Augmented Lagrangian DIC (AL-DIC) Code Manual

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
Last modified date: 2019-02-27

Contents

Section 0. Introduction 2
Section 1. Mex set up for bi-cubic or bi-cubic splines interpolations 2
Section 2. Load DIC images 4
Choose interest region ZOI and DIC parameters 5
Section 3. DIC Initial guess based on Fourier transform (FFT) cross correlation 7
Section 4. ALDIC first local step: IC-GN iterations 9
Section 5. ALDIC first global step 11
Section 6. ALDIC ADMM iterations 12
Section 7. Check convergence 13
Section 8. Post processing: compute strain 13
Acknowledgment 15
References 15

Comment

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 written in `red` color. Important words and details are highlight in yellow background color.

To use this code, please cite our paper: Yang, J. and Bhattacharya, K. Augmented Lagrangian Digital Image Correlation. *Exp.Mech.* (2019) 59: 187. https://doi.org/10.1007/s11340-018-00457-0.

Section 0. Introduction

AL-DIC is a fast, parallel-computing DIC algorithm, which combines advantages of Local Subset DIC (fast, compute in parallel) and Global DIC (guarantee kinematic compatibility).

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.

Here I list some benefits of AL-DIC algorithm:

- Fast algorithm using distributed parallel computing;
- Add global kinematic compatibility as a global constraint in the form of augmented Lagrangian, and implemented using Alternating Direction Method of Multipliers scheme;
- Correlate both displacement fields and affine deformation gradients at the same time;
- Don't need much manual experience of choosing displacement smoothing filters.
- Work 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.

Section 1. Mex set up for bi-cubic or bi-cubic splines interpolations

Build mex functions from C/C++ source codes.
I use bi-cubic interpolations, mex set up file "ba_interp2.cpp".
Based on practical experience, mex-setup bi-cubic spline interpolation is also recommended.

1.1. Test mex -setup

Input "mex -setup" in the Command windows, to see whether a C/C++ complier has already installed or not. If already installed, please jump to Section §1.3.

1.2. Install mex C/C++ compliers

All the steps are extracted from website:
https://www.mathworks.com/matlabcentral/fileexchange/52848-matlab-support-for-mingw-w64-c-c-compiler and https://www.mathworks.com/help/matlab/matlab_external/install-mingw-support-package.html

If using Windows system, you can directly do the following steps.

- Download: TDM-gcc complier: http://tdm-gcc.tdragon.net/
- Install to your computer. For example, I install at: 'C:\TDM-GCC-64'.
- Input "setenv('MW_MINGW64_LOC','YourTDMGCCPath'); mex -setup;" to check whether mex is set up successfully or not.

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

Command window screen outputs:

```
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
```

1.3. Mex Splines_interpolation c++ files

If using Windows system, please uncomment Line "setenv('MW_MINGW64_LOC','C:\TDM-GCC-64')", correct 'C:\TDM-GCC-64' to 'YourTDMGCCPath', and then run Section 1.

Command window screen outputs:

```
------------ 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 ------------
```

Section 2. Load DIC images and set DIC parameters

Load reference image and the deformed image. (You need to put your images on the MATLAB active path beforehand.) Run Section 2,

```
--- Please load first image ---
--- Please load second image ---
```

And then you can choose your images in the folder.

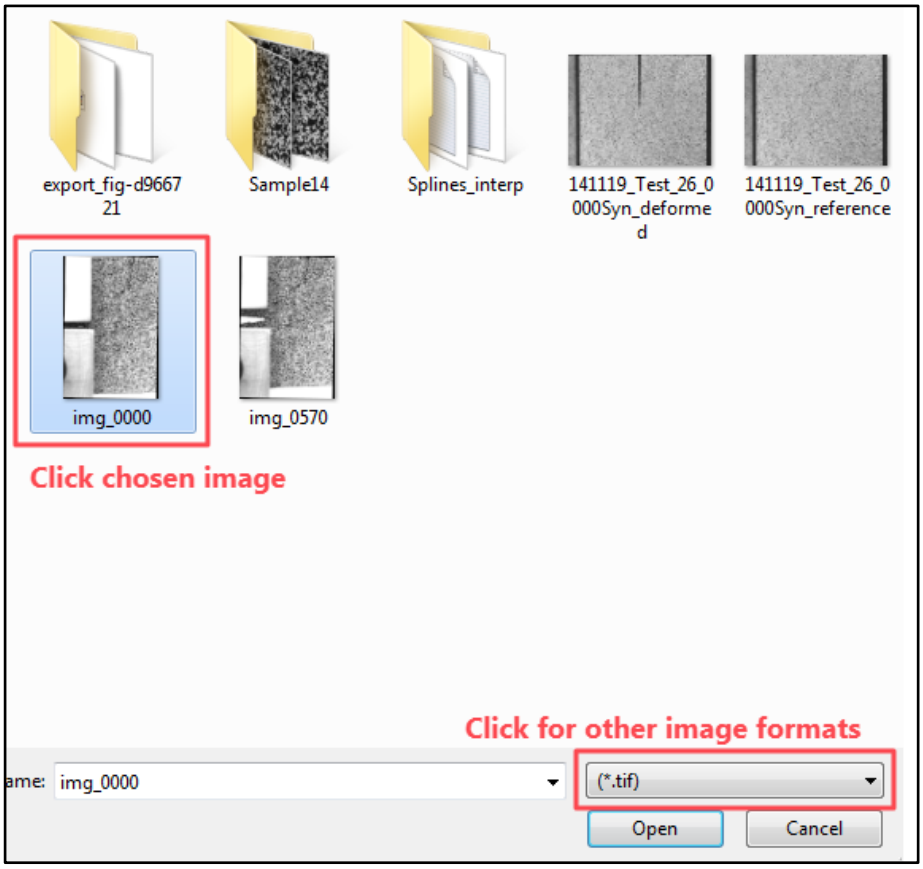


Figure 1. Load DIC images. Click reference image and deformed image.

Comment of current version: I set first image as reference image and set second image as deformed image. 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)
```

Put in "0" if you want to continue uploading images, and put in "1" if you want to stop uploading images. Current version: Please always Put "1".

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

Choose interest region ZOI and DIC parameters

After that, you are asked to choose ZOI left-top and right-bottom two corner points from the image directly. On the screen, it shows:

```
--- Choose ZOI two boundary points from the left-top to the right-bottom ---
```

First click left-top point and then right-bottom point to define your interest region in the DIC image.

Figure 2. Click left-top and right-bottom corner points to define DIC interest region ZOI.

After clicking, the command window screen also outputs the coordinates of clicked corner points.

E.g., in the above DIC speckle image, I define an interest region ZOI where

```
The left-top coordinates are (322.786,74.730)
The right-bottom coordinates are (750.063,1128.439)
```

Comment: If left-top or right-bottom corner points are clicked out of image borders, it will automatically contract back to the original DIC image corners.

Comment: If you don't want to define ZOI 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.

Then you are asked to choose 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? ---
--- What is the subset step size? ---
```

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.

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

We use FFT cross correlation for 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? ---
```

You are asked for initial guess search zone size. Usually this value could be larger than the displacement amplitude. For example, we put in "70" in our example. Then the integer initial search starts and it's usually very fast.

```
Init search using FFT cross correlation on grid: MxN
```

Once the integer searches are done, on the screen, you will be asked again whether you want to increase or decrease your initial guess search zone size.

```
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 the search process.

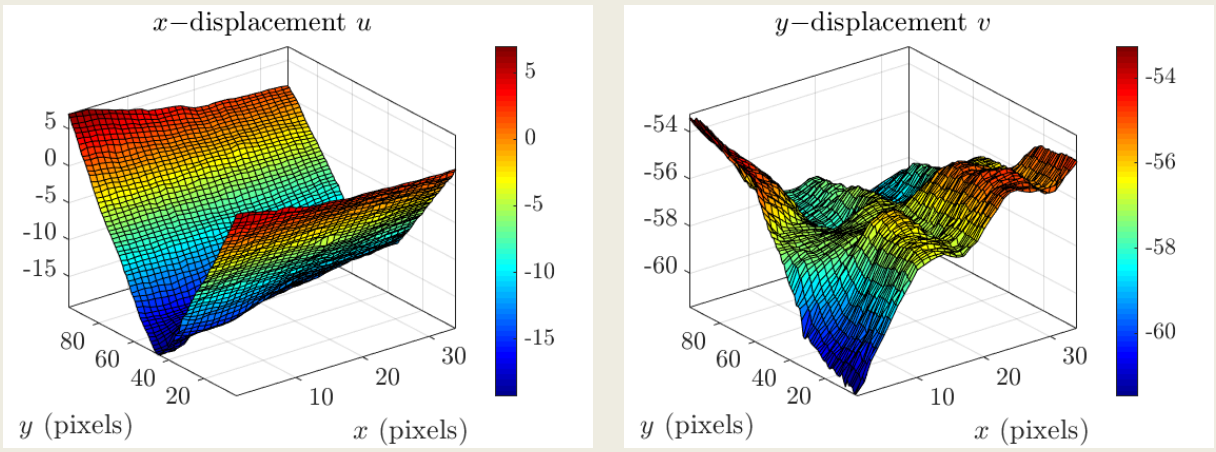


Figure 3. Solved displacement fields by FFT cross correlation.

Next, due to there may be some bad initial guesses, you can manually eliminate them by setting upper bound and the lower bound of the displacement.

```
Do you clear bad points by setting upper/lower bounds once more? (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. And then you will be asked until you think the bad points removal is good enough.
If you put in "1", you will skip this process.

```
% ======= 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)
```

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

```
% ======= Find bad initial guess 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)';
```

Directly clicking all the bad points and then click "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

We run first ALDIC ADMM iteration local step in this section. Before running the code, we need to set up parallel pools or tell MATLAB we don't want to use parallel pools.

```
***** Start step1 Subproblem1 *****
How many parallel pools to open? (Put in 1 if no parallel computing)
```

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).

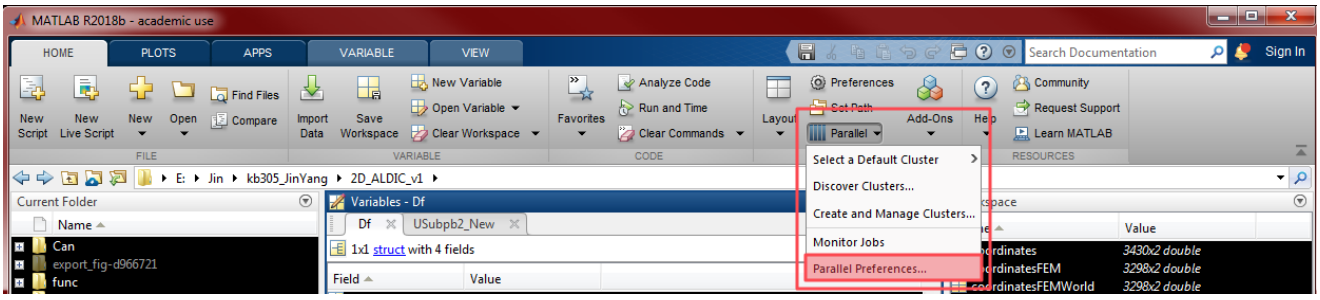
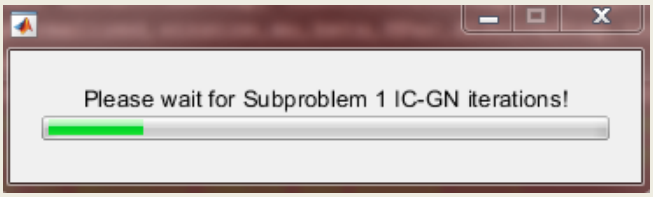


Figure 4. Set MATLAB Parallel Preferences.

Run this section, ALDIC local step using IC-GN algorithm will start and a wait bar will pop out automatically.

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



When the solver finishes, a report will print in the command screen and you can further remove some bad points manually.

```
Local step bad subsets total # is: xxxxxx
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)
```

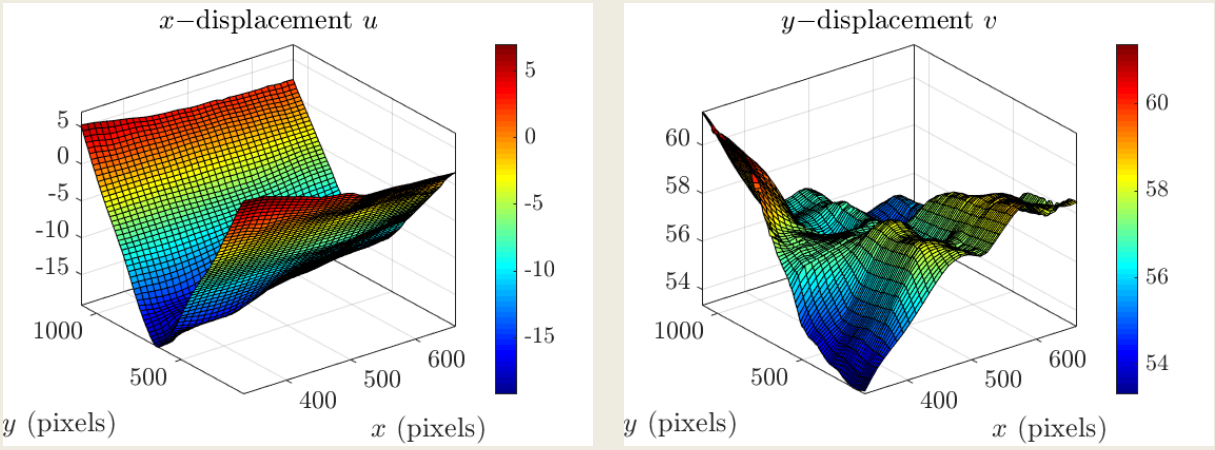


Figure 5. Solved displacement fields from ALDIC first local step.

Next, we start to remove all the local bad points, where the removal process is same with Section §3.

If you put in "0", you will further be asked to set the upper bound and lower bound of the x and y displacements. And then you will be asked until you think the bad points removal is good enough.
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)
```

```
% ======= Find bad initial guess 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)';
```

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

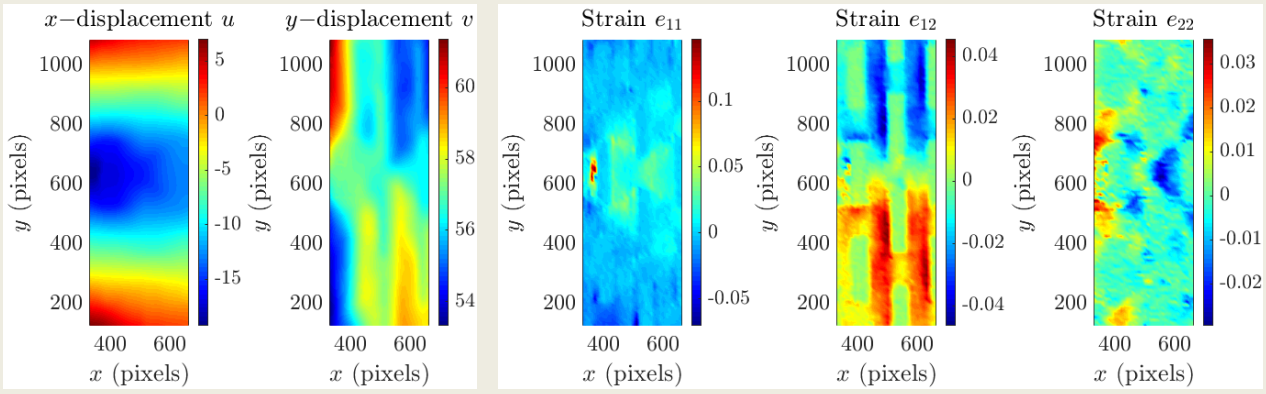


Figure 6. Solved displacement fields and deformation gradient tensors from ALDIC ADMM first local step.

Section 5. ALDIC first global step

After first ALDIC ADMM iteration local step, we run global step in this section. Both finite difference and finite element methods can be applied. For regular grid meshes, both finite difference and finite element method work very well. For arbitrary mesh, finite element method is much easier to apply (Our h-adaptive mesh ALDIC code will come soon. Please contact me if you want finite element method version code now: jyang526@wisc.edu).

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

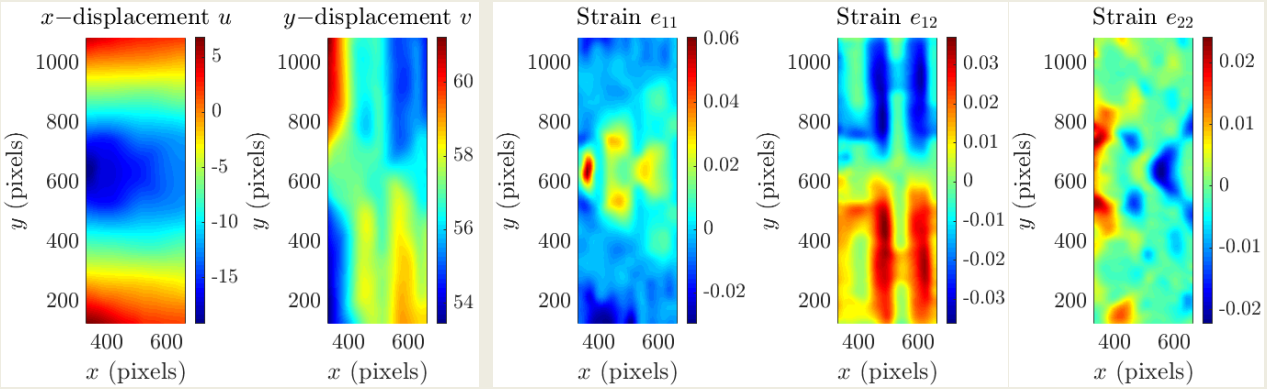


Figure 7. Solved displacement fields and deformation gradient tensors from ALDIC ADMM first global step.

Section 6. ALDIC ADMM iterations

Then we go to overall ALDIC Alternating direction method of multipliers (ADMM) iterations. Run this section, and ADMM iterations will run automatically.
The iteration stop tolerance threshold is set as 1e-4 by default. But this threshold can be decided manually by modifying value of "tol2" in the third line of 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
**********************************
```

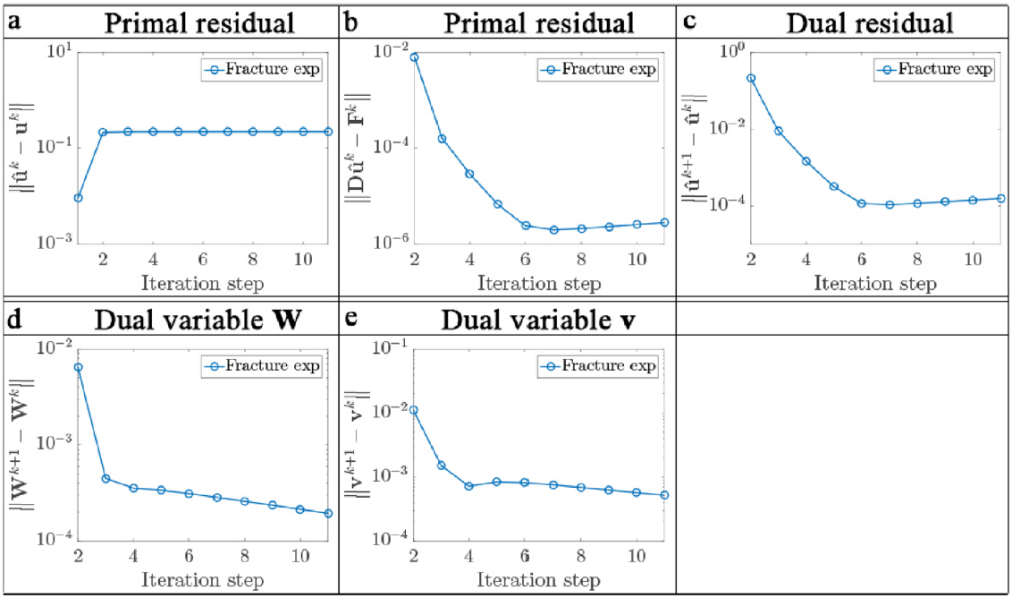


Figure 8. ALDIC ADMM algorithm converges after 6 iterations.

Section 7. Check convergence

This part is to check convergence of ALDIC ADMM iterations. This section can be skipped if you don't need it.

Section 8. Post processing: compute strain

8.1. Smooth displacement field if needed

In the Compute-strain section, let's at first smooth the displacement field if needed. In general, if the displacement field is already quite smooth, please don't smooth the displacement field anymore.

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

If you put in "0", Here I use the Gaussian smooth filter with standard deviation 0.5,

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

You can do this smooth process many times until you are satisfied.
If you think you should have stronger smoothing effects, please manually change the Gaussian filter parameter "DispFilterSize=0; DispFilterStd=0".

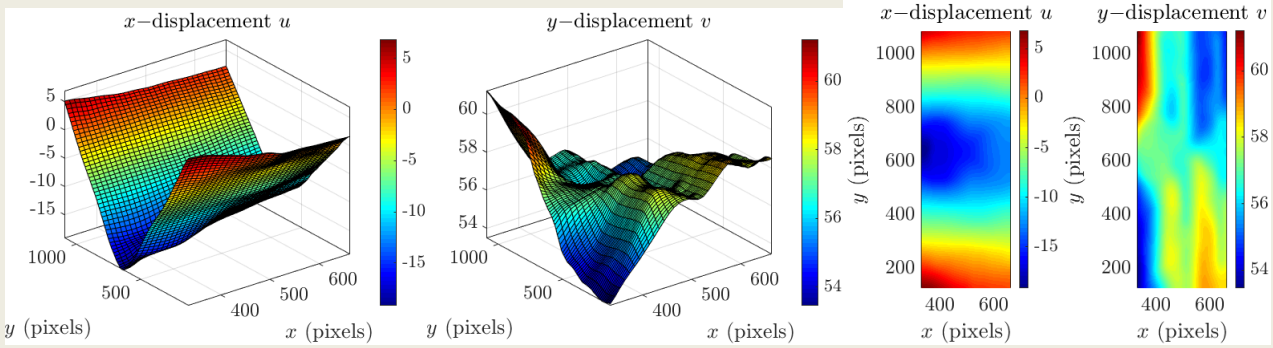


Figure 9. Solved displacement fields by ALDIC method.

8.2. Compute the strain field

After smooth the displacement fields, we introduce three computing strain methods.

- First method is based on the ALDIC automatically solved deformation gradient tensor.
- Second method is based on central finite difference of solved displacement fields.
- 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 size.

You can choose what method to use following this sentence:

```
What method to use to compute strain? (0-None; 1-Finite difference; 2-Plane fitting)
```

If input "2", please also input half window size for Plane fitting.

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

E.g., All the three methods to compute the strain field of our example are shown in Figure 10.

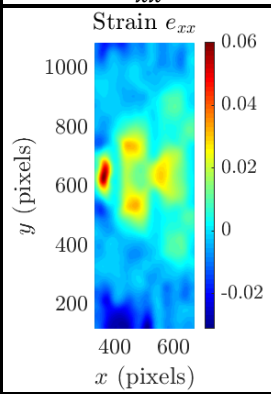
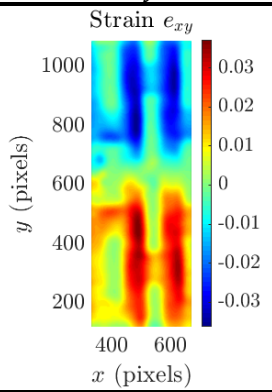
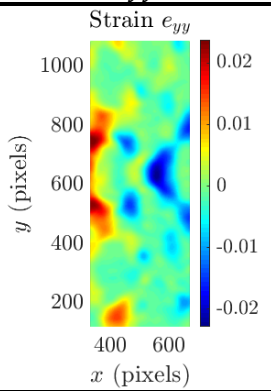
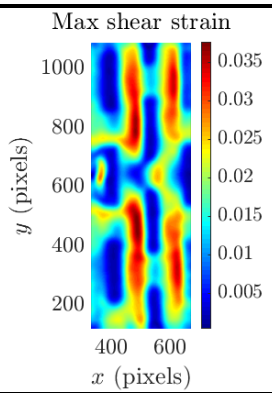
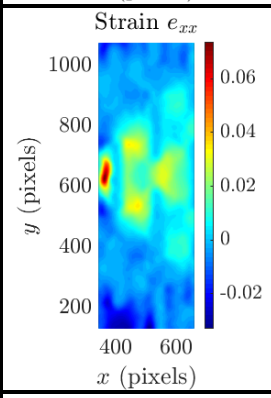
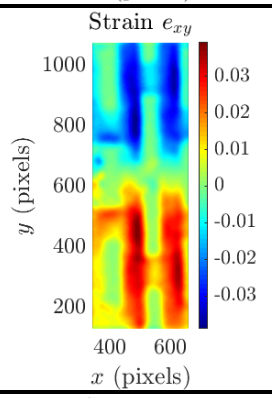
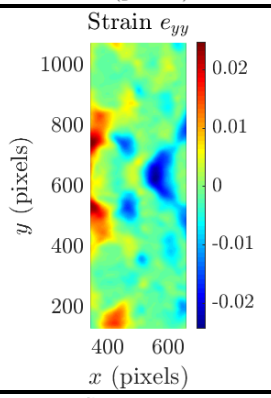
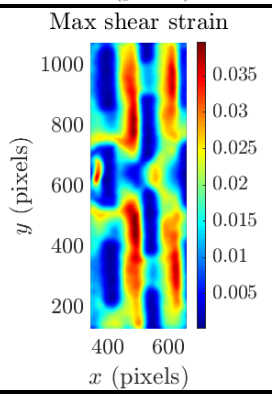
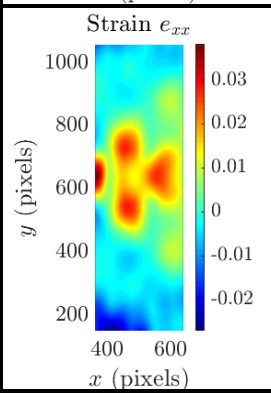
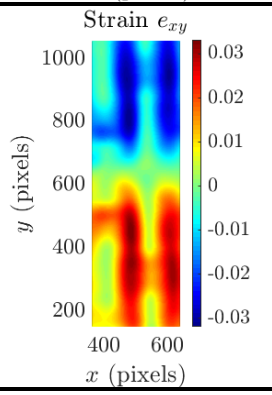
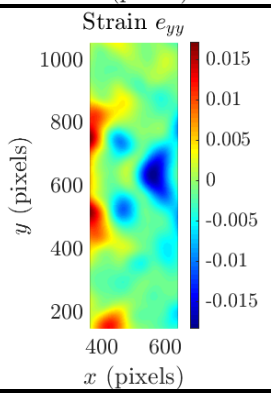
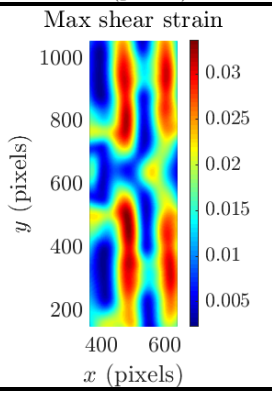
Method	e_{xx}	e_{xy}	e_{yy}	Max shear strain
Method 0 ALDIC directly solved strain				
Method 1 Central difference of displacement field				
Method 2 Plane fitting method (half window size: 3 pixels)				

Figure 10. Solved strain fields by ALDIC method.

Acknowledgment

I am grateful to Dr. Louisa Avellar for sharing her unpublished images of fracture. 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.