Supplementary Material for

Minimising Damage in High Resolution Scanning Transmission Electron Microscope Images of Nanoscale Structures and Processes

Daniel Nicholls[1], Juhan Lee[1,2], Houari Amari[1,2], Andrew J. Stevens[3,4], B. Layla Mehdi[1,2,5], Nigel D. Browning[1,2,3,5]

[1] Department of Mechanical, Materials and Aerospace Engineering and Department of Physics, University of Liverpool, Liverpool, L69 3GH

[2] The Faraday Institution, Quad One, Harwell Science and Innovation Campus, Didcot OX11 0RA, United Kingdom

[3] Sivananthan Laboratories, 590 Territorial Drive, Bolingbrook, IL 60440. USA

[4] OptimalSensing LLC, Southlake, TX 76092. USA

[5] Physical and Computational Science Directorate, Pacific Northwest National Laboratory, Richland, WA 99352, USA

The supplemental material includes information regarding equivalent dose sampling, the benefits of line hop scanning over line separated scanning, how the reconstructed images in the main manuscript were analysed and compared, and a condensed version of the MATLAB code used in this investigation.

Equivalent Dose Sampling

In Figure 2a of the main manuscript two datasets are presented; one utilising a single pass method and the other using a lap scanning method. The purpose of the lap scanning method is, as stated, to show that the reduction in maximum beam influence is due to the step separation and not the reduced dose. While this is presented as part of the dwell time investigation in the manuscript, the following figures (S1 and S2) show the same method applied to both the thickness and diffusion coefficient investigations. These figures show how the lap scanning method, which is functionally analogous to compressive sensing STEM video, performs with decreasing sampling percentage. As the sampling percentage decreases the number of laps required to keep the dose equivalent increases. After a point, increasing the step separation provides negligible decreases to the line-to-line overlap, but increases the scan-to-scan overlap, causing a rise in the maximum beam influence as sampling percentage decreases. This implies that lower is not always better, and that for equivalent dose sampling there exists an optimum step separation.

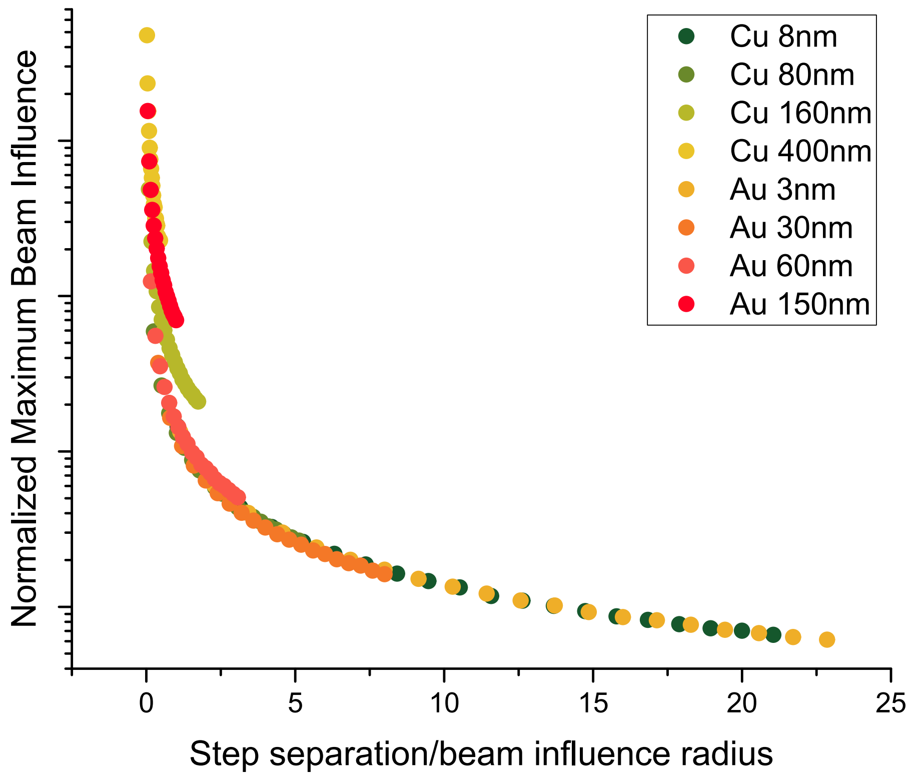

Figure S1. Equivalent dose sampling shows a deviation from the curve presented in figure 2b. This is due to the presence of scan-to-scan overlap, and the effect is more prevalent in higher step separations as more laps are performed.

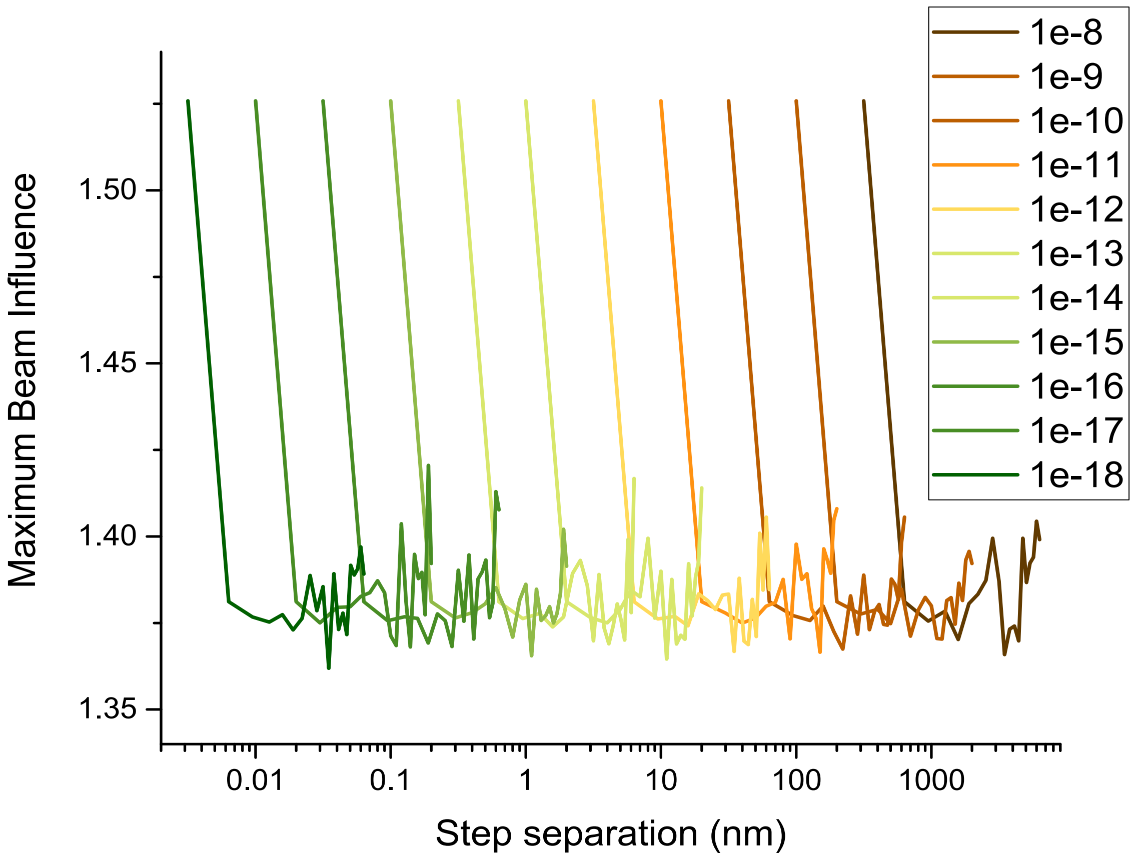

Figure S2. The equivalent dose scan variant of figure 2c shows a large variance in the maximum beam influence as the sampling percentage decreases. Similar to figure S1 this is due to scan-to-scan overlap.

Line Hop Scanning

A question that often comes up is why line hop scanning is used over a sampling scheme in which straight scan lines are separated, as shown in figure S3. Referring to the main text, the distinction is made between the kinds of beam overlap – point-to-point, line-to-line, and scan-to-scan. While both line hop and line separated scanning reduces the amount of line-to-line overlap, line hop scanning provides a reduction to the other two types due to the random variation perpendicular to the scan direction. Scan-to-scan overlap is reduced as there is a reduced chance that the same pixel will be sampled multiple times. Point-to-point overlap is reduced due to the average distance between successive on-path pixels increasing, which is explainable by Pythagoras' theorem and is shown in figure S4. This effect is also accumulative as each point-to-point overlap reduction applies to future scan positions along the same path as well. In figure S3 it is apparent that there are some locations during the line hop scan where successive lines become close to one another. While this will result in an increase in the overlap from the next line, it also means a reduction in overlap from the previous line, and so over a number of lines this effect averages out.

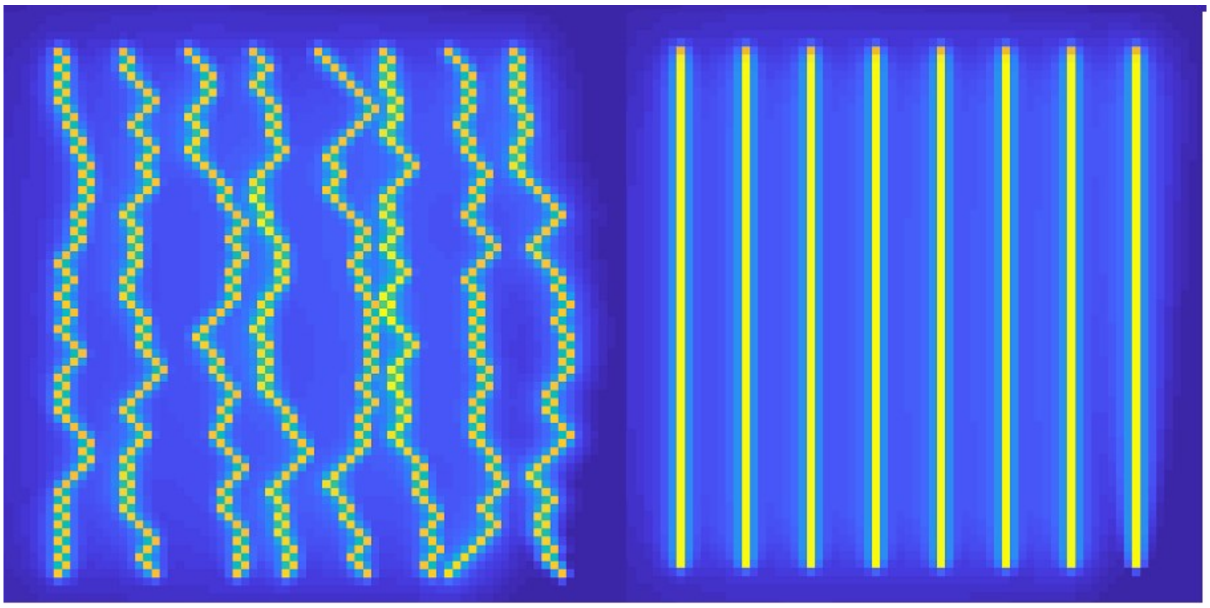

Figure S3. A 12.5% line hop scan (left) and the equivalent line separated scan (right).

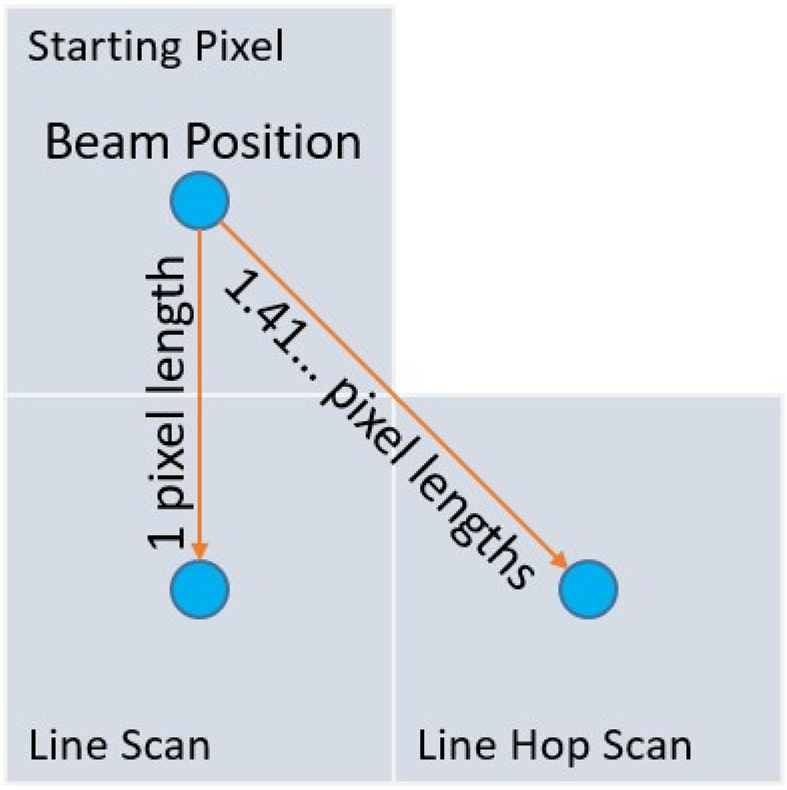

Figure S4. Jumping diagonally, as in line hop sampling, increases the pixel-to-pixel jump distance from 1 unit to 1.4142 units (square root of 2).

Image Comparison

The quality of figure 4c was validated by comparison against figure 4a using two metrics; peak signal-to-noise ratio (PSNR) and maximum cross correlation (CC). Both the PSNR and CC were calculated using MATLABs native functions, Peak Signal-to-Noise Ratio (https://uk.mathworks.com/help/images/ref/psnr.html), and the Normalized 2-D cross-correlation (https://uk.mathworks.com/help/images/ref/normxcorr2.html).

MATLAB Code

A condensed version of the MATLAB code has been made available in this document that is able to replicate the data shown in the main manuscript, as well as the MATLAB files required to run the code. Copy and paste the following texts into MATLAB and save as beam_overlap.m and fickdiff.m respectively.

```
%----------------------------------------------------------------------%
% Beam_overlap.m - A study of how changing STEM sampling schemes affects
% beam influence generation in homogenous monoatomic films.
%
% Other m-files required: fickdiff.m
%
% Instructions:
%     Input desired microscope and sample parameters. For line hop sampling
%     investigation, choose beam_type = 300. For all beam_types, sampling
%     percentage is determined by 100/beam_step_y. Without modification,
%     this program performs a series of line hop exposures with increasing
%     step separation (from 1nm to 20nm, 100% to 5% sampling). Changing dt
%     and dx allows the user to scale the system appropriately for the
%     diffusion coefficient they are interested in, quicker diffusion
%     requires larger space steps/smaller time steps to be stable
%     mathematically. Raster and random scanning options are available to
%     compare line hop sampling against.
%
%     By plotting max_BI against step_sep this program can be used to
%     determine the optimum sampling percentage required to image a sample
%     at the defined microscope parameters to reduce the maximum beam
%     influence generated during the scan by minimising the amount of
%     beam overlap that takes place.
%
%  Main parameters are:
%     max_step_separation   - largest step separation to cycle to (in pixels)
%     dt, dx                - time and space steps
%     beam_type             - type of sampling scheme implemented
%     D                     - diffusion coefficient of material
%     dwell_time            - dwell time in us
%     thickness             - sample thickness in nm
%     ZN                    - Z number
%
% Outputs:
%     step_sep              - step separations used in calculations
%     max_BI                - maximum beam influence of a scan at
%                             incremental step separations
%
% Author: Daniel Nicholls
% email: d.nicholls@liverpool.ac.uk
% October 2019; Last revision: 31-Oct-2019
%----------------------------------------------------------------------%
plotting = 0;                               %live plotting (slow)
                                            %0 = off, 1 = on
```

```
max_step_separation = 30;                    %maximum step separation (in pixels) (d)
max_BI = zeros(max_step_separation,1);         %preallocation for saving the max. beam influence
(BI)
step_sep = zeros(max_step_separation,1);       %preallocation for saving the step separation values

dt = 1e-6;                              %1*10^-6 for microsecond timestep. Lower values for higher
accuracy data.
dx = 1e-9;                               %space step
dx2 = dx^2;                               %space step squared
                                        %change dt and dx appropriately to
                                        %balance accuracy and sim runtime.



%--Microscope Properties-------------------------------------------------%
beam_type = 200;                              %100 = raster
                                        %200 = random
                                        %300 = lane constrained random lh
dwell_time = 1;                           %dwell time in microseconds (usec)
dose_rate = 1*(dt/1e-6);                      %1 'dose' per us
E0 = 200.;                              %accelerating voltage in keV
%--Material Properties---------------------------------------------------%
D = 1 * 10^(-13);                         %diffusion coefficient
thickness = 3;                            %sample thickness (nm)
ZN = 29;                                %sample atomic number
lattice_p = 3.6149 * 10^(-10);                 %lattice constant (m)
lattice_n = 4;                           %number of atoms per unit cell
Nv = lattice_n/(lattice_p^3);                 %number of atoms per m^3
%--Space Initilisation---------------------------------------------------%
X = 256;                                %size of array in X
Y = X;                                 %size of array in Y
Z = floor(thickness * 1e-9/dx);               %sample thickness in pixels
                                        %use these to control region of
                                        %interest size. Increases
                                        %computational power exponentially.

sinkco = 0.5;                             %relative size of unradiated area...
aoi = floor(X * (1-sinkco));                   %...between sample and heat sink
sinksize = X * sinkco;                        %size of sink used for beam positioning

dwell_time = dwell_time * (1e-6/dt);           %dwell time in dt units
DPPL = D * (dt/dx2);                         %stability criteria
DCH = 6*DPPL;
%--Beam Broadening Equations---------------------------------------------%
t_z = Z * dx;                             %foil thickness (m)
B = 8 * 10^(-12) * (ZN / E0) * (Nv)^(0.5) * t_z^(1.5);
                                        %from TEM - Williams &
                                        %Carter for reference only
C = 8 * 10^(-12) * (ZN / E0) * (Nv)^(0.5);      %used in later calculation
%------------------------------------------------------------------------%
%
%
%
%          Nothing beyond this point requires changing.
%
%
%
%------------------------------------------------------------------------%
for var1 = 1:1:max_step_separation            %cycle from d = 1 to d = max_step_separation
```

```
    beam_step_y = var1;                     %distance between successive line scans in y
    if DCH > 1                          %break if unstable
        disp("CONDITIONALLY UNSTABLE")
        disp("CONDITIONALLY UNSTABLE")
        disp("CONDITIONALLY UNSTABLE")
        disp("CONDITIONALLY UNSTABLE")
        disp("CONDITIONALLY UNSTABLE")
        break
    end
%--Scan Start Position--------------------------------------------------%
    ix = ceil(1+(sinksize/2));              %x start position
    iy = ceil(1+(sinksize/2));              %y start position
    iy_home = ceil(1+(sinksize/2));             %beam home position in y
%--Preallocation-------------------------------------------------------%
    main = zeros(X,Y,Z);                    %preallocated array
    dcrt = main;
    lapm = 0;                           %lap marker
    lapch = 0;                          %lap checker
%--Time Initilisation--------------------------------------------------%
    ttot = (aoi) *  aoi * dwell_time * (1/(beam_step_y));
                                        %estimated runtime (just for
                                        %self referencing)
    tt = 0;                              %dwell time tracker
    time = 0;                              %time initialise

    max_a = zeros(ceil(ttot),1);
    max_i = zeros(ceil(ttot),1);
%--Simulation Start----------------------------------------------------%
    while lapm == 0                         %perform 1 scan

        disp([time/ttot var1])              %progress tracker
        if lapch == 1
            disp([lapm ttot])
            lapch = 0;
        end                     %lap checker

        tt = tt + 1;                        %dwell time checker
        time = time + 1;                       %sim time (in dt)
%--Beam at position----------------------------------------------------%
        ix = ceil(ix);                      %center probe
        iy = ceil(iy);                      %center probe
        for dZ = 1:Z
            dB = C * ((dZ-1) * dx)^(1.5);       %Beam broadening in nm
            dB = dB * 1/dx;                 %Beam broadening in pixels
            R = floor(floor(dB)/2);
        %beam assumed to be approximately circular, symettrical and uniform
            if floor(R) == 0
                main(ix,iy,dZ) = main(ix,iy,dZ) + dose_rate;
            else
                for rx = 1-R:R
                    for ry = 1-R:R
                        if sqrt(rx^2 + ry^2) <= R
                            main(ix+rx,iy+ry,dZ) = main(ix+rx,iy+ry,dZ) + dose_rate/(pi * length(1-R:R));
                        end
                    end
                end
            end
        end                     %sample irradiation
%--Raster Scan---------------------------------------------------------%
        if beam_type == 100
```

```
            if tt == dwell_time
                ix = ix + 1;
                if ix > X-(sinksize/2)
                   iy = iy + (beam_step_y);
                   ix = ceil(1+(sinksize/2));
                   if iy > Y-(sinksize/2)
                       iy = ceil(1+(sinksize/2));
                       lapm = lapm + 1;
                       lapch = 1;
                   end
                end
                tt = 0;
            end
        end              %generate next [ix iy] raster
%--Random Sampling----------------------------------------------------%
        if beam_type == 200
            if tt == dwell_time
                ix = randi([ceil(1+(sinksize/2)) ceil(X-(sinksize/2))],1);
                iy = randi([ceil(1+(sinksize/2)) ceil(Y-(sinksize/2))],1);
                tt = 0;
            end
            if mod(time,floor(aoi * aoi * dwell_time * (1/beam_step_y))) == 0
                lapm = lapm + 1;
                lapch = 1;
            end
        end              %generate next [ix iy] random
%--Lane Constrained Linehop Sampling-------------------------------------%
        if beam_type == 300
            if ix > X-(sinksize/2)
                iy_home = iy_home + beam_step_y;
                if iy_home > Y-(sinksize/2)
                    iy_home = 1+ceil((sinksize/2));
                    ix = 1+ceil((sinksize/2));
                    lapm = lapm + 1;
                else
                    ix = 1+ceil(sinksize/2);
                    iy = iy_home;
                    tt = 0;
                end
            else
                if tt == dwell_time
                    ix = ix + 1;
                    variance = iy - iy_home;
                    if variance == 0 && var1 > 1
                        iy = iy + 1;
                    elseif variance == 0 && var1 == 1
%                           iy = iy;
                    elseif variance == var1-1
                        iy = iy - 1;
                    else
                        switch randi([1 2],1)
                            case 1
                                iy = iy - 1;
                            case 2
                                iy = iy + 1;
                        end
                    end
                end
                tt = 0;
            end
        end                 %generate next [ix iy] linehop
%--Transfer Action (Fick's Law)-----------------------------------------------%
        alpha = fickdiff(main,dx2,dt,D,X,Y,Z); %Fick's Law diffusion calc.
        main = main + alpha;
%--Heat Sink-------------------------------------------------------------------%
        main(1,:,:) = 0;                        %sink at X and Y edges (not top surface)
        main(X,:,:) = 0;
        main(:,1,:) = 0;
        main(:,Y,:) = 0;
%--Max at all space-----------------------------------------------------------%
        for x = 1:X
          for y = 1:Y
            for z = 1:Z
              if main(x,y,z) > dcrt(x,y,z)
                dcrt(x,y,z) = main(x,y,z);
              end
            end
          end
        end                     %records maximum BI at all positions
%--Data Acqusition------------------------------------------------------------%
        max_main = max(max(max(main)));        %calculating maximum BI
        if max(max(max(main))) > max_main
          max_main = max(max(max(main)));
        end

        max_a(time) = max_main;                %record max BI
        max_i(time) = time*dt;                %record time series
%--Plotting--------------------------------------------------------------------%
        if plotting == 1                    %live plots on or off
    %--Flat Heatmap------------------------------------------------------------%
          figure(1)
          subplot(2,2,1)
          imagesc(main(:,:,1))        %live plot top surface
          axis square
          title("Top surface")
          subplot(2,2,2)
          imagesc(main(:,:,end))      %live plot bottom surface
          axis square
          title("Bottom surface")
          subplot(2,2,3)
          imagesc(dcrt(:,:,1))
          axis square
          title("Top surface - max")
          subplot(2,2,4)
          imagesc(dcrt(:,:,end))
          axis square
          title("Bottom surface - max")
        end
    end

    max_BI(var1) = max(max_a);
    step_sep(var1) = var1*(dx/1e-9);

    figure(1)
    clims = [0 max(max(max(main)))];
    subplot(1,2,1)
    imagesc(main(:,:,1),clims);
    axis square
    title("Top surface")
    subplot(1,2,2)
    imagesc(main(:,:,end),clims);
    axis square
    title("Bottom surface")

    figure(2)
    subplot(1,2,1)
    plot(max(dcrt(:,:,end)))
    title("Maximum Beam Influence Profile over all sim time")
    hold on
    axis square

    subplot(1,2,2)
    plot(max(main(:,:,end)))
    title("Maximum Beam Influence Profile al final sim time")
    hold on
    axis square

    figure(3)
    imagesc(dcrt(:,:,end))
    axis square
    set(gca,'YTickLabel',[]);
    set(gca,'XTickLabel',[]);
end

filename_ml = strcat('dwell_time_',num2str(dwell_time*dt/1e-6),'_us_max_BI','.mat');
filename_lv = strcat('dwell_time_',num2str(dwell_time*dt/1e-6),'_us_step_sep','.mat');
save(filename_ml,'max_BI');
save(filename_lv,'step_sep');

%--Script End------------------------------------------------------------%
```

```
function [A1] = fickdiff(main,dx2,dt,D,X,Y,Z)
%------------------------------------------------------------------------%
% fickdiff.m - Diffusion mechanism utilised in Beam_overlap.m
%
% Author: Daniel Nicholls
% email: d.nicholls@liverpool.ac.uk
% October 2019; Last revision: 31-Oct-2019
%------------------------------------------------------------------------%
   alpha = zeros(X,Y,Z);

   for h = 1
      for k = 2:Y-1
         for l = 2:Z-1
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k-1,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l-1) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h+1,k,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k+1,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l+1) - main(h,k,l));
         end
      end
   end

   for h = X
      for k = 2:Y-1
         for l = 2:Z-1
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h-1,k,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k-1,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l-1) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k+1,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l+1) - main(h,k,l));
         end
      end
   end

   for h = 2:X-1
      for k = 1
         for l = 2:Z-1
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h-1,k,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l-1) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h+1,k,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k+1,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l+1) - main(h,k,l));
         end
      end
   end

   for h = 2:X-1
      for k = Y
         for l = 2:Z-1
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h-1,k,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k-1,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l-1) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h+1,k,l) - main(h,k,l));
            alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l+1) - main(h,k,l));
         end
      end
   end

   for h = 2:X-1
```

```
        for k = 2:Y-1
            for l = 1
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h-1,k,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k-1,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h+1,k,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k+1,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l+1) - main(h,k,l));
            end
        end
    end

    for h = 2:X-1
        for k = 2:Y-1
            for l = Z
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h-1,k,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k-1,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l-1) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h+1,k,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k+1,l) - main(h,k,l));
            end
        end
    end

    for h = 2:X-1
        for k = 2:Y-1
            for l = 2:Z-1
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h-1,k,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k-1,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l-1) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h+1,k,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k+1,l) - main(h,k,l));
                alpha(h,k,l) = alpha(h,k,l) + D * (dt/dx2) * (main(h,k,l+1) - main(h,k,l));
            end
        end
    end
    A1 = alpha;
end

%--Script End------------------------------------------------------------%
```