "slopeThreshold",
% "ampThreshold" and "smoothwidth" control peak sensitivity.
% Higher values will neglect smaller features. "Smoothwidth" is
% the width of the smooth applied before peak detection; larger
% values ignore narrow peaks. "Peakgroup" is the number points
% around the top part of the peak that are taken for measurement.
% The argument "smoothtype" determines the smooth algorithm:
%   If smoothtype=1, rectangular (sliding-average or boxcar)
%   If smoothtype=2, triangular (2 passes of sliding-average)
%   If smoothtype=3, pseudo-Gaussian (3 passes of sliding-average)
% See http://terpconnect.umd.edu/~toh/spectrum/Smoothing.html and
% http://terpconnect.umd.edu/~toh/spectrum/PeakFindingandMeasurement.htm
% T. C. O'Haver, 1995.  Version 4.2, Last revised May, 2012
```

```
if nargin~=7;smoothtype=1;end% smoothtype=1 if not specified in argument
if smoothtype>3;smoothtype=3;end
if smoothtype<1;smoothtype=1;end
smoothwidth=round(smoothwidth);
peakgroup=round(peakgroup);
d=diff(y);% **The derivation function in the original version was changed to the
simpler diff command as it better suits the FAST spICPMS data sets.
n=round(peakgroup/2+1);
P=[0 0 0 0 0];
vectorlength=length(y);
peak=1;
AmpTest=AmpThreshold;
for j=1:(length(d)-1);
if sign(d(j)) > sign (d(j+1)) && sign(d(j))~=0, % Detects zero-crossing
if d(j)-d(j+1) >SlopeThreshold*y(j), % if slope of derivative is larger than
SlopeThreshold
if y(j+1) >AmpTest,  % if height of peak is larger than AmpThreshold
xx=zeros(size(peakgroup));yy=zeros(size(peakgroup));
for k=1:peakgroup, % Create sub-group of points near peak
groupindex=j+k-n+2;
if groupindex<1, groupindex=1; end
if groupindex>vectorlength, groupindex=vectorlength; end
xx(k)=x(groupindex);yy(k)=y(groupindex);
end
[coef,S,MU]=polyfit(xx,log(abs(yy)),2);  % Fit parabola to log10 of
%sub-group with centering and scaling
c1=coef(3);c2=coef(2);c3=coef(1);
PeakX=-((MU(2).*c2/(2*c3))-MU(1));   % Compute peak position and height of fitted
% parabola
PeakY=exp(c1-c3*(c2/(2*c3))^2);
MeasuredWidth=norm(MU(2).*2.35482/(sqrt(2)*sqrt(-1*c3)));
% if the peak is too narrow for least-squares technique to work
% well, just use the max value of y in the sub-group of points near peak.
if peakgroup<3,
PeakY=max(yy);
pindex=val2ind(yy,PeakY);
PeakX=xx(pindex(1));
end
% Construct matrix P. One row for each peak
% detected, containing the peak number, peak
% position (x-value) and peak height (y-value).
P(peak,:) = [round(peak) PeakX PeakY MeasuredWidth  1.0646.*PeakY*MeasuredWidth];
peak=peak+1;
end
end
end
end
% --------------------------------------------------------------------
function [index,closestval]=val2ind(x,val)
% Returns the index and the value of the element of vector x that is closest to val
% If more than one element is equally close, returns vectors of indicies and values
% Tom O'Haver (toh@umd.edu) October 2006
% Examples: If x=[1 2 4 3 5 9 6 4 5 3 1], then val2ind(x,6)=7 and val2ind(x,5.1)=[5 9]
% [indices values]=val2ind(x,3.3) returns indices = [4 10] and values = [3 3]
dif=abs(x-val);
index=find((dif-min(dif))==0);
closestval=x(index);

function d=deriv(a)
% First derivative of vector using 2-point central difference.
%  T. C. O'Haver, 1988.
```

```
n=length(a);
d(1)=a(2)-a(1);
d(n)=a(n)-a(n-1);
for j = 2:n-1;
d(j)=(a(j+1)-a(j-1)) ./ 2;
end

function SmoothY=fastsmooth(Y,w,type,ends)
% fastbsmooth(Y,w,type,ends) smooths vector Y with smooth
%  of width w. Version 2.0, May 2008.
% The argument "type" determines the smooth type:
%   If type=1, rectangular (sliding-average or boxcar)
%   If type=2, triangular (2 passes of sliding-average)
%   If type=3, pseudo-Gaussian (3 passes of sliding-average)
% The argument "ends" controls how the "ends" of the signal
% (the first w/2 points and the last w/2 points) are handled.
%   If ends=0, the ends are zero.  (In this mode the elapsed
%     time is independent of the smooth width). The fastest.
%   If ends=1, the ends are smoothed with progressively
%     smaller smooths the closer to the end. (In this mode the
%     elapsed time increases with increasing smooth widths).
% fastsmooth(Y,w,type) smooths with ends=0.
% fastsmooth(Y,w) smooths with type=1 and ends=0.
% Example:
% fastsmooth([1 1 1 10 10 10 1 1 1 1],3)= [0 1 4 7 10 7 4 1 1 0]
% fastsmooth([1 1 1 10 10 10 1 1 1 1],3,1,1)= [1 1 4 7 10 7 4 1 1 1]
%  T. C. O'Haver, May, 2008.
if nargin==2, ends=0; type=1; end
if nargin==3, ends=0; end
switch type
case 1
SmoothY=sa(Y,w,ends);
case 2
SmoothY=sa(sa(Y,w,ends),w,ends);
case 3
SmoothY=sa(sa(sa(Y,w,ends),w,ends),w,ends);
end

function SmoothY=sa(Y,smoothwidth,ends)
w=round(smoothwidth);
SumPoints=sum(Y(1:w));
s=zeros(size(Y));
halfw=round(w/2);
L=length(Y);
for k=1:L-w,
s(k+halfw-1)=SumPoints;
SumPoints=SumPoints-Y(k);
SumPoints=SumPoints+Y(k+w);
end
s(k+halfw)=sum(Y(L-w+1:L));
SmoothY=s./w;
% Taper the ends of the signal if ends=1.
if ends==1,
startpoint=(smoothwidth + 1)/2;
SmoothY(1)=(Y(1)+Y(2))./2;
for k=2:startpoint,
SmoothY(k)=mean(Y(1:(2*k-1)));
SmoothY(L-k+1)=mean(Y(L-2*k+2:L));
end
SmoothY(L)=(Y(L)+Y(L-1))./2;
end
```

```
% ------------------------------------------------------------------------

%************************************************************
function Intensities=peakintensities(A,Pp,w,t) % This function integrates the particle
events, and returns the vector "intensities" consisting of a list of particle signal
intensities.
aa=length(Pp); % aa is set to the number of particle events.
B=Pp; % The vector Pp containing the particle event positions is copied to B.
s=length(A) ;% s is set to the length of the raw data vector.
Intensities=[];% The intensities vector is created.
c=0; % The parameter c is set to its initial value 0.

Background=Meanbackgroundlevel(A,B,w); % The intensity of the dissolved background
signal is calculated.
for i=1:aa; % This loop goes through and integrates the particle events.
situation=0;% The variable "Situation" is set to 0.
if (B(i)-w)>0;
if (B(i)+w)<s || (B(i)+w)==s;

          c=1 ;% If the edge of the raw data vector is not located within +- w from the
peak dwell c is set to 1
end
end
if c==1 ;
if aa==1
                    c=4; % If the particle event is the only one, and it is not located
at the edge c is set to 4.
end
end
if c==1
if i==1
                    c=2; % If the particle event is the first one, and it is not located
at the edge c is set to 2.

end
end
if c==1
if i==aa
                    c=3; % If the particle event is the last one, and it is not located
at the edge c is set to 3.

end
end

if c==1; % The following if clauses investigate whether there is other particle events,
or a boundary of a continuous sequence of dwells within the range +-w from the peak
dwell. If so, the variable "situation" is set to 1. In this case for c=1, i.e. a
particle event that is neither the first nor the last one

if (B(i)-w)>B(i-1)+w;
if (B(i)+w)<B(i+1)-w;
for a=(B(i)-w):(B(i)+w)
if round(a/t)==(a/t)
situation=1;
end
end
if situation==0
                        I=sum(A((B(i)-w):(B(i)+w)));
                        I=I-(2*w+1)*Background;
```

```
                        Intensities=[Intensities I]; % If "situation" is 0, the
particle event is integrated as described in the main text.
end
end
end
end

if c==3 % The investigation described above is performed for the case c=3 (i.e.
particle event is the last one, but not at the edge)
if (B(i)-w)>B(i-1)+w;
for a=(B(i)-w):(B(i)+w)
if round(a/t)==(a/t)
situation=1;
end
end
if situation==0
                        I=sum(A((B(i)-w):(B(i)+w)));
                        I=I-(2*w+1)*Background;
                        Intensities=[Intensities I];
end
end
end
if c==2 % The investigation described above is performed for the case c=2 (i.e.
particle event is the first one, but not at the edge)

if (B(i)+w)<B(i+1)-w;
for a=(B(i)-w):(B(i)+w)
if round(a/t)==(a/t)
situation=1;
end
end
if situation==0
                  I=sum(A((B(i)-w):(B(i)+w)));
                  I=I-(2*w+1)*Background;
                  Intensities=[Intensities I];
end
end
end
if c==4 % The investigation described above is performed for the case c=4 (i.e. the
particle event is the only one)

            I=sum(A((B(i)-w):(B(i)+w)));
for a=(B(i)-w):(B(i)+w)
if round(a/t)==(a/t)
situation=1;
end
end
if situation==0
                        I=I-(2*w+1)*Background;
                        Intensities=[Intensities I];
end
end
         c=0;
end


%*********************************************************
function [ Pp Background] =Inspect_particle_events_cluster_2(A,P,w,s) % This function
ensures that the particle events fulfill the criteria set by M=2.
```

```
Pp=P(:,2); %The vector Pp is created and the preliminary list of particle events are
copied into it.
Background=Meanbackgroundlevel(A,Pp,w); %This function determines a preliminary value
of the dissolved signal intensity.
z=0; % The initial value of the parameter z is set.
Pp=round(Pp); % This rounds the values of Pp to the nearest integer values.
y=length(Pp); % Sets the parameter y to the number of particle events.
e=length(A); % Sets the parameter e to the number of dwells in the raw data vector.
i=1; % Sets the starting value of i.
while i<=y % The while loop goes through all the y particle events.
if Pp(i)<3
         z=1;
end
if Pp(i)>(e-3) % The particle events are not inspected, but marked for deletion by
setting z to 1 if they are too close to the edges of the vector A.
z=1;
end
if z==0
if A(Pp(i))<2;  % If the peak intensity is less than 2 counts they will undergo
examination.
if i<length(Pp);
if round(A(Pp(i)))==1 && Pp(i+1)<(Pp(i)+3) % This prevents the tails of large particle
events from becoming counted as separate particles.
                    z=1; % If the criteria is fulfilled the particle event is marked
for deletion.
end
end
if round(A(Pp(i)-1))<1 &&  round(A(Pp(i)+1))<1 % If the peak intensity is 1, and both
dwells surrounding the peak have zero intensity the particle event is marked for
deletion.

z=1;
end

end
end
if z==1
Pp(i)=[]; % The particle event is deleted from the preliminary list of particle events
if it has been marked for deletion (z=1) because of not fulfilling the criteria set by
the parameter M.
i=i-1;   % The value of i is decreased to compensate for the deletion.

end

     z=0;   % z is reset.
i=i+1;    % The loop parameter is increased.
y=length(Pp); % The parameter y is updated
end
   Background=Meanbackgroundlevel(A,Pp,w); % The updated mean level of the dissolved
background signal is calculated.
i=1;
while i<(length(Pp)+1)
if s>0 && ((A(Pp(i)-1)-Background)<(s*(A(Pp(i))-Background)) || (A(Pp(i)+1)-
Background)<(s*(A(Pp(i))-Background))); % If the parameter s has a non-zero value, and
either of the dwells surrounding the peak dwell don't exceed it, the particle event is
marked for deletion.

Pp(i)=[];

end
```

```
i=i+1;
end
function [ Pp Background]=Inspect_particle_events_cluster_3(A,P,w,s) % This function
ensures that the particle events fulfill the criteria set by M=3.

Pp=P(:,2);
Background=Meanbackgroundlevel(A,Pp,w);
z=0;
Pp=round(Pp);
y=length(Pp);
e=length(A);
i=1;
while i<=y
if Pp(i)<3
z=1;
end
if Pp(i)>(e-3)
z=1;
end
if z==0
if round(A(Pp(i)))<3; % The particle event is examined if the peak intensity is less
than 3 counts.
if round(A(Pp(i)))==1 && i<y
if i<length(Pp);
if Pp(i+1)<(Pp(i)+3) % This prevents the tails of  particle events from becoming
counted as separate particles.

                      z=1;

end
end
end
if round(A(Pp(i)))==2 && round(A(Pp(i)-1))<1 &&  round(A(Pp(i)+1))<1 % If the peak
intensity is 2 counts, but both dwells surrounding the peak have zero intensity, the
particle event is marked for deletion

                     z=1;

end
if round(A(Pp(i)))==1 && (round(A(Pp(i)+1))<1 || round(A(Pp(i)+2))<1) % If the peak
intensity is 1, and the particle event is not of the type 1:1:1, it is marked for
deletion.

Z=1;

end

end

end

if z==1
Pp(i)=[];
          i=i-1;
end
     z=0;
    i=i+1;
y=length(Pp);
```

```
end

Background=Meanbackgroundlevel(A,Pp,w);
i=1;
while i<(length(Pp)+1)
if s>0 && ((A(Pp(i)-1)-Background)<(s*(A(Pp(i))-Background)) || (A(Pp(i)+1)-
Background)<(s*(A(Pp(i))-Background))); % If the parameter s has a non-zero value, and
either of the dwells surrounding the peak dwell don't exceed it, the particle event is
marked for deletion.

Pp(i)=[];

end
i=i+1;
end
%************************************************************
function [ Pp Background] =Inspect_particle_events_cluster_4(A,P,w,s) %This function
ensures that the particle events fulfill the criteria set by M=4.

Pp=P(:,2);
Background=Meanbackgroundlevel(A,Pp,w);
z=0;
Pp=round(Pp);
y=length(Pp);
e=length(A);
i=1;
while i<=y
if Pp(i)<3
z=1;
end
if Pp(i)>(e-3)
z=1;
end
if z==0
if round(A(Pp(i)))<4;
if round(A(Pp(i)))==1 && i<y
if i<length(Pp);
if Pp(i+1)<(Pp(i)+3)
                    z=1;
end
end
end
if round(A(Pp(i)))==2 && (round(A(Pp(i)-1))<1 || round(A(Pp(i)+1))<1) % If the peak
intensity is 2, and any of the  surrounding dwells have zero intensity, the inspection
is taken to the next step on the line below.

if (round(A(Pp(i)-1))+round(A(Pp(i)-2)))<2 && (round(A(Pp(i)+1))+round(A(Pp(i)+2)))<2
% If there is not two consecutive dwells with non-zero ion count on either size of the
peak dwell the particle event is marked for deletion.


z=1;
end
end

if round(A(Pp(i)))==1 && (round(A(Pp(i)+1))+round(A(Pp(i)+2))+round(A(Pp(i)+3)))<3 % If
the peak intensity is 1, there must be 4 consecutive dwells with non-zero intensity.
```

```
Z=1;

end

if  round(A(Pp(i)))==3 && (round(A(Pp(i)-1))<1 &&  round(A(Pp(i)+1))<1) % If the peak
intensity is 3, either of the adjacent dwells must have non zero intensity.


z=1;

end
end

end

if z==1
Pp(i)=[];
         i=i-1;
end
    i=i+1;
z=0;
    y=length(Pp);


end
  Background=Meanbackgroundlevel(A,Pp,w);
i=1;
while i<(length(Pp)+1)
if s>0 && ((A(Pp(i)-1)-Background)<(s*(A(Pp(i))-Background)) || (A(Pp(i)+1)-
Background)<(s*(A(Pp(i))-Background))); % If the parameter s has a non-zero value, and
either of the dwells surrounding the peak dwell don't exceed it, the particle event is
marked for deletion.

Pp(i)=[];

end
i=i+1;
end

%*************************************************************
function Background=Meanbackgroundlevel(A,Pp,w)% Determines the level of the dissolved
background signal.
B=Pp;
aa=length(B); % Sets the value of aa to the number of particle events.

s=length(A);% Sets the value of s to the length of the raw data vector.

for i=1:aa% This loop marks datapoints located within +- w from the peak dwell as not a
number, NaN.

if (B(i)-w)<1;
         A(1:(B(i)+w))=NaN; % If the particle event is located at the beginning of the
raw data vector.
elseif (B(i)+w)>s;
A((B(i)-w):s)=NaN; %If the particle event is located at the end of the raw data vector.

else
```

```
A((B(i)-w):(B(i)+w))=NaN; %This takes care of all other cases when the particle event
is not close to the edge.
end
end
i=1;                % Creates variable i
while i<=length(A);     % This makes "i" go from 1 to the length of the raw data vector
if isnan(A(i))==1;      % Checks if the datapoint in position i is NaN
A(i)=[];                   % If the above is true the datapoint is deleted.
else
i=i+1;                  %Otherwise the value of i is just increased.
end
end
Background=sum(A)/length(A);% The mean level of the signal in the dwells not considered
to contain any particle signal is calculated.
%************************************************************
function [average, stdbackgroundlevel]=Stdbackgroundlevel(A,P,w)
B=round(P(:,2));
aa=length(B);

s=length(A);

for i=1:aa % The loop marks the range +-w surrounding the particle events as NaN.

if (B(i)-w)<1;
A(1:(B(i)+w))=NaN;
elseif (B(i)+w)>s;
A((B(i)-w):s)=NaN;

else
A((B(i)-w):(B(i)+w))=NaN;
end
end
i=1;                % Creates variable i
while i<=length(A);     % This makes "i" go from 1 to the length of the data vector. The
dwells within +-w of the peak dwell are deleted.
if isnan(A(i))==1;      % checks if the datapoint in position i is NAN
A(i)=[];                   % If the above is true the datapoint is deleted.
else
i=i+1;                  %Otherwise the value of iis just increased.
end
end
stdbackgroundlevel=std(A);% Calculates a preliminary value of the background standard
deviation.
average=mean(A); % Calculates a preliminary value of the mean of the background signal.
% The trimming procedure facilitates convergence of the detection limit towards values
determined by dissolved background only in samples having high particle frequency.
As=sort(A);
AA=average+3*stdbackgroundlevel; % A 3*σ Cut off is set.
M=length(As);
for i=1:length(As) % The loop removes dwells above the cut off
if As(i)>=AA
        M=i;
break
end

end
stdbackgroundlevel=std(As(1:M)); % New values for the standard deviation and mean of
the dissolved background are calculated using the trimmed data set
average=mean(As(1:M));
```

Validation of the mathematical model for the dependence of DL_s on dwell time

In Figure ESI-4 the size detection limits predicted by Eqn 5 in the main text are compared with values determined by the algorithm (n=5, w=3). The M_{min} determined by the algorithm (Experimental), and Eqn 5 in the main text (Calculated) is shown as function of the dissolved Ag concentration (Determined from the I_{diss} determined by the algorithm.) in different dilutions of the 40 nm AgNP colloid (106 000, 53 000, 15 900, 10590 particles mL^{-1}). The experimental and calculated values agree for the lowest concentrations of dissolved silver. There are particles present with sizes below the DL_s. They make the background noise to seem higher than what could be predicted by Eqn 1 in the main text for the two samples with higher concentrations of dissolved Ag. Therefore the difference between the experimental and theoretically calculated values of M_{min}, increases from ~0 for the lowest concentrations, to around two times the experimental values for the two higher concentrations. However, in this case the M_{min} are converted to equivalent spherical diameters, the differences between calculated and actual values are reduced to their third root, which is 25% in this case. The model can therefore be regarded accurate enough to be useful.

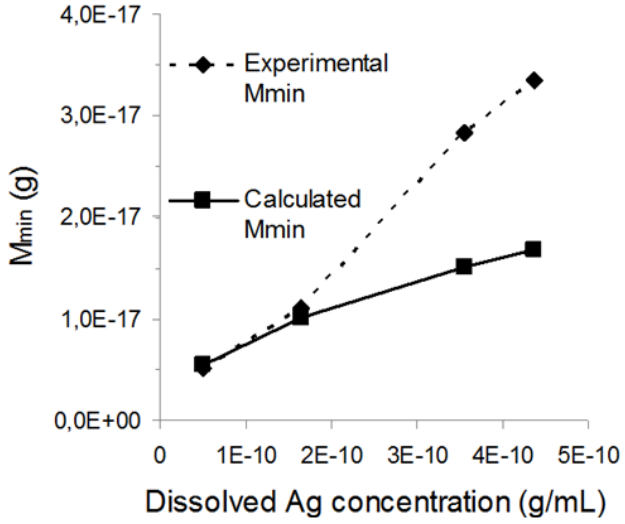

Figure ESI-4. The M_{min} determined experimentally, and predicted by Eqn 5 in the main text.

List of references

ESI-1. Tuoriniemi, J.; Cornelis, G.; Hassellöv, M. *J. Anal. At. Spectrom.*, 2014, 29, 743-752