In this document, the Matlab codes designed for the implementation of the methodology and the computation of the contributions of the shear transfer mechanisms are presented.

```
UC_PARAMS, example of the script that contains the parameters needed
to run the codes

%%%PHOTO PARAMETERS
Param.vPhotoSize=[8688 5792]; %photo size in pixels
Param.nScale=.3140;  %scale mm/pix
Param.nScale2=Param.nScale*grid_setup.step;   %Scale multiply by the
grid
Param.nImages=image_setup.N_images_total;  %total number of images
Param.nTheta(1:Param.nImages)=0.0060;  %rotation angle in rad

%PARAMETERS TO FILTER THE CRACKS (MEAN; STD & DISP)
Param.nThreshold=.3;       %mean threshold (less->more pixels)
Param.nDisp=.95;              %Dispersion of the pixels (more->more
pixels)
Param.nFactorStd=0.012;            %Standard deviation (less->more
pixels)
Param.nVprt=30;             %number of vertical lines to divide the
Data
Param.nHprt=20;               %number of horizontal lines to divide
the Data

%GROUPING OF THE CRACKS
Param.nRadio=1;             %DO NOT CHANGE
Param.nFisMin=30;   %Min number of pixels per crack
Param.nPixFisVH=1000;    %Size from which the check verticality or
horizontality

%FILTERING OF THE CRACKS
Param.nR2=0.8;
Param.nPixMin=25;      % Min number of pixels of the smallest parts
of the crack
Param.nMinPixAlin=1000;  % Min number of pixels that must be aligned
to be eliminated
Param.nSemiAnchBanda=0;   %Semi width of the area to check

%Spacing
Param.Spacing.nH=80;
Param.Spacing.nV=8;

%Center of the Crack
Param.Cent.Legs=20;  %free distance in the same row to consider a
new leg in the crack
Param.Cent.Conex=10; %free distance to consider a connection between
pixels of different rows
```

```
%Contour
Param.Cont.R=2.5; % is the distance (R) from the center of the crack
to a grid  to a point that contains information of displacements

%Concrete parameters for the calculation of aggregate interlock
Param.Conc.nfc=70;
Param.Conc.nDa=16;
Param.Conc.TenSoft.nDelta_u=.1;
Param.Conc.TenSoft.nC1=3;
Param.Conc.TenSoft.nC2=6.93;

%Beam dimensions and rebar properties
Param.Beam.nWidth=300;               %Width of the beam
Param.Beam.RebarDiam=25;              %Diameter of the longitudinal
rebar
Param.Beam.RebarPos=42;               %Position of the rebars, from
the botom of the beam
Param.Beam.RebarNum=4;              %Number of rebars
Param.Beam.Heigth=1200;             %Height of the beam
Param.Beam.d=Param.Beam.Heigth-Param.Beam.RebarPos;      %effective
height

%Shear force for uncracked concrete
Param.Force=234000;  %[N]
%Dowel Action limit from Baumman and Rüsch
Param.DA.Lim=0.08;

%-------------------------------------------------------------------
-------------------------------------------------
%% Script for crack displacements in x and y direction (crack
width and Shear displacements)
%This script plots the exx strains of a chosen image to identify
the cracks. It will provide a plot of the distribution of the exx
strains along the longitudinal axis of the beams at a given level
(r). The center of the crack will be located at the peaks and the
points to the left and
%right can be determine manually. Select first the point in the
left and then the one in the right. The input consists of:

%DATAXXXX, DIC results
%force, mat file vector containing the applied shear force
%time, mat file vector containing time from loading procedure
%UC_Params, script with all parameters
%DISTORTION, refers to the indices of the polynomial function to
correct the radial distortion
%center, is the center of the photographs in pixels
%nImage, information of the image to use as display to identify
the cracks
%mData, information of the exx strains in matrix
%r, is the row in the mData matrix that represents the a level or
height in the beam. The units are in grid point.
```

```
clear all
load DATA12150
load force
load time
UC_Params
DISTORTION = [1.874946e-13, -7.833352e-10, -8.288187e-07,
1.003525e+00, -2.115492e+00];
center=[8688/2+.5, 5792/2+.5];
nImage=48; %image to display information
 r=36;     %%Row that contains the information, for instance the
rebar position or mid height of the beam

%% DATA information
nR=length(unique(gridy_DU)); %length of y's
nC=length(unique(gridx_DU)); %length of x's
vVect=small_strain.exx(:,nImage);   %change to display results
from chosen pic
mData=aux_V2M(vVect,nR,nC);     %information in a matrix
mData(isnan(mData))=mean_UC(mData(:))*100; %NaN data to mean*3

% r=round(nR-Param.Beam.RebarPos/Param.nScale2); %%rebar location
r=round(nR-0.5*Param.Beam.Heigth/Param.nScale2) %%mid-height

y=mData(r,:); %information

%% FOR THE IDENTIFICATION OF CRACKS
figure(1)
clf
imshow(mData)
title('Raw information')
axis on
grid on
grid minor
hold on
rebar=[0,r;nC,r];

plot(rebar(:,1),rebar(:,2),'Linewidth', 2);

hold off

%% PLOT OF exx strains
figure(2)
plot(y,'Linewidth', 1);
% Add a title
title('exx distribution')
hold on
plot(l(:,1),l(:,2),'Linewidth', 1);

xlabel('x');
ylabel('exx');
grid on
grid minor
```

```
ax=gca; ax.LineWidth = 1;

prompt = 'How many cracks? ';
LVDT.nLVDT = input(prompt)
LVDT.mCoor{LVDT.nLVDT}=zeros(2);       %Coordinates of the 2 ends
in mm
LVDT.mPos{LVDT.nLVDT}=zeros(2);       %Position of the 2 ends in
pix

[x,y] = ginput(LVDT.nLVDT*2);

    LVDT.mXY=[(x) (y)];          %we store the position in pixels

hold off

%% matrix of coordinates

LVDT.mXY(:,2)=r;
X1=LVDT.mXY(:,1)
Y1=LVDT.mXY(:,2)

for nLVDT=1:LVDT.nLVDT
    LVDT.mPos{nLVDT}=[X1((nLVDT-1)*2+1:(nLVDT)*2) Y1((nLVDT-
1)*2+1:(nLVDT)*2)];
end

%% We transform Coor in mm to position in pix

Coorx=grid_setup.x_matrix(1)+LVDT.mXY(:,1)*grid_setup.step;
Coory=grid_setup.y_matrix(1)+LVDT.mXY(:,2)*grid_setup.step;

%% TAKING DISTORTION

rad = (Coorx(:,1).^2+Coory(:,1).^2).^0.5;
r1 = polyval(DISTORTION,rad);
Coordx = [((Coorx(:,1)-center(1)).*r1./rad + center(1))]-.5;
Coordy = [((Coory(:,1)-center(2)).*r1./rad + center(2))]-.5;

%% We rescale the grid to the size of the number of 'Finite
Elements'

for nLVDT=1:LVDT.nLVDT
    LVDT.mCoor{nLVDT}=[Coordx((nLVDT-1)*2+1:(nLVDT)*2)
Coordy((nLVDT-1)*2+1:(nLVDT)*2)];
%LVDT.mPos{nLVDT}*grid_setup.step;
end

%% We look at the displacements on the brackets

for nLVDT=1:LVDT.nLVDT
    for nSide=1:2
```

```
LVDT.mW{nLVDT,nSide}=aux_03_Weights(LVDT.mPos{nLVDT}(nSide,:));
%Weights
    end
end

%% Load displacements of the test
nR=grid_setup.Ny-1;
nC=grid_setup.Nx-1;

for nImage =1:Param.nImages
    % % X displ
    vVect=dispx_raw(:,nImage);

uData.DispX.data{nImage}=aux_V2M(vVect,nR+1,nC+1)*Param.nScale;
    uData.DispX.name={'raw'};% 'rdcd_interp' 'rdcd_raw'};

    % % Y disp
    vVect=dispy_raw(:,nImage);

uData.DispY.data{nImage}=aux_V2M(vVect,nR+1,nC+1)*Param.nScale;
    uData.DispY.name={'raw'};% 'rdcd_interp' 'rdcd_raw'};


[LVDT.Displ.X{nImage},LVDT.NaNs{nImage}]=A_09_DispLVDT(LVDT,uData.
DispX.data{nImage});

[LVDT.Displ.Y{nImage},~]=A_09_DispLVDT(LVDT,uData.DispY.data{nImag
e});

end

LVDT.Def=A_10_DefLVDT(LVDT,Param);

% % We calculate the displacements of the top and the bottom of
the
% especimen
mDispl=zeros(4,size(uData.DispX.data{1},2));
nWidth=size(uData.DispX.data{1},2);
vRange=floor(nWidth/6):floor(nWidth*5/6);
for nImage =1:Param.nImages
    vPos=floor(LVDT.mPos{1}(:,2));

    mDispl(1:2,:)=uData.DispX.data{nImage}(vPos,:);
    mDispl(3:4,:)=uData.DispY.data{nImage}(vPos,:);

    LVDT.Def.LineX(nImage)=mean_UC(mDispl(2,vRange))-
mean_UC(mDispl(1,vRange));
    LVDT.Def.LineY(nImage)=mean_UC(mDispl(4,vRange))-
mean_UC(mDispl(3,vRange));

end
```

```
%-------------------------------------------------------------------
-------------------------------------------------
function D=A_04_LocalDeltaTau(SC,D,Param)
% % Function that takes the crack profile and the displacements at
the adjacent grid points and gives the projected displacements (n
and t)

% % Left and right-side difference
% D.Delta is the displacement normal to the crack profile
% D.Tau is the displacement parallel to the crack profile

Delta{length(SC.mFis),2,length(SC.vCent)}=[];
Tau{length(SC.mFis),2,length(SC.vCent)}=[];

for nCrack=1:length(SC.mFis)

    for nCent=1:size(SC.mFisCent3{nCrack},1)
        for nSide=0:1


            nAlfa=SC.vAlfa{nCrack}(nCent); %crack angle

Delta{nCrack,nSide+1,nCent}=D.DispX{nCrack,nSide+1,nCent}*sin(nAlf
a)-D.DispY{nCrack,nSide+1,nCent}*cos(nAlfa);

Tau{nCrack,nSide+1,nCent}=D.DispX{nCrack,nSide+1,nCent}*cos(nAlfa)
+D.DispY{nCrack,nSide+1,nCent}*sin(nAlfa);
        end

% *(-1)^1 at the end to make Delta and Tau positive sign

        Delta{nCrack,3,nCent}=((Delta{nCrack,1,nCent}-
Delta{nCrack,2,nCent}))*(-1)^1 ;
        Tau{nCrack,3,nCent}=(Tau{nCrack,1,nCent}-
Tau{nCrack,2,nCent});
    end
end

D.Delta=Delta;
D.Tau=Tau;
%-------------------------------------------------------------------
-------------------------------------------------

function TC=A_05_LocalSigmaTau(DC,Conc)

% % This function takes as an input the difference of the
displacements along the crack profile and computes the stresses using
Walraven equation.
%The input is as follows:
%---Conc structure that contains the following information:
%nfc, concrete strength
%nDa, size of aggregate
```

```
%TenSoft, structure with nDelta_u,nC1 and nC2
%----DC, refers to the structure with the information of the
displacements
%in X and Y direction, as well as the normal and tangential
displacements

da=Conc.nDa;      %Size of agregate

Delta=DC.Delta;
Tau=DC.Tau;

nSide=3;
TC.Sigma=zeros(size(Delta,1),size(Delta,3));
TC.Tau=zeros(size(Delta,1),size(Delta,3));
TC.Model=zeros(size(Delta,1),size(Delta,3));

for nCrack=1:size(Delta,1)
    nCont=0;
    for nCent=1:size(Delta,3)
        if ~isempty(Delta{nCrack,nSide,nCent})
            w0=Delta{nCrack,nSide,nCent};
            D0=Tau{nCrack,nSide,nCent};

            if w0<0.10
                sig=0;
                tau=0;
            elseif Conc.TenSoft.nDelta_u>w0 && nCont==0
                [sig,tau]=AI_TenSoft(w0,Conc); %tension softening
                TC.Model(nCrack,nCent)=1;
            else
                [sig,tau]=AI_walraven(w0,   D0,   da,   Conc.nfc);
%Walraven's aggregate interlocking formulation
                TC.Model(nCrack,nCent)=2;
                nCont=1;
            end

            TC.Sigma(nCrack,nCent)=sig;
            TC.Tau(nCrack,nCent)=tau;
        end
    end
end
%-------------------------------------------------------------------
-------------------------------------------------

function DA=A_07_DwAction(DC,SC,Param)

% % Function that gets the displacements in y direction and
returns the dowel action force using Baumman and Rüsch equation
%input parameters:
%DC, refers to the structure that contains the displacements along
the crack profile
%SC, information of the crack coordinates
```

```
%Param, contains the parameters

% Prelocating data
DA.DispY{length(SC.mFis)}=[];
DA.fY{length(SC.mFis)}=[];
DA.Cent{length(SC.mFis)}=[];
DA.Cruce{length(SC.mFis)}=[];

% Find the position of the rebars
nH=Param.Beam.RebarPos/Param.nScale2;       %height of the rebars
in nodes

% Maximum Dowel Action Baumman and Rüsch equation
nDA=1.64*Param.Conc.nfc^(1/3)*Param.Beam.RebarDiam*(Param.Beam.nWi
dth-Param.Beam.RebarNum*Param.Beam.RebarDiam);

for nCrack=1:length(SC.mFis)
    vPosV=max(SC.mFis{nCrack}(:,1))-SC.mFis{nCrack}(:,1)+.5;
%The position of the pixels from the bottom of the photo
    [~,Pos]=min(abs(vPosV-nH));  %We find the closest crack pixel

    if vPosV(Pos)>nH     %The bars are in a lower position
        nPos=-1;
    else
        nPos=1;
    end
    Cent=SC.mFis{nCrack}(Pos(1),1);
    DA.Cruce{nCrack}=Cent

    if isempty(SC.mFisCent{nCrack});
       Pos=0;
       DA.DispY{nCrack}=0;
       DA.fY{nCrack}=0;
    else

    Pos1=find(Cent==SC.mFisCent{nCrack}(:,1));
    Pos=max(Pos1);

    DA.DispY{nCrack}=zeros(length(Pos),1);
    DA.fY{nCrack}=zeros(length(Pos),1);
    DA.Cent{nCrack}=Pos;
    for nPart=1:length(Pos)

        DA.DispY{nCrack}(nPart)=DC.DispY{nCrack,1,Pos(nPart)}-
DC.DispY{nCrack,2,Pos(nPart)};

DA.fY{nCrack}(nPart)=min(abs(DA.DispY{nCrack}(nPart)/Param.DA.Lim)
,1)*nDA*-1^1;
    end
    end

end
```

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

function UC=A_26_UnConAction(SC,Param)
% % Function to get the contribution of the uncracked concrete
part using Mörsch shear stress formulation
%input parameters:
%SC, crack profile coordinates
%Param, contains the parameters

% Prelocating data
UC.Height{length(SC.mFis)}=[];
UC.FY{length(SC.mFis)}=[];
d=Param.Beam.d;
h=Param.Beam.Heigth;
%Get the crack height and force
for nCrack=1:length(SC.mFis)
    zone=min(SC.mFisCent{nCrack}(:,1))*Param.nScale2;
    z=(2/3)*d+(1/3)*SC.mFisCent{nCrack}(1,1)*Param.nScale2;
    force=(2/3)*(zone/z)*Param.Force

    UC.Heigth{nCrack}=h-zone;
    UC.FY{nCrack}=force*-1^1;
end
%-------------------------------------------------------------------
-------------------------------------------------

% This script was modified to allow the manual selection of the
points that conform a crack profile. It calculates the
contribution of aggregate interlock, dowel action and uncracked
concrete.
%The important variables are explained as follows:
%DATAXXXXX, is a matfile that contains the results from DIC. The
relevant
%variables are: dispx_raw & dispy_raw are the raw displacements in
x and y direction. Units [pixels]. The structure small_strain
contains the strains:small_strain.exx, small_strain.eyy and
small.strain.eeqv.
%UC_Params, is the script that contains all the important
parameters
% nImage, refers to the image that will be used to get the data
%vVect, is the chosen information in form of a vector
%mData, is the matrix that contains the chosen informtaion
%POINTS.nPOINT is the number of points chosen to describe the
crack profile
%Crack.nCrack is the number of cracks
%Param.nScale, is the scale mm/pix

%% Folder to save the plots, user must change the directory
doutput='C:\Users\gzara\Desktop\TEST\20.
I123A\Results\DICjoin\final\352';
%& To load the DIC information & the parameters
```

```
load('DATA12150')
UC_Params

nImage=351;  %Picture

Param.Cont.R=1.5; % to change the distance R

%% To choose the strains or displacements and pass it from vector
to matrix
nR=length(unique(gridy_DU)); %number of rows
nC=length(unique(gridx_DU)); %number of columns

%User can change the strains, equivalent strains are chosen in
this example
vVect=small_strain.eeqv(:,nImage);    %eeqv %exx %eyy
mData=aux_V2M(vVect,nR,nC);

mData(isnan(mData))=mean_UC(mData(:))*3; %to change the NAN values

%% FILTERING OF CRACKS
%Plot the raw mData
figure(1)
clf
imshow(mData)
title('Raw information')
axis on
grid on
hold on

% % Filtering the get the peaks
mIn=A_06_GridFilter(mData,nR,nC,Param);

%Group the pixels, find the direction of the crack
mCrack=zeros(nR,nC);
mCrack(mIn)=mData(mIn);
Crack=A_00_Crack(mCrack,Param);

%Plot the filtered crack profile
figure(6)
clf
imshow(Crack.mImg)
axis on
title('Filtered and grouped')
grid on

% Select the number of points to describe the crack profile
prompt = 'How many points? ';
POINTS.nPOINT = input(prompt)

%Number of cracks
promp='How many cracks? ';
Crack.nCrack= input(promp)
```

```
Crack.nFis=Crack.nCrack;

mFis_Cent{Crack.nCrack}=zeros((POINTS.nPOINT),2);
mFis_New{Crack.nCrack}=zeros((POINTS.nPOINT),2);
for i=1:Crack.nCrack;
[x,y] = ginput(POINTS.nPOINT);
    plot(x,y,'o','linewidth',2)
    mFis_Cent{i}=[(y) (x)];
    mFis_New{i}=mFis_Cent{i};
end

Crack.SimpCrack.mFisCent=mFis_Cent;

Crack.SimpCrack.mFis=mFis_New;

hold off

%Function to save the coordinates of the crack profile
Crack.SimpCrack=A_15_CrackLineReg(Crack,Param);

TEST_SC %to plot the crack profile

%% STRAINS

% % Small strain
vVect=small_strain.exx(:,nImage);
uData.small.data{1}=aux_V2M(vVect,nR,nC);
vVect=small_strain.exy(:,nImage);
uData.small.data{2}=aux_V2M(vVect,nR,nC);
vVect=small_strain.eyy(:,nImage);
uData.small.data{3}=aux_V2M(vVect,nR,nC);
vVect=small_strain.eeqv(:,nImage);
uData.small.data{4}=aux_V2M(vVect,nR,nC);
uData.small.name={'xx' 'xy' 'yy' 'eqv'};

% % Large Strain
vVect=large_strain.Exx(:,nImage);
uData.large.data{1}=aux_V2M(vVect,nR,nC);
vVect=large_strain.Exy(:,nImage);
uData.large.data{2}=aux_V2M(vVect,nR,nC);
vVect=large_strain.Eyy(:,nImage);
uData.large.data{3}=aux_V2M(vVect,nR,nC);
vVect=large_strain.Eeqv(:,nImage);
uData.large.data{4}=aux_V2M(vVect,nR,nC);
uData.large.name=uData.small.name;

%% DISPLACEMENTS

% % Displacement in x direction
vVect=dispx_raw(:,nImage);
uData.DispX.data{1}=aux_V2M(vVect,nR+1,nC+1)*Param.nScale;
%units[mm]
```

```
uData.DispX.name={'raw'};% 'rdcd_interp' 'rdcd_raw'};

% % Displacement in y direction
vVect=dispy_raw(:,nImage);
uData.DispY.data{1}=aux_V2M(vVect,nR+1,nC+1)*Param.nScale;
uData.DispY.name={'raw'};% 'rdcd_interp' 'rdcd_raw'};

Crack.SimpCrack.R=Param.Cont.R; %R distance
Crack.SimpCrack2.R=Param.Cont.R; %R distance

Crack.SimpCrack=A_02_Contour(Crack.SimpCrack,uData.DispX.data{1}+u
Data.DispY.data{1}); %Finds the pixels located at the R distance

Crack.DispCont.DispX=A_03_DispContour(Crack.SimpCrack,uData.DispX.
data{1}); %Finds the displacements at the points located at the R
distance

Crack.DispCont.DispY=A_03_DispContour(Crack.SimpCrack,uData.DispY.
data{1}); %Finds the displacements at the points located at the R
distance

Crack.DispCont=A_04_LocalDeltaTau(Crack.SimpCrack,Crack.DispCont,P
aram); %projected displacements

TEST_DISP %PLOT the displacements along the crack profile

Crack.TensionCont=A_05_LocalSigmaTau(Crack.DispCont,Param.Conc);

TEST_STRESS %PLOT the aggregate interlock stresses along the crack
profile

Crack.ForcesCont=A_20_FxFy(Crack.TensionCont,Crack.SimpCrack,Param
);

TEST_FC %PLOT Vai

%Function to evaluate the dowel action with Baumman and Rüsch eq.
Crack.ForcesDA=A_07_DwAction(Crack.DispCont,Crack.SimpCrack,Param)
;
%Funtion for the contribution of uncracked concrete Mörsch eq.
Crack.ForceUC=A_26_UnConAction(Crack.SimpCrack,Param);


TEST_ALL %PLOT all the forces

TEST_plot %PLOT contour and crack profile

%% Influence analysis of the distance R in aggregate interlock
forces

vR=Param.Cont.R/4:Param.Cont.R/4:Param.Cont.R*1;
uFCrack=A_08_InfluenceofR(vR,Crack,uData,Param);
```