Augmented Lagrangian DIC (AL-DIC) Code Manual (v3.1)

Jin Yang[1,2], Kaushik Bhattacharya[1]

[1]Division of Engineering and Applied Science, California Institute of Technology
[2]Department of Mechanical Engineering, University of Wisconsin-Madison

Email: jyang526@wisc.edu; aldicdvc@gmail.com
Last modified date: 2020-04-28

Contents

Preface

In this Manual, all the lines needed to input to the MATLAB command window are marked with black background color; all the lines printed on the MATLAB command window are written in red color. Important words and details are highlighted in yellow background color.

Most important workspace variables and DIC results are summarized in Table 1-Table 3.

If you are dealing with multi-frames in an image sequence, feel free to modify some codes to save computation time. For example, please modify lines where require manual inputs or comment some lines which always pop out figures.

For full details, and to use this code, please cite our paper:

Yang, J. and Bhattacharya, K. *Exp.Mech.* (2019) 59: 187. https://doi.org/10.1007/s11340-018-00457-0.

Any question or good suggestions are welcome, please feel free to contact me at: jyang526@wisc.edu or aldicdvc@gmail.com. Thank you for your usage and I hope you will enjoy your DIC post-processing!

Section 0. Introduction

AL-DIC is a fast, parallel-computing hybrid DIC algorithm, which combines advantages of local subset DIC method (fast computation speed, and parallel computing) and finite-element-based global DIC method (guarantee global kinematic compatibility and decrease noise).

For a review of both local DIC and global DIC methods, and full details of the new proposed ALDIC method, please see our paper:
Yang, J. and Bhattacharya, K. *Exp.Mech.* (2019) 59: 187. https://doi.org/10.1007/s11340-018-00457-0. Full text can be requested at:
www.researchgate.net/publication/329456141_Augmented_Lagrangian_Digital_Image_Correlation

Here I highlight some advantages of AL-DIC algorithm:

- It's a fast algorithm using distributed parallel computing for a global nonconvex optimization.
- Global kinematic compatibility is added as a global constraint in the form of augmented Lagrangian, and solved using Alternating Direction Method of Multipliers scheme.
- Both displacement fields and affine deformation gradients are correlated at the same time.
- No need of much manual experience about choosing displacement smoothing filters.
- It works well with compressed DIC images and adaptive mesh. See our paper: Yang, J. & Bhattacharya, K. *Exp Mech* (2019). https://doi.org/10.1007/s11340-018-00459-y;
- Both cumulative and incremental DIC modes are implemented to deal with image sequences, where the latter mode is quite useful for measuring very large deformations.

Section 1. MATLAB mex setup

Execute this section and we will try to build "mex" functions from C/C++ source codes for image grayscale value interpolation, where both bi-cubic (by default) and bi-cubic splines interpolations are implemented in this code. For example, I use bi-cubic interpolations where the associated mex set up file is called "ba_interp2.cpp" [6] .

1.1. Test MATLAB mex setup

First, we test whether there is already a C/C++ compiler installed on your computer by inputting "mex -setup" and press Enter key on the MATLAB command window. If an available C/C++ compiler is already installed, please skip Section 1.2 and jump to Section 1.3.

1.2. Install mex C/C++ compilers

The step of installing mex C/C++ compilers is a common step for users to run C/C++ codes with MATLAB [7] [8] . For Windows users, you can follow these steps:

- Download: TDM-gcc compiler from: http://tdm-gcc.tdragon.net/
- Install TDM-gcc compiler on your computer. For example, I install it at: 'C:\TDM-GCC-64'. In practice, we find that this TDM-gcc compiler only works if installed on the first level main disks, such as "C:\", "D:\", "E:\", etc.
- Restart MATLAB and Input "setenv('MW_MINGW64_LOC','YourTDMGCCPath'); mex -setup;" on the MATLAB command window to check whether "mex" is set up successfully or not. Don't forget to replace the above YourTDMGCCPath using your own installation location of TDM-gcc package in the last step.

E.g. Input: setenv('MW_MINGW64_LOC','C:\TDM-GCC-64'); mex -setup;
Command window screen should output:

```
MEX configured to use 'MinGW64 Compiler (C)' for C language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
        variables with more than 2^32-1 elements. You will be required
        to update your code to utilize the new API.
        You can find more information about this at:
        https://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-
use-64-bit-api.html.

To choose a different language, select one from the following:
 mex -setup C++
 mex -setup FORTRAN
```

For Windows users, please uncomment line "setenv('MW_MINGW64_LOC','C:\TDM-GCC-64')", correct 'C:\TDM-GCC-64' to 'YourTDMGCCPath', and then execute Section 1 in main_ALDIC.m. Command window screen will print the following words:

```
------------ Section 1 Start ------------
Building with 'MinGW64 Compiler (C++)'.
Warning: You are using an unsupported version of MinGW Compiler. To
install the supported version of MinGW compiler, see: Install
MinGW-w64 Compiler.
 For a list of currently supported compilers visit
 https://www.mathworks.com/support/compilers.
MEX completed successfully.
------------ Section 1 Done ------------
Or
------------ Section 1 Start ------------
Building with 'MinGW64 Compiler (C++)'.
MEX completed successfully.
------------ Section 1 Done ------------
```

Comment: If you see some error messages as below, it's because the mex file is already executed. Please comment line (mex -O ba_interp2.cpp;), and re-execute Section 1 one more time.

```
------------ Section 1 Start ------------
Building with 'MinGW64 Compiler (C++)'.
Error using mex
C:/TDM-GCC-64/bin/../lib/gcc/x86_64-w64-mingw32/5.1.0/../../../../x86_64-w64-
mingw32/bin/ld.exe:
cannot open output file ba_interp2.mexw64: Permission denied
collect2.exe: error: ld returned 1 exit status

Error in main_ALDIC (line 15)
mex -O ba_interp2.cpp;
```

Section 2. Load DIC images and set up DIC parameters

This section is to load DIC reference and deformed images and set up DIC parameters. (You need to put your images on the MATLAB path beforehand.) All the DIC parameters will be stored in the workspace in "DICpara", and all the DIC ROI mesh properties are stored in "DICmesh" after executing Section 3. Here we make a brief summary of both these two data structures in Table 1 and Table 2.

Table 1. Summary of DIC parameters in "DICpara" structure

MATLAB workspace DICpara variable name	DIC parameter	Description
LoadImgMethod	Method to load images	Assign DIC image folder path to load all the included images or load images manually.
gridxyROIRange	To define DIC region of interest (ROI)	ROI is defined by clicking the top-left and right-bottom corner points.
winsize	DIC subset size	DIC local subset size in ALDIC Subproblem 1.
winstepsize	DIC subset step	Distance between neighboring local subsets, also the finite element size in ALDIC Subproblem 2.
Subpb2FDOrFEM	Method to solve ALDIC Subproblem 2	Both finite difference ('0-FD') and finite element ('1-FEM') methods are implemented in this code.
ClusterNo	To set up parallel pool	ALDIC Subproblem 1 can be sped up by applying parallel computing.
NewFFTSearch	Method to update initial guess when dealing with image sequence	In solving an image sequence, we can directly apply the result of last frame as the initial guess for the next frame.
ImgSeqIncUnit	To choose cumulative or incremental DIC mode	Cumulative DIC is always to compare with the first frame as the reference; while incremental DIC could update the reference frame after a certain number of frames.
ImgSeqIncROIUpdate	In incremental mode, ROI can be updated at the same time of updating reference image.	It's recommended to manually update ROI at the same time of updating reference image for measuring large deformations.

Table 2. Summary of details of DIC mesh in "DICmesh" structure

MATLAB workspace DICmesh variable name	DIC mesh parameter	Description
coordinatesFEM, elementsFEM	Coordinates of nodal points in the finite element mesh and their connectivity	Linear Q4 elements are used in the current code. However, it can be extended to other type of finite elements with arbitrary shape function.
dirichlet, neumann	FE-mesh nodal points at the boundary	Indices of nodal points at ROI borders are assigned with Dirichlet or Neumann boundary conditions.
x0, y0, M, N	Regular rectangular FE-mesh nodal grids	Regular rectangular FE-mesh nodal points can be represented in grids [x0,y0] with size as [M,N].
coordinatesFEMWorld, y0World	FE-mesh in the world coordinates	Coordinates in y-direction are transferred to world coordinates from intrinsic coordinates.

2.1. Load DIC images

```
------------ Section 2 Start ------------
Choose method to load images:
     0: Select images folder;
     1: Use prefix of image names;
     2: Manually select images.
Input here:
```

2.1.1. Load DIC images within certain folder

If we select method "0: Select images folder" to load DIC images, you will be asked to select the path of DIC image folder, and all the DIC images in this folder will be loaded. In the cumulative DIC mode, the first frame is assumed to be the fixed reference image, and all the other frames in the image sequence are deformed images whose deformation are calculated based on the Lagrangian description. In the incremental DIC mode, the reference image can be updated once after every certain number of frames. After loading DIC images, please jump to Section 2.4.

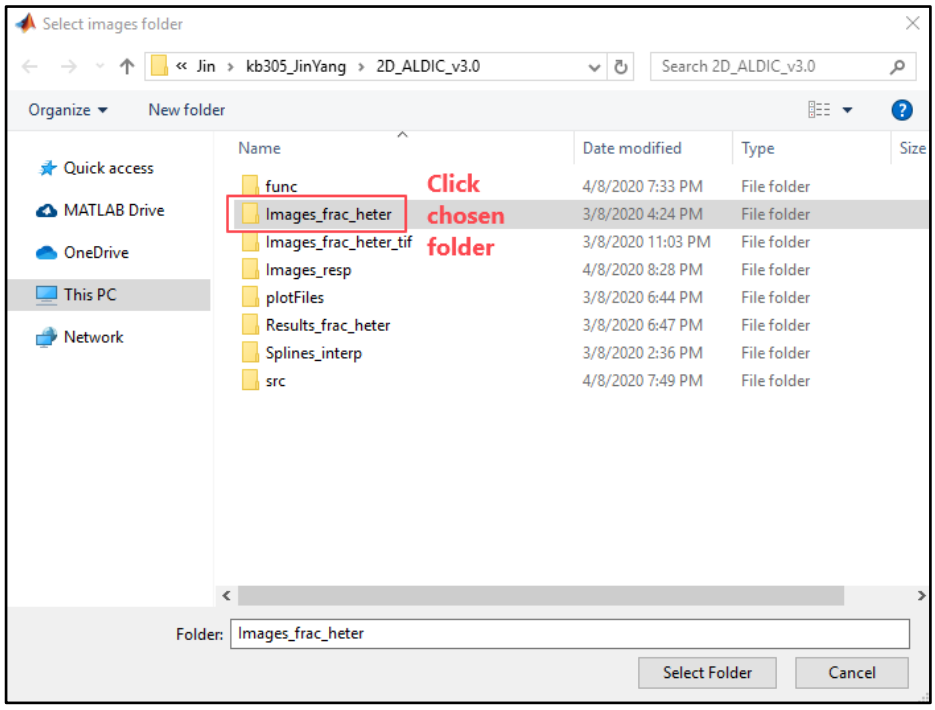


Figure 1. Loading DIC images by selecting the folder which includes all DIC images. For example, after clicking chosen folder "Images_frac_heter", all the images within this folder will be loaded.

2.1.2. Load DIC images with prefix of image name

If we select method "1" to load DIC images, you will be asked to input prefix text words of DIC image sequence frames (this image folder should be already added to MATLAB path). For example, if all the DIC images are named in the format as: "img_0001.tif", "img_0002.tif", …, "img_0100.tif", you should input "img_0*.tif" on MATLAB command window to load all the DIC images started with prefix "img_0" and in the "tif" format. After loading DIC images, please jump to Section 2.4.

```
What is prefix of DIC images? E.g. img_0*.tif.
Input here:
```

Comment: If you choose this method to load images, all the images should be saved in the "current Folder", or you should enter the image folder as the "current Folder". In practice, Section 2.1 method is preferred than Section 2.2

2.1.3. Load DIC images manually

If we select method "2" to load DIC images, you need to load DIC images frame by frame manually.

```
--- Please load first image ---
--- Please load next image ---
```

And then you can choose your images in the folder.

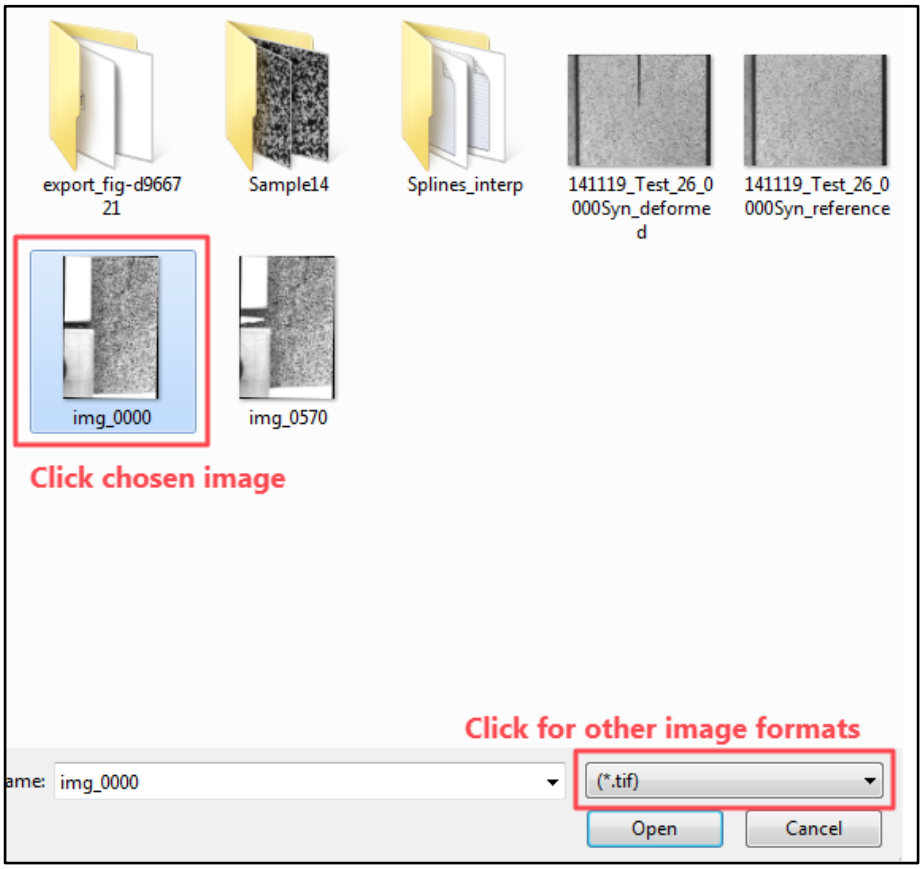


Figure 2. Load DIC images. Click reference image and deformed image.

Comment: I set first image as the reference image and following images as deformed images. and I always manipulate the deformed image transform back to the reference image to calculate the deformation field: which is based on the Lagrangian description.

If you want to describe the deformation in the Eulerian description, you should choose the reference image as the second image, and choose the deformed image as the first image, and manipulate the reference image transform to the deformed image.

```
Do you want to load more deformed images? (0-yes; 1-no)
```

Input "0" if you want to continue uploading images and input "1" if you want to stop uploading images. Current version: Please always input "1".

E.g., We choose first image as "img_0000.tif", and choose second image as "img_0570.tif".

2.2. Define region of interest (ROI)

After loading DIC images, please click both the top-left and bottom-right corner points on a popped-out image to define DIC region of interest (ROI). On the screen, it displays:

```
--- Define ROI corner points at the top-left and the bottom-right ---
```

First click a left-top point and then click a right-bottom point to define your ROI on the DIC image.

Figure 3. Click top-left and bottom-right corner points to define DIC region of interest (ROI).

After clicking both the top-left and bottom-right corner points, the command window screen will print their coordinates in the unit of pixels.

E.g., in the above DIC image, I define a ROI where two corner points are

```
Coordinates of top-left corner point are (322.786,74.730)
Coordinates of bottom-right corner point are (750.063,1128.439)
```

Comment: If the top-left or bottom-right corner point is clicked out of image border, it will be adjusted to the nearest point on the original DIC image borders automatically.

Comment: If you don't want to define ROI region from clicking on the images, instead you want to put in some fixed values using code, please go back to main_ALDIC.m and uncomment Line "% gridxROIRange = [gridxROIRange1, gridxROIRange2]; gridyROIRange = [gridyROIRange1, gridyROIRange2];" and modify the values of gridxROIRange and gridyROIRange as you want.

2.3. Set up DIC parameters

Then user will be asked to decide local subset size and the subset step. "Subset size" is the edge length of local subset window; while the "subset step" is the distance between two neighboring subsets, and choice of "subset step" can be independent of choice of subset size.

```
--- What is the subset size? ---
Input here:
--- What is the subset step? ---
Input here:
```

Subset size is the window size where we use to obtain initial guess through FFT cross correlation method, or through SSD minimization IC-GN iterations in the ALDIC local step.
Subset step size is also the finite element mesh size in the ALDIC global step.

Practically you can choose subset size the same or a little bit larger than the subset step size.
E.g., we both subset size as 20, and subset step is chosen as 10.

We also need to choose the solver method beforehand for ALDIC Subproblem 2 (global step).
For this version of code, we can only deal with uniform grid mesh, it almost doesn't matter whether you choose the "Finite difference method" or "Finite element method". Even you choose "Finite difference method", there are still finite element mesh generated (cf "DICmesh" in Matlab workspace) which could help you conduct FEA analysis if you want to combine DIC with other FEA codes/software. My personal experience is that finite difference method is a little bit more accurate than finite element method in the ALDIC Subproblem 2 (global step) because of the boundary effects.

```
--- Method to solve ALDIC global step Subproblem 2 ---
    1: Finite difference(Recommended)
    2: Finite element method
Input here:
```

We need to set up parallel pools or tell MATLAB we don't want to use parallel pools.

```
--- Set up Parallel pool ---
How many parallel pools to open? (Put in 1 if no parallel computing)
Input here:
```

If we don't want to use parallel pools, input "0" or "1". If we want to use parallel computing, put the number of your parallel pools. E.g. Input "4".

MATLAB parallel computing environment can be set in the "Home->Environment->Parallel->Parallel Preferences…". (Reference: https://www.mathworks.com/help/distcomp/parpool.html).

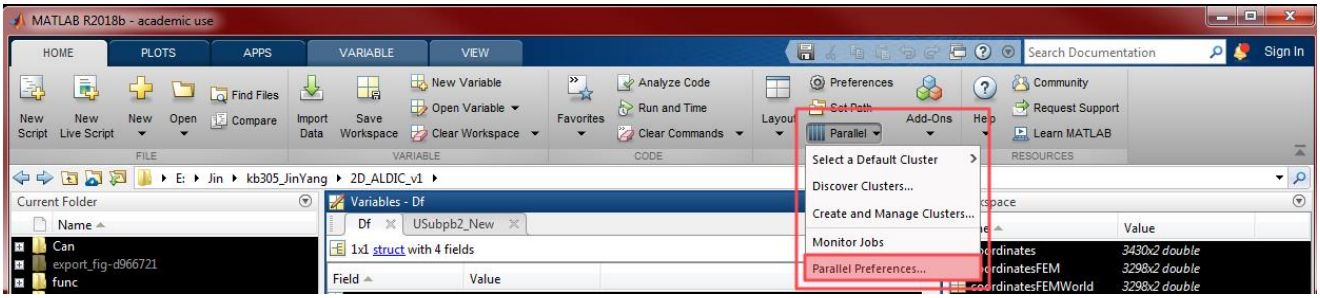


Figure 4. Set MATLAB Parallel Preferences.

2.4. Extensions: Additional parameters setup when dealing with image sequence

If we upload an image sequence with more than two frames, as for each new DIC frame, we can choose to use the displacement results in last frame as the initial guess for the new image frame, or we can just redo FFT initial guess for every new frame. This choice depends on how big relative displacements can be between two consecutive frames. Generally, if the relative displacement field between two consecutive frames is smaller than 5-7 pixels, we can use the deformation result of last frame as the initial guess displacement field for the new frame; otherwise it is suggested that we still need to redo the FFT initial guess process.

```
Since we are dealing with an image sequence with multiple frames, for each new frame,
do we use the result of last frame as an initial guess or redo FFT initial guess for
every new frame?
0: Use last frame;
1: Redo initial guess.
Input here:
```

E.g., In the example of "Images_frac_heter" image folder, we can use DIC results of last frame as the initial guess for the new frame's deformation field.

When postprocessing image sequence with more than two frames, user could decide to perform either *cumulative* mode DIC or *incremental* mode DIC. The cumulative mode is the default setup and all the following frames will be compared with the first frame. However, incremental mode is preferred when dealing with extremely large deformations but may lose some accuracy because of the reference image update. We recommend the user to try cumulative mode first, and if cumulative mode doesn't work very well, then try incremental mode. If you choose to use incremental mode, you will further be inquired to input how often you would like to update the reference image:

```
--- Choose cumulative or incremental mode ---
     0: Cumulative(By default);
     1: Incremental;
Input here:
```

```
Incremental mode: How many frames to update reference image once?
Input here:
```

E.g., I want to update my reference image once every ten frames, so I input: "10". The minimum number you can input is "1", which means to update reference image every frame.

Every time the reference image is updated, you can choose to update the region of interest (ROI) at the same time or not. To achieve this, user will be asked as follows:

```
Update ROI at the same time of updating reference image?
    0: Do not update ROI;
    1: Manually(Recommended);
    2: Automatically;
Input here:
```

E.g., Input "1" or "2" if you want to update ROI at the same time of updating reference image. Theoretically, this ROI update can be done automatically using the solved deformation of the last frame. However, we find it is most robust to update this ROI manually.

Section 3. DIC Initial guess based on Fourier transform (FFT) cross correlation

Here FFT cross correlation is computed as an initial guess.

```
--- Whole field initial guess (0) OR Several seeds (1) initial guess? ---
```

Whole field initial guess (0) is recommended.

```
--- What is your initial guess search zone size? ---
```

User needs to provide an initial guess search zone size. Here we play the trial-and-error approach until the value of search zone size is a little larger than the displacement amplitude. For example, we put in "70" in our example. Then the integer initial search starts and usually this step finishes very fast.

```
Are you satisfied with initial guess with current search region? (0-yes; 1-no)
```

Plug in "1" if you want to modify the above parameter "initial guess search zone size" and redo the initial integer guess search. Or you could put in "0" to stop it.

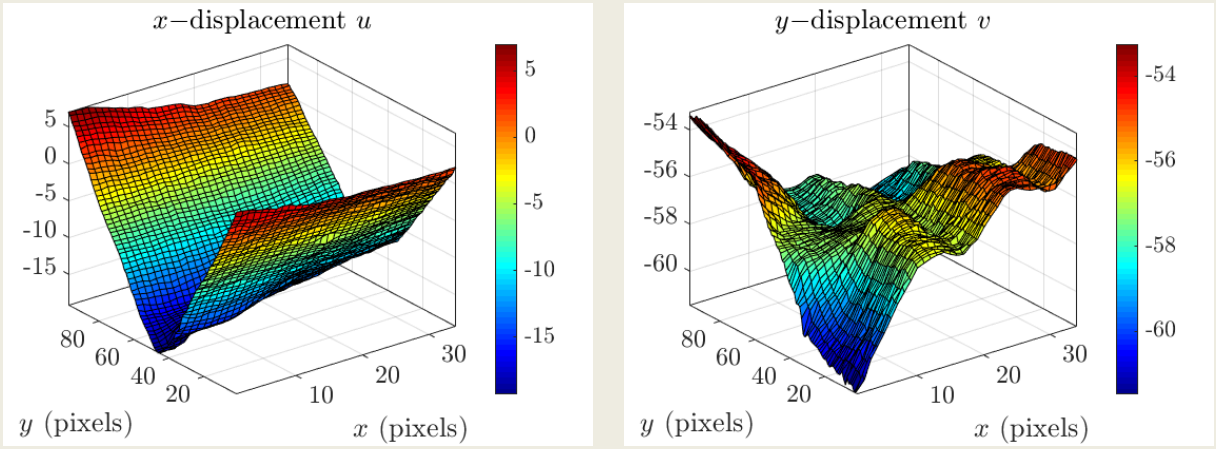


Figure 5. Solved displacement fields by FFT cross correlation.

After computing an initial guess, user can remove bad points by setting upper and lower bounds of displacements.

```
Do you clear bad points by setting upper/lower bounds once more? (0-yes; 1-no)
```

```
% ======= Find bad initial guess points manually by setting bounds =======

What is your upper bound for x-displacement?
What is your lower bound for x-displacement?
What is your upper bound for y-displacement?
What is your lower bound for y-displacement?

Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
```

If you put in "0", you will further be asked to set the upper bound and lower bound of the x and y displacements. If you put in "1", you will skip this process.

Besides setting upper and lower bounds to remove local bad points, we can continue to remove bad points by clicking them directly. (In the future, this part would be extended with qDIC codes.)

```
% ======= Find bad initial guess points manually =======

'Do you clear bad points by directly pointing x-disp bad points? (0-yes; 1-no)';
'Do you clear bad points by directly pointing y-disp bad points? (0-yes; 1-no)';
```

Directly clicking all the bad points and then press "Enter" key.

If you stop the bad points removal, mesh setting up and assigning initial value will be done automatically.

```
Finish setting up mesh and assigning initial value!
```

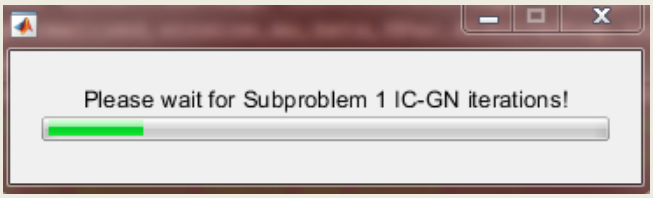
Image pixel grayscale gradients will also be computed very fast using finite difference operator and convolution operations.

```
--- Start to compute image gradients ---
--- Computing image gradients done ---
```

Section 4. ALDIC first local step: IC-GN iterations

In this section, we solve ALDIC ADMM iteration Subproblem 1 (local step) using IC-GN (inverse compositional-Gauss Newton) scheme, where distributed parallel computing has been implemented. Execute this section, a wait-bar will pop out automatically.

```
--- Set up Parallel pool ---
Starting parallel pool (parpool) using the 'local' profile ...
connected to 4 workers.
```



When this step finishes, a report will display on the command window and user can further choose to remove bad points by setting up upper & lower bounds or clicking bad points manually, where the bad points removal process is same with Section §3.

```
Local ICGN bad subsets %: xxx/xxx=xxxx%
Elapsed time is xxxxxx seconds.

--- Start to manually remove bad points ---
Do you clear bad points by setting upper/lower bounds once more? (0-yes; 1-no)
```

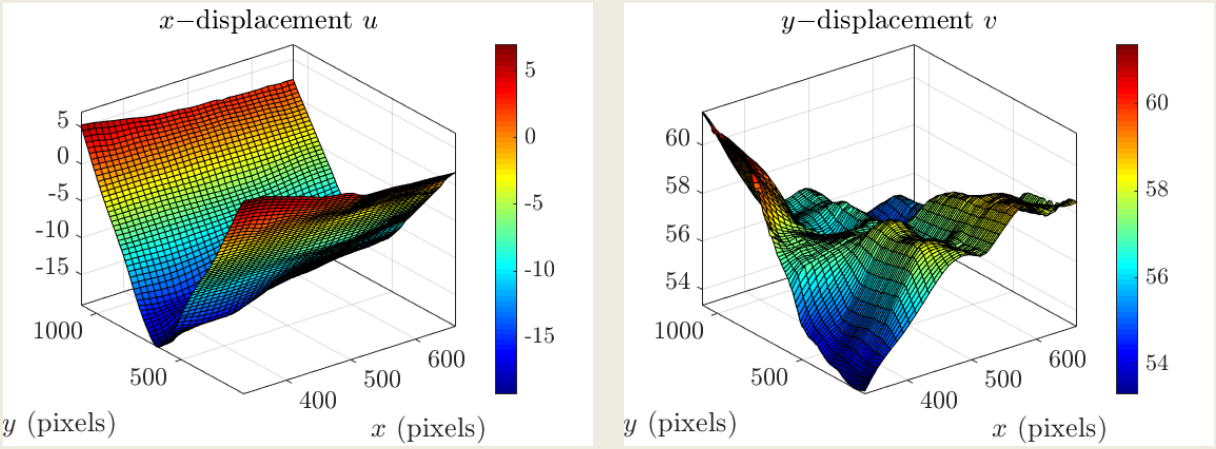


Figure 6. Solved displacement fields from ALDIC first local step.

```
% ======= Find bad Local IC-GN step points manually by pointing them =======

'Do you clear bad points by directly pointing x-disp bad points? (0-yes; 1-no)';
'Do you clear bad points by directly pointing y-disp bad points? (0-yes; 1-no)';
```

If you put in "0", you will further be asked to set the upper bound and lower bound of the x and y displacements. If you put in "1", you will skip this process.

```
% ======= Find bad guess points manually by setting bounds =======

What is your upper bound for x-displacement?
What is your lower bound for x-displacement?
What is your upper bound for y-displacement?
What is your lower bound for y-displacement?

Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
```

```
--- Remove bad points done ---
```

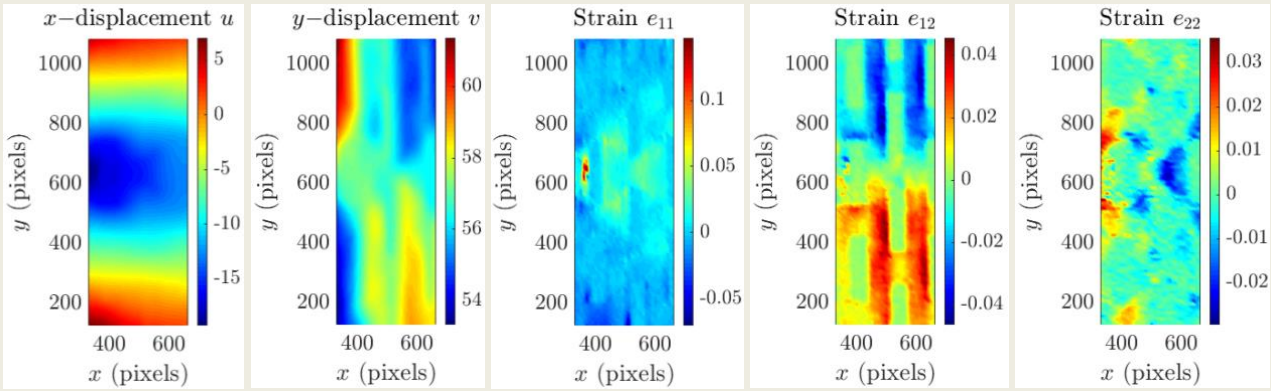


Figure 7. Solved displacement and deformation gradient minus identity from ALDIC ADMM first local step.

FAQ Comment: (Thanks question from PythonZhou)

```
'Insufficient data for surface estimation.','[u1temp] = gridfit(coordinatesFEM
(notnanindex,1), coordinatesFEM(notnanindex,2),U(2*notnanindex-1), Coordxnodes,
Coordynodes,'regularizer','springs'); u1temp = u1temp';'and 'main_ALDIC (line 106)
LocalICGN(U0,DICmesh.coordinatesFEM,Df,fNormalized,gNormalized,DICpara,'GaussNewton'
,tol);'.
```

This problem usually happens when there are *too few* local subsets are converged in Subproblem 1 (local step), which breaks down the "gridfit" function. Here are three possible steps to fix this problem.

1) In code Section 2, Check parameter of DIC local subset size (DICpara.winsize). In 2D-DIC, each subset is expected to have at least 3~5 features (e.g. speckle dots) in your DIC pattern. Usually with larger subset size, Subproblem 1 will have better convergence. But this subset size cannot be too large since it will also decrease the DIC overall spatial resolution. Theoretically, distance between neighboring subsets can be an arbitrary integer. Considering speed and accuracy, I recommend the winstepsize (DICpara.winstepsize) to be (0.25~1)* winsize.

2) In code Section 3, Check the initial guess (around main_ALIDC.m line 56~64), plot initial guess U0 to see whether this initial guess makes sense or not (U0 = Init(u,v,cc.max,DICmesh.x0,DICmesh.y0,0); PlotuvInit;). Here the newest version of code is to use the multiscale-method to search the initial guess in an adaptive way (line 60, IntegerSearchMg). If

this initial guess doesn't look great, please replace line (60) with line (58), where you can manually define the fixed size of FFT-search zone.

3) After checking (1-2), please re-run code Section 4. To see whether each local subset obtains convergence or not, you can uncomment (LocalICGN.m: line 35 (ClusterNo=1) or line 58 (ClusterNo>1)) to display this info message on the command window (However, this will slow down the code). So please still comment that display line after you fix this issue.

Section 5. ALDIC first global step

After solving ALDIC ADMM Subproblem 1 local step, we run Subproblem 2 global step in this section. Both finite difference and finite element methods are implemented. For uniform regular grid meshes, both finite difference and finite element method work very well. For arbitrary mesh, finite element method is much easier to implement and can be easily implemented with adaptive mesh technique. (Current ALDIC code is implemented on uniform Q4 mesh. H-adaptive mesh ALDIC code will come soon.

```
Assemble finite difference operator D
Elapsed time is xxxxxx seconds.
Finish assembling finite difference operator D
***** Start step1 Subproblem2 *****
Elapsed time is xxxxxx seconds.
```

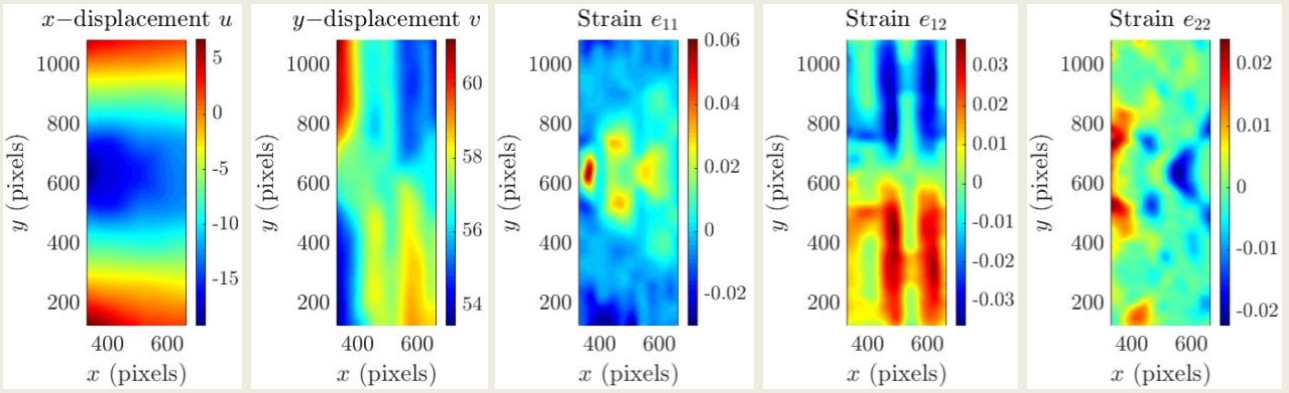


Figure 8. Solved displacement and deformation gradient minus identity from ALDIC ADMM first global step.

Section 6. ALDIC ADMM iterations

Then ALDIC will do Alternating direction method of multipliers (ADMM) iterations.
The tolerance threshold of ADMM iteration is set to be 1e-4 by default. But this threshold can be manually adjusted by modifying "tol2" in code Section 6.

```
------------ Section 6 Start -------
***** Start step2 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 6.078113 seconds.
***** Start step2 Subproblem2 *****
Elapsed time is 0.007632 seconds.
Update local step  = 0.018188
Update global step = 0.028851
**********************************

***** Start step3 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 6.036331 seconds.
***** Start step3 Subproblem2 *****
Elapsed time is 0.007165 seconds.
Update local step  = 0.0027562
Update global step = 0.025722
**********************************

***** Start step4 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 5.937378 seconds.
***** Start step4 Subproblem2 *****
Elapsed time is 0.007198 seconds.
Update local step  = 0.0019051
Update global step = 0.0018707
**********************************
```

```
***** Start step5 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 5.892075 seconds.
***** Start step5 Subproblem2 *****
Elapsed time is 0.007503 seconds.
Update local step  = 0.00015808
Update global step = 0.00027234
**********************************

***** Start step6 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 6.106291 seconds.
***** Start step6 Subproblem2 *****
Elapsed time is 0.007841 seconds.
Update local step  = 2.2404e-05
Update global step = 4.8639e-05
**********************************
```

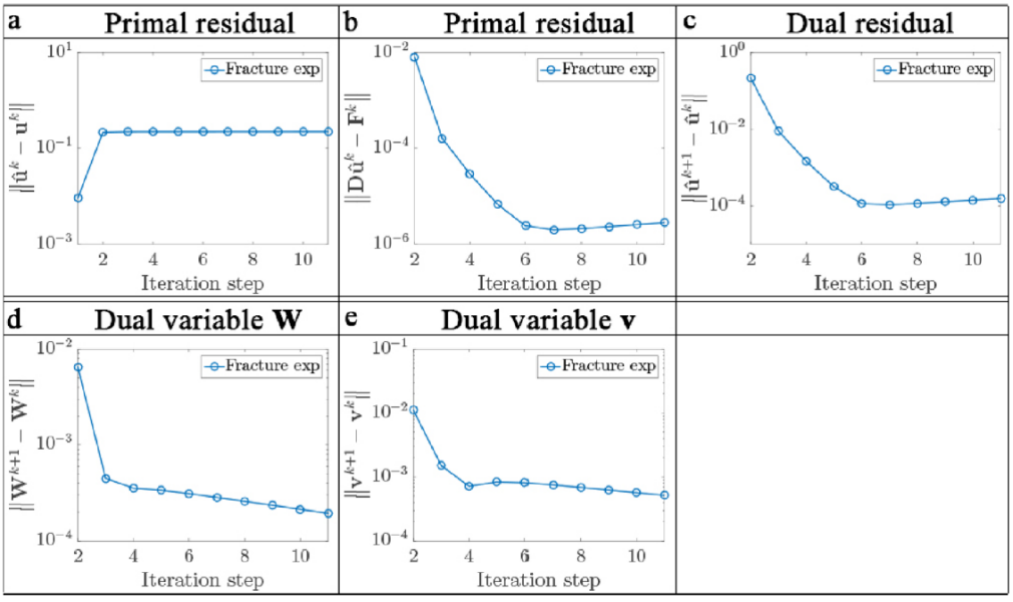


Figure 9. ALDIC ADMM algorithm usually converges after 3~5 iterations.

Section 7. Check convergence

This section is to check convergence of ALDIC ADMM iterations and delete some temporary variables in the workspace. This section can be skipped if you don't need it.

Section 8. Compute strain and output figures

8.1. Smooth displacement field if needed

Before computing strains, solved displacement fields can be further denoised if necessary. In most cases, ALDIC solved displacement fields are already denoised (cf Figure 10), so usually there is no need to further smooth solved displacement fields anymore.

```
Do you want to smooth displacement? (0-yes; 1-no)
```

If you put in "0", a Gaussian smooth filter with standard deviation 0.5 will be applied. If a stronger smoothing filter is needed, please edit the Gaussian filter parameters "DispFilterSize, and DispFilterStd".

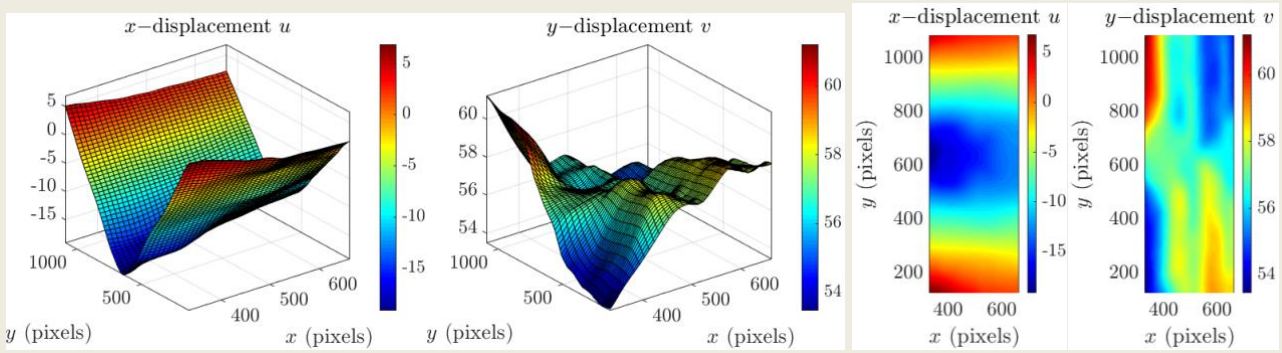


Figure 10. Solved displacement fields by ALDIC method is much denoised and usually there is no need to further smooth solved displacement fields anymore.

8.2. Compute strain field

In ALDIC algorithm, deformation gradient is a direct output besides displacement field. Strain field can also be computed from numerically differentiating solved displacement field. We implement total four methods of computing the strain field.

0) First method is a direct output from ALDIC solved "deformation gradient minus identity" **F**.
1) Second method is doing central finite difference of solved displacement fields.
2) Third method is based on the plane fitting method to differentiate solved displacement fields. In this method, you will be asked to input *half* plane width to define the size of the fitted plane.
3) Strain field can also be computed from finite element Gauss points.

```
What method to use to compute strain?
    0: Direct output from ALDIC;
    1: Finite difference(Recommended);
    2: Plane fitting;
    3: Finite element;
Input here:
```

If input "2-Plane fitting", please also input half window size to define the square plane size.

```
What is your half window size:
```

```
Infinitesimal stran or finite strain?
    0: Infinitesimal stran;
    1: Eluerian strain;
    2: Green-Lagrangian strain;
    3: Others: code by yourself;
Input here:
```

Three popular types of strains are implemented: infinitesimal strain, Eulerian strain based on deformed configuration, and finite Green-Lagrangian strain.

E.g., different methods to compute strain fields of our example are shown in Figure 11, and results of multi-frames in our heterogeneous fracture example are stored in "./results_frac_heter/".

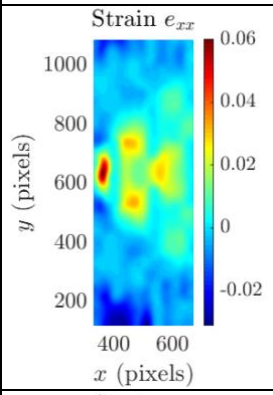
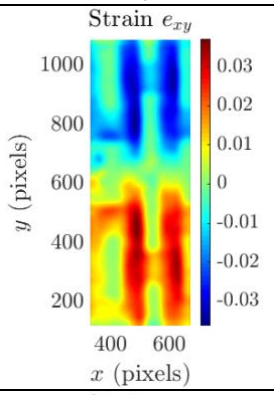
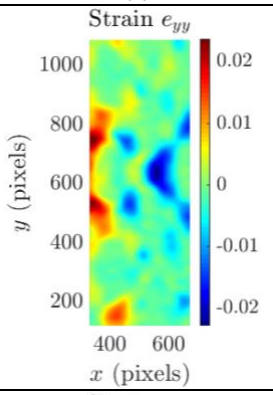
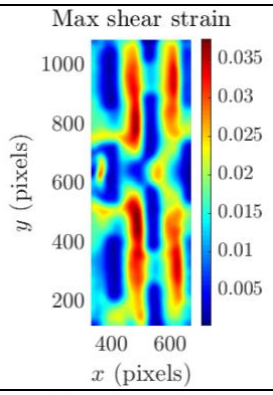
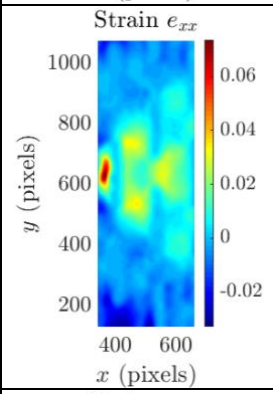
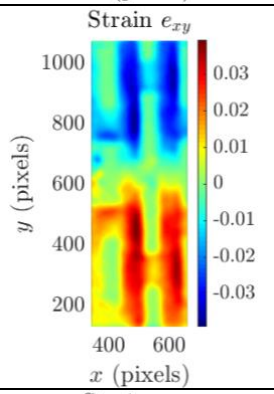
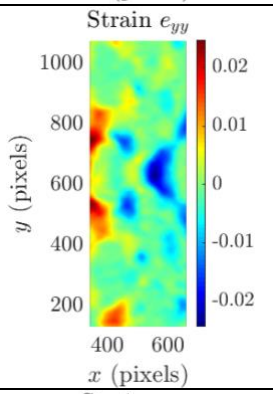
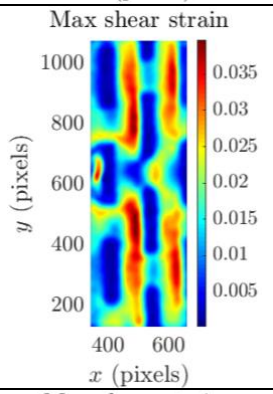
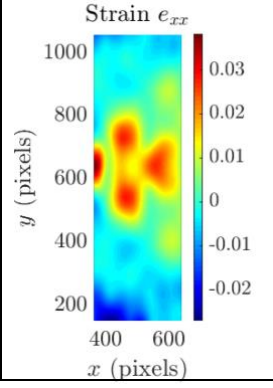
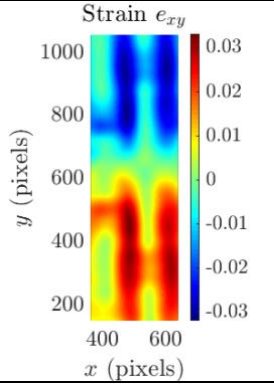
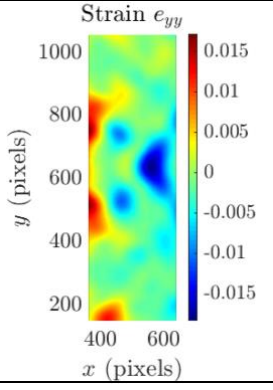
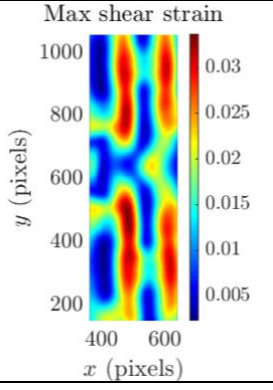
Method	e_{xx}	e_{xy}	e_{yy}	Max shear strain
Method 0 ALDIC directly solved strain				
Method 1 Central difference of displace-ment field				
Method 2 Plane fitting method (half window size: 3 pixels)				

Figure 11. Solved strain fields by ALDIC method.

8.3. Plot and save results

Users can visualize solved displacement and strain field and plot them overlaid with original DIC images [1]. Finally, don't forget to save your results for future use.

```
Save figures into different format:
      1: jpeg(Choose transparency 0~1)
      2: pdf(Choose transparency = 0)
      3: Others: Edit codes in ./plotFiles/SaveFigFiles.m
Input here:
```

```
Define transparency for overlaying original images:
Input a real number between 0(Only original images)
and 1(Non-transparent deformation results).
Input here(e.g. 0.5):
```

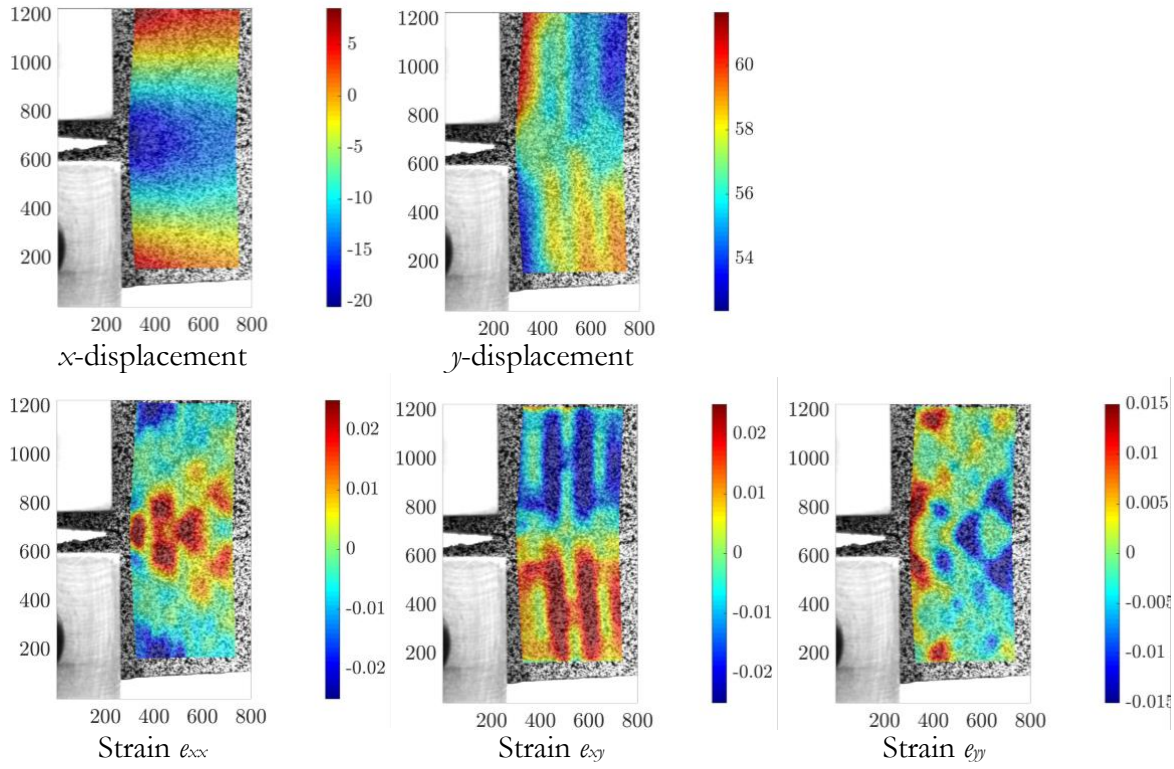


Figure 12. Plot and save computed deformation fields with original DIC images as background.

Table 3. Summary of DIC results

MATLAB workspace variable	Description
DICpara	ALDIC parameters, see Table 1
DICmesh	ALDIC finite element mesh, see Table 2
ResultDisp	Solved displacements and number of bad subsets during ADMM iterations
ResultDefGrad	Direct solved "deformation gradient minus identity" **F** from ALDIC
ResultStrain	Finally solved strain fields after Section 8
ResultFEMesh	Stored FE-mesh due to reference frame update in *incremental mode* DIC
ALSub1Time, ALSub2Time	Computation time in solving Subproblem 1 and Subproblem 2

[1] In this code, overlaying original DIC image can only be saved in the "jpg" format. I really appreciate if user/reader has better solution, feel free to contact me.

Acknowledgments

I am grateful to Dr. Louisa Avellar for sharing her unpublished images of fracture. I also want to thank Prof. Can C. Aydıner, Dr. Alex K. Landauer and Jialiang Tao for lots of helpful discussions. I also acknowledge the support of the US Air Force Office of Scientific Research through the MURI grant 'Managing the Mosaic of Microstructure' (FA9550-12-1-0458).

References

[1] Yang, J. and Bhattacharya, K. Augmented Lagrangian Digital Image Correlation. *Exp.Mech.* 59: 187, 2018. https://doi.org/10.1007/s11340-018-00457-0.

[2] Yang, J. and Bhattacharya, K. Combining Image Compression with Digital Image Correlation. *Exp Mech*, 2019. https://doi.org/10.1007/s11340-018-00459-y.

[3] Yang, J. and Bhattacharya, K. Fast Adaptive Global Digital Image Correlation. In preparation.

[4] Yang J. and Bhattacharya K. Fast Adaptive Global Digital Image Correlation. In: Advancement of Optical Methods & Digital Image Correlation in Experimental Mechanics, Volume 3. Conference Proceedings of the Society for Experimental Mechanics Series. Springer, 2019.

[5] Avellar L, Ravichandran G (2016) Deformation and fracture of 3d printed heterogeneous materials. Society for Experimental Mechanics Annual Conference.

[6] Website page: https://www.mathworks.com/matlabcentral/fileexchange/20342-image-interpolation-ba_interp2

[7] Website page: https://www.mathworks.com/matlabcentral/fileexchange/52848-matlab-support-for-mingw-w64-c-c-compiler

[8] Website page: https://www.mathworks.com/help/matlab/matlab_external/install-mingw-support-package.html