Module 2: A clever way to spot the crack propagation for neo-hookian solids in Mode-I fracture testing using Matlab

Gaëtan Cortes, Damien Delespaul, and Naïm Sabaghi
ME-412 : Experimental Methods in Engineering Mechanics, École Polytechnique Fédérale de Lausanne
(Dated: November 26, 2023)

During the DIC analysis of a Mode-I fracture test for a Neo-Hookian solid, numerous images can be generated as a function of the spreading speed, but also as a function of the material's elasticity and toughness. To calculate the J-integral, only the image just before fracture propagation is required. It is therefore necessary to analyse the images one by one to determine exactly where the crack is propagating. This paper proposes a simplified way of determining the image at which the crack begins to propagate, for a highly elastic solid. The software used for this method is NCORR, and a fairly qualitative DIC is required to make this method work. All the calculation and coding are done with Matlab R2023b.

The method described here uses the displacement gradient, in the direction perpendicular to the crack, i.e. the direction in which the sample elongates during the test. Given that the sample elongates at a constant speed, the evolution of the mean strain gradient for a Neo-Hookian sample should be linear. When the curve jumps or becomes non-linear, this indicates a sudden displacement, which in turn indicates crack propagation. This abrupt displacement is due to the 'slackening' of the two free sides as they are enlarged by the crack.

To begin with, we need to import all our data from the DIC and assign them to variables:

```
rehash
dir data_DIC_1.mat

%First, we import the data from the DIC
    . If needed, the name of the file
%can be changed
%We register all the data into a
    structure
load data_DIC_1.mat data_dic_save

%Now we need to detect the number of
    rows, columns and pictures involved
    .
%We load the first picture displacement
     grid to get the rows and columns,
    and load
%the whole diplacements to detext the
    number of picture
Mat_Size = size(data_dic_save.
    displacements(1).plot_u_dic);
Pic_Size = size(data_dic_save.
    displacements);

rows = Mat_Size(1);
col = Mat_Size(2);
pictures = Pic_Size(2);

%Now, we save displacements and strains
     into variables to be used ! The
%third dimension of the matrix
    correspond to the picture. The
    first is row,
%and second is column !
for i = 1:pictures %displacements
    u_dic(:,:,i) = data_dic_save.
        displacements(i).
        plot_u_ref_formatted; %
        displacement in the x direction
    v_dic(:,:,i) = data_dic_save.
        displacements(i).
        plot_v_ref_formatted; %
        displacement in the y direction

    corrcoeff(:,:,i) = data_dic_save.
        displacements(i).
        plot_corrcoef_dic;
end

for i = 1:pictures %strains
    exx_dic(:,:,i) = data_dic_save.
        strains(i).
        plot_exx_ref_formatted; %x
        direction
    exy_dic(:,:,i) = data_dic_save.
        strains(i).
        plot_exy_ref_formatted; %shear
        strain
    eyy_dic(:,:,i) = data_dic_save.
        strains(i).
        plot_eyy_ref_formatted; %y
        direction
end
```

All the displacements and strains are now recorded in variables. For this example, the crack is propagating in the y direction, so the displacement gradient studied will be the perpendicular one, i.e. the xx gradient. The displacement gradient is calculated for each pixel in the image, for each image :

```
for i = 1:pictures
    [grad_xx(:,:,i),grad_xy(:,:,i)] =
       gradient(u_dic(:,:,i));
    [grad_yx(:,:,i),grad_yy(:,:,i)] =
       gradient(v_dic(:,:,i));
end
```

Now that the gradients have been calculated, they first need to be cleaned. Because of the nature of the gradient and DIC calculations, abnormal values appear at the top edge of the image, where no displacement should occur. To counter this problem, the gradient is set to 0 for the entire top edge:

```
for p=1:pictures
    for c=1:col
        grad_xx(1,c,p) = 0;
        grad_xy(1,c,p) = 0;
        grad_yx(1,c,p) = 0;
        grad_yy(1,c,p) = 0;
    end
end
```

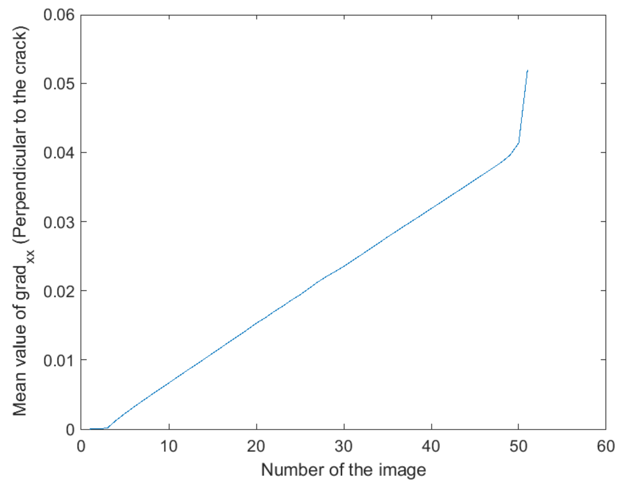
Now that the gradient is ready to be used, we need to determine the average displacement gradient for each image. Zero values should be excluded, as they are not representative of the displacement undergone by the sample. The non-zero values of the displacement gradient for each pixel in each image are therefore stored in a new variable, and this variable is then used to calculate the average of each gradient, for each image!

```
for p=1:pictures
    %We now want to find the mean value
        of each gradient, ignoring the
        ones
    %that are at 0. This mean value
       will then be used to eliminate
    %inexpected values of grad
    i=1;
    k=1;
    l=1;
    m=1;
    for c=1:col
        for r= 2:rows
            if(grad_xx(r,c,p) > 0)
                grad_xx_0(p,i) =
                   grad_xx(r,c,p);
                i = i + 1;
            end
            if(grad_xy(r,c,p) > 0)
                grad_xy_0(p,k) =
                   grad_xy(r,c,p);
                k = k + 1;
            end
            if(grad_yx(r,c,p) > 0)
                grad_yx_0(p,l) =
                   grad_yx(r,c,p);
                l = l + 1;
            end
            if(grad_yy(r,c,p) > 0)
                grad_yy_0(p,m) =
                   grad_yy(r,c,p);
                m = m + 1;
            end
        end

    end


    %We calculate the mean value for
       each direction
    Mavg_xx(p) = mean(grad_xx_0(p,:));
    Mavg_xy(p) = mean(grad_xy_0(p,:));
    Mavg_yx(p) = mean(grad_yx_0(p,:));
    Mavg_yy(p) = mean(grad_yy_0(p,:));
end
```

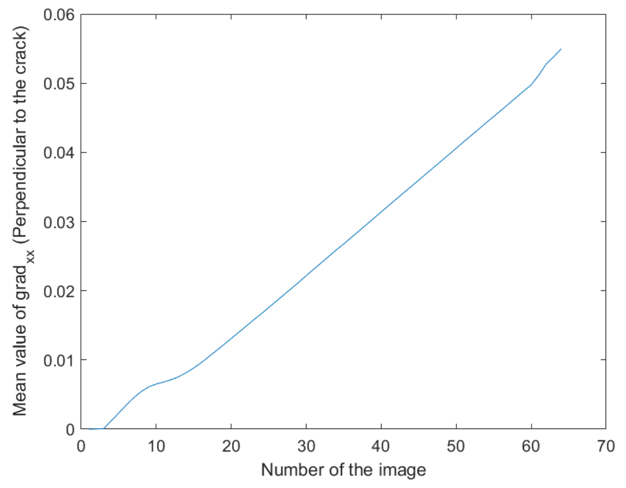
All the averages of each image for each displacement gradient are now stored in Mavg$_n m$, with n and m representing values that can be x or y. These averages can also be used to clean up the deformation gradient mappings, by applying tresholds based on the average. This could be useful, for example, to exclude unusual values linked to DIC errors. However, for the present paper, it is the plots of these averages against images that will be analyzed:

```
%We plot the mean values of each image
   to help us locate the crack :
%indeed, the crack is propagating on
   the part where the plot do a "jump"
plot(1:pictures,Mavg_xx);
xlabel('Number of the image')
ylabel('Mean value of grad_{xx} (
   Perpendicular to the crack)')
saveas(gcf,'Mean_curve_gradxx.png')
```

Two examples, for two different tests, are shown in Figure 1. Two things can be said. Firstly, in some cases, the curve is non-linear at the very start of the experiment. This non-linearity is mainly due to the sample not being taut at the start of the experiment. Once stretched, it becomes linear again. Secondly, in both cases, the curve becomes non-linear after a certain stage, or makes a slight jump. This part clearly indicates crack propagation. Figure 2 and 3 show the images before and after the non-linearity, and propagation is indeed visible.

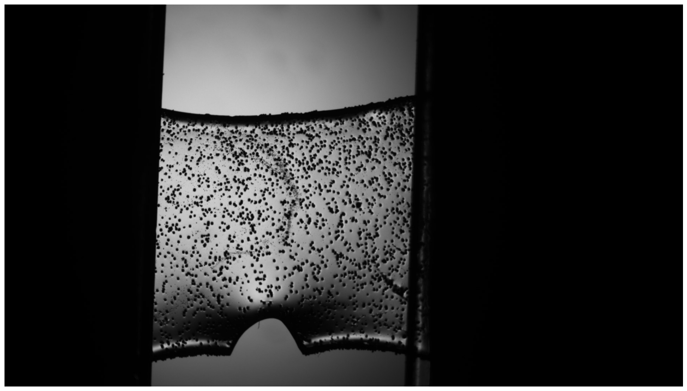


(a) Example A

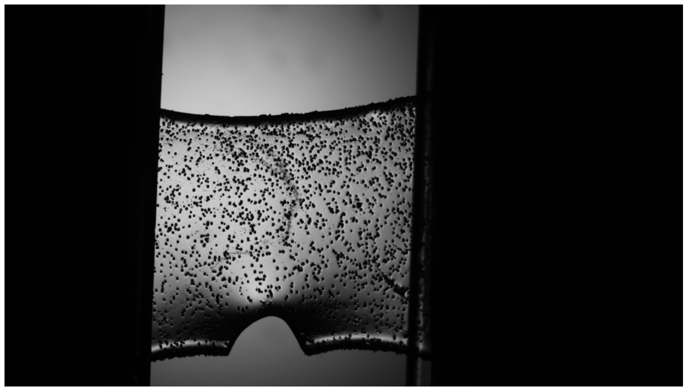


(b) Example B

FIG. 1: Evolution of the mean deformation gradient xx as a function of images. Non-linearity on the right-hand side of the curve indicates crack propagation. The exact propagation image can be obtained by analyzing the curve directly in matlab.

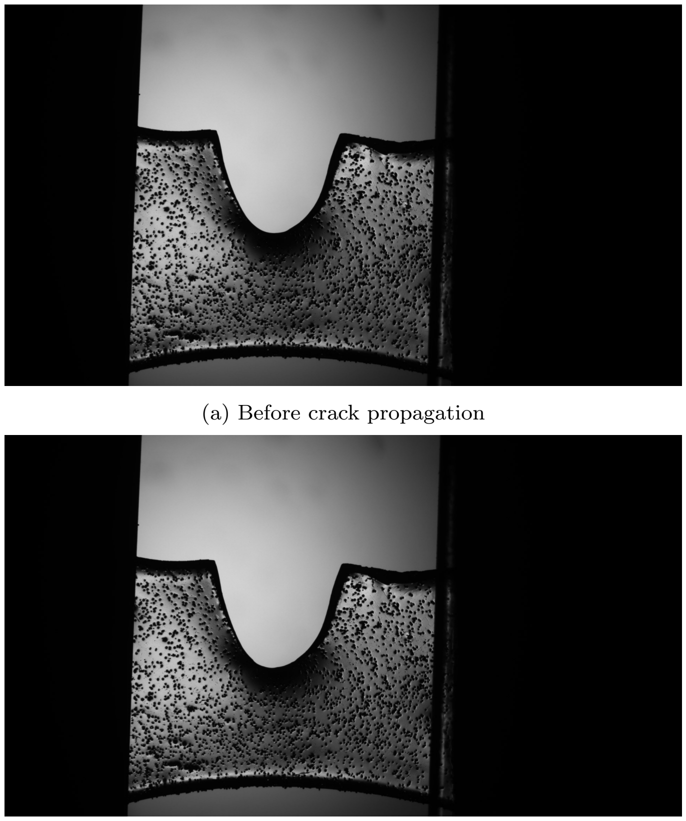


(a) Before crack propagation



(b) After crack propagation

FIG. 2: Image before and after crack propagation for example A. The propagation is barely perceptible, but takes place. This is the first propagation, before an unstable propagation takes place and tears the sample into two parts.



(a) Before crack propagation

(b) After crack propagation

FIG. 3: Image before and after crack propagation for example B. The propagation is barely perceptible, but takes place.

For the two examples chosen, this method shows a strong ability to reliably detect crack propagation. After individual verification of the images, no propagation was detected prior to that discovered by curve analysis. Logically enough, a non-linearity of the displacement gradient curve perpendicular to the crack as a function of the images indicates crack propagation, due to the partial release of the sample. After analysis on several samples and DIC, this method did not reveal any particular error, and could therefore be used quite reliably for crack propagation determination. It is important to note, however, that this method is only effective for neo-Hookian solids, as it requires elastic deformation throughout the test. Further tests are required to confirm the robustness of this program, and may be carried out in the future.

ACKNOWLEDGMENTS

We wish to acknowledge Mr. John Martin Kolinski and the Teaching Assistants for their help during the experiment, especially Chen Zhuo for his assistance in using the loading apparatus. We are grateful for this opportunity to know better about crack structures and the theory behind those, which we were unaware of.