
Finite Element Global Digital Image Correlation (FE-Global-DIC) Code Manual (v1.1)

Jin Yang^{1,2†}

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

Email: [†]jyang526@wisc.edu

Github page: https://github.com/jyang526843/2D_FE_Global_DIC
MATLAB FileExchange page: <https://www.mathworks.com/matlabcentral/fileexchange/82873-2d-finite-element-global-digital-image-correlation-fe-dic>

Last time updated on June 1, 2021

Contents

1	Introduction	3
2	Code installation	3
3	Code Section 1. MATLAB mex setup	4
3.1	Test MATLAB mex setup	5
3.2	Install mex C/C++ compiler	5
3.3	Execute code Section 1	6
4	Code Section 2: Load images and set up DIC parameters & mesh	6
4.1	Load DIC images	8
4.1.1	Load DIC images from certain folder	8
4.1.2	Load DIC images with prefix of image name	8
4.1.3	Load DIC images manually	8
4.2	Define region of interest (ROI)	9
4.3	Set up DIC parameters	11
4.4	Additional parameters setup when dealing with image sequence	11
5	Code Section 3: Computing an initial guess of unknown deformation from FFT-based cross correlation	12
5.1	FFT-based methods to compute initial guess	12
5.2	Remove noise in computed initial guess	16
6	Code Section 4: Finite-element-based Global DIC iterations	17
6.1	Finite element DIC method math formulation	17
6.2	Regularization technique	18
6.3	Find the best coefficient of the regularizer	18
7	Code Section 5: Compute strains	19
7.1	Smooth displacement field if needed	19
7.2	Compute strain fields	19
7.3	Plot and save results	21
8	Code Section 6: Compute stress	24
9	Summary of the MATLAB command window screen outputs in the heterogeneous fracture example: “main_FE_GlobalDIC.m”	25
9.1	Code Section 1	25
9.2	Code Section 2	25
9.3	Code Section 3	25
9.4	Code Section 4	26
9.5	Code Section 5	26
9.6	Code Section 6	27
10	Frequently Asked Questions (FAQs)	28

10.1 About MATLAB mex set up	28
10.1.1 Where do I install a TDM-gcc compiler?	28
10.1.2 Mex permission denied	28
10.1.3 File “ba_interp2.cpp” is not found	28
10.2 About the algorithm	29
10.2.1 MATLAB reports error “Undefined function or variable”	29
10.2.2 Which image to load as the first image?	29
10.2.3 How to compute other types of strains?	29
Acknowledgements	31
References	31

1 Introduction

Digital image correlation (DIC) technique is a powerful experimental tool for measuring full-field displacements and strains. Most current DIC algorithms can be categorized into either *local* or finite-element-based *global* methods, see Fig. 1. As with most experimental approaches, there are drawbacks with each of these methods. In the local method the subset deformations are estimated independently and the computed displacement field may not necessarily be kinematically compatible. Thus, the deformation gradients can be noisy, especially when using small subsets. Although the global method often enforces kinematic compatibility, it generally incurs substantially greater computational costs than its local counterpart, which is especially significant for large data sets. Recently, we have presented a new hybrid DIC algorithm, called *augmented Lagrangian digital image correlation (ALDIC)* [1, 2], which combines the advantages of both the local (fast computation times) and global (compatible displacement field) methods.

ALDIC code is freely available at Github and MATLAB File Exchange (link: [3]). We demonstrated that our ALDIC algorithm has high accuracy and precision while maintaining low computational cost, and is a significant improvement compared to current local and global DIC methods. For a review of both local and global DIC methods, and details of this new proposed ALDIC method, please see our paper [1] (full text can also be requested at [4]).

In addition, we also provide the finite-element-based global DIC (FE-Global-DIC) MATLAB code as an open source to the community. This manual is to guide users/readers to run the finite-element-based global DIC method.

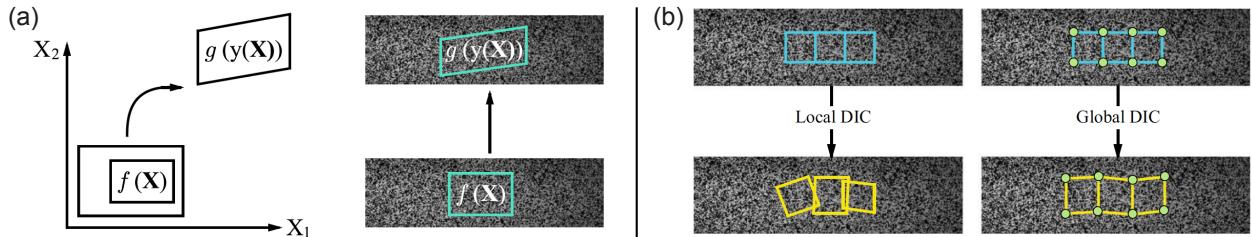


Figure 1: (a) Schematic showing a DIC reference image $f(\mathbf{X})$, with a general speckle pattern, deforming into the deformed image $g(\mathbf{y}(\mathbf{X}))$ under some mapping \mathbf{y} . \mathbf{X} and \mathbf{y} coordinates are in the reference and deformed images, respectively. (b) A schematic comparison between the local DIC method (left), where all the subsets are analyzed independently, and the finite-element-based global DIC method (right), where a global basis set is used to represent the full-field deformation.

2 Code installation

FE-Global-DIC code can be downloaded at [5]. It has been tested on MATLAB versions later than R2018a.

To install the code, please download and unzip the code folder and put this folder on the MATLAB current working path.

Run the whole mfile Run the selected section

The screenshot shows the MATLAB Editor window with the file `/Users/yangjin/Documents/MATLAB/2D_Global_DIC/main_FE_GlobalDIC.m`. The code is divided into two sections:

- Code Section 1** (highlighted in red): Lines 8-19. It includes comments about finite-element-based global DIC, author information, and a timestamp. It also contains code to clear the MATLAB environment, set up Spline interpolation, and compile a mex function.
- Code Section 2** (highlighted in yellow): Lines 20-38. It includes comments for section 2 start, reading images, setting ROI ranges, normalizing images, initializing variable storage, and defining a ResultFEMesh cell.

Annotations include a red arrow pointing to "Code Section 1" and a yellow arrow pointing to "Code Section 2". A red box highlights the "Run Section" button in the MATLAB toolbar, which is located next to the "Run" button. The status bar at the bottom shows "Ln 5 Col 37".

```

1 % Finite-element-based global DIC (FE-Global-DIC)
2 % Author: Jin Yang, Postdoc @UW-Madison; PhD @Caltech 19'
3 % Contact: aldicdvc@gmail.com; jyang526@wisc.edu
4 % 2015.04.06,07; 2016.03.04; 2020.1|
5 %
6 %
7 %
8 % Section 1: Clear MATLAB environment & mex set up Spline interpolation
9 - close all; clear; clc; clearvars -global
10 - fprintf('----- Section 1 Start ----- \n')
11 - setenv('MW_MINGW64_LOC','C:\TDM-GCC-64')
12 - % cd("./Splines_interp/lib_matlab"); CompileLib; cd("../.."); %% mex bi-cubic spline interpola
13 - % addpath("./Splines_interp/lib_matlab"); % dbstop if error % Old version codes.
14 - mex -O ba_interp2.cpp;
15 - addpath("./func"); addpath("./src"); addpath("./plotFiles/");
16 - % addpath("./YOUR IMAGE FOLDER");
17 - fprintf('----- Section 1 Done ----- \n \n')
18 -
19 -
20 % Section 2: Load DIC parameters and set up DIC parameters
21 - fprintf('----- Section 2 Start ----- \n')
22 - % ===== Read images =====
23 - [file_name,Img,DICpara] = ReadImage; close all;
24 - % ##### Uncomment the line below to change the DIC computing ROI manually #####
25 - %gridxROIRange = [gridxROIRange1,gridxROIRange2]; gridyROIRange = [Val1, Val2];
26 - %gridxROIRange = [224,918]; gridyROIRange = [787,1162];
27 - % ===== Normalize images =====
28 - [ImgNormalized,DICpara.gridxyROIRange] = funNormalizeImg(Img,DICpara.gridxyROIRange);
29 - % ===== Initialize variable storage =====
30 - ResultDisp = cell(length(ImgNormalized)-1,1);
31 - ResultDefGrad = cell(length(ImgNormalized)-1,1);
32 - ResultStrain = cell(length(ImgNormalized)-1,1);
33 - ResultFEMesh = cell(cell(ceil((length(ImgNormalized)-1)/DICpara.ImgSeqIncUnit),1)); % For incremental DIC
34 - fprintf('----- Section 2 Done ----- \n \n')
35 -
36 -
37 - % Start each frame in an image sequence
38

```

Figure 2: Main file of the FE-Global-DIC code: “main_FE_GlobalDIC.m”. Each section can be executed in order by clicking the “Run Section” button.

After opening the main file “main_FE_GlobalDIC.m”, as shown in Fig. 2, FE-Global-DIC code can be executed by each section. Once you are familiar with the FE-Global-DIC code, you can execute the whole main file by clicking the “Run” button to run the whole file (“EDITOR >> RUN >> ”). FE-Global-DIC code is easy to modify based on users’ custom parameter choices. In this code manual, we will introduce the FE-Global-DIC code section by section.

3 Code 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 linear, bi-cubic (by default) and bi-cubic splines interpolations

The image shows a MATLAB Command Window titled "Command Window". The window displays the following text:

```

>> mex -setup
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

```

Figure 3: Message display on the command window when a mex C/C++ compiler is stalled successfully.

are implemented in this code. For example, by default we use bi-cubic interpolations where the associated mex set up file is called “ba_interp2.cpp” [6].

3.1 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 3.2 and jump to Section 3.3.

3.2 Install mex C/C++ compiler

The step of installing mex C/C++ compilers is a common step for users to run C/C++ codes with MATLAB. Mac users usually don’t come across the error message from mex C/C++ compilers. For Windows users, you can follow these steps to install mex C/C++ compiler. More details can be found in [7, 8].

- 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-64’¹.
- Restart MATLAB and input these codes on the command window:


```
setenv('MW_MINGW64_LOC', 'YourTDMGCCPath'); mex -setup;
```

 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. For example, if it’s installed at ‘C:\TDM-GCC-64’, please replace previous ‘YourTDMGCCPath’ with ‘C:\TDM-GCC-64’. If a mex C/C++ compiler is installed successfully, a message similar to (Fig. 3) will display on the MATLAB command window.

¹In practice, we find that this TDM-gcc compiler only works if installed on the first level main disks, such as ‘C:\’, ‘D:\’, ‘E:\’, etc.

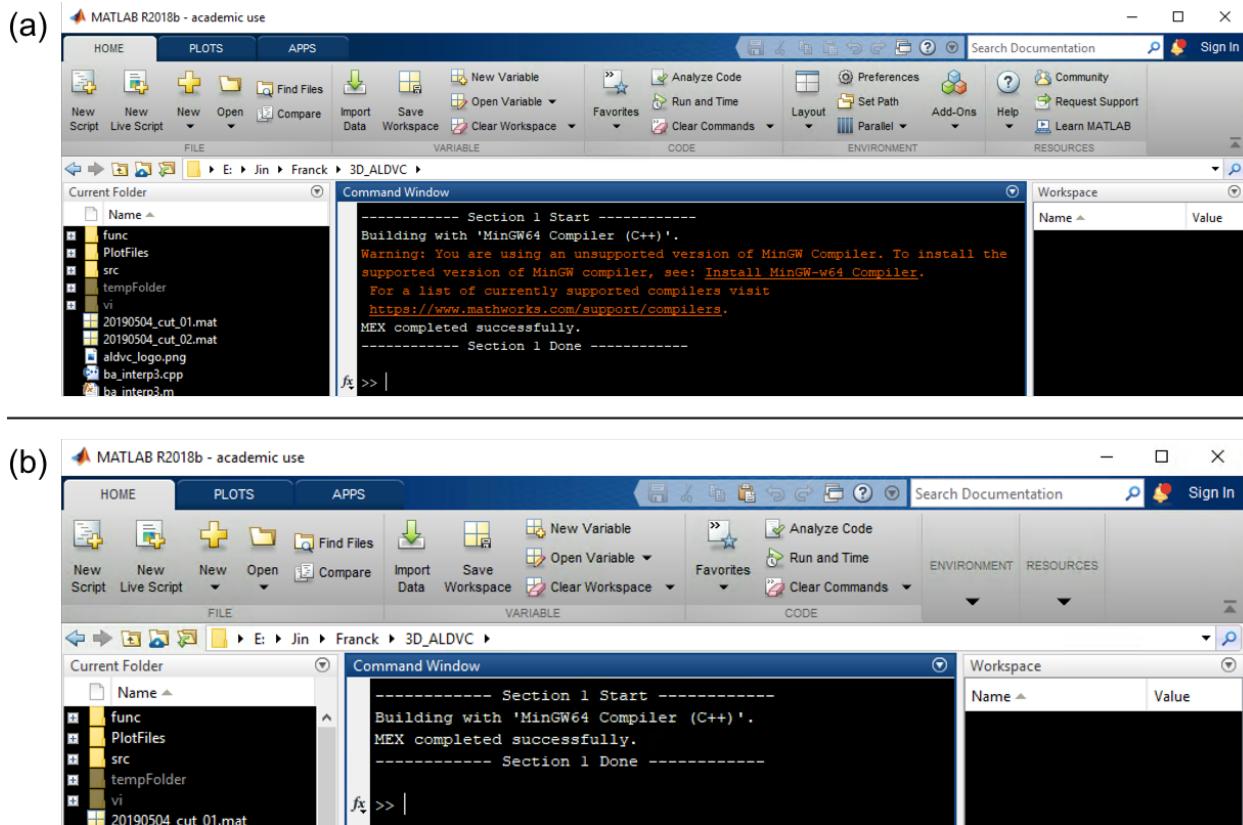


Figure 4: FE-Global-DIC code Section 1 is executed successfully and a message similar to (a) or (b) will display on the command window.

3.3 Execute code Section 1

Once a mex C/C++ compiler is installed, we can execute main_FE_GlobalDIC.m code Section 1 and a successful message will display on the MATLAB command window, see Fig. 4.

4 Code Section 2: Load images and set up DIC parameters & mesh

This section is to load DIC reference and deformed images and set up DIC parameters. First, please put your images on the MATLAB working path. After executing this section, all the FE-Global-DIC associated parameters will be stored in the workspace in “DICpara”. All the mesh properties will be stored in “DICmesh” after executing code 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

DICpara variable	DIC parameter	Description and comments
winsize	Subset window size [winsize_x, winsize_y]	Search area size to compute the initial guess of the displacement fields (code section 3) by the FFT-based cross correlation
winstepsize	Subset window step [winstepsize_x, winstepsize_y]	The finite element size, or the distance between neighboring nodal points
gridxyROIRange	DIC region of interest (ROI)	ROI can be defined by clicking top-left and bottom-right corner points.
ImgSize	Image size	Size of 2D DIC images
ImgSeqIncUnit	To decide to perform cumulative or incremental DIC mode	Cumulative DIC is always to compare with the first reference frame; incremental DIC could update the reference frame.
ImgSeqInc-ROIUpdate-OrNot	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.
NewFFTSearc	Method to update initial guess to solve an image sequence	Result of last frame can be assigned as the initial guess for the next frame.
interpmethod	Interpolation scheme of volumetric image grayscale at subvoxels	Interpolation scheme can be chosen from {'linear', 'cubic'(recommended), 'spline'}.
tol	Tolerance threshold of FE-Global-DIC iterations	E.g., tol takes a value of 10^{-4} pixels (px) by default.
alpha	Coefficient of the added regularization penalties	See Sections 6.2-6.3 for more details.

Table 2: Summary of DIC mesh in “DICmesh” structure

DICmesh variable	DIC parameter	Description and comments
coordinatesFEM elementsFEM	Coordinates of nodal points in the finite element mesh and their connectivity	Linear 4-node quadrilateral (Q4) elements are used here. 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.

4.1 Load DIC images

MATLAB command screen displays these lines to allow user to select their choice to load DIC images, which will be explained in Section 4.1.1-4.1.3.

```
1 ----- Section 2 Start -----
2 Choose method to load images:
3 0: Select images folder;
4 1: Use prefix of image names;
5 2: Manually select images.
6 Input here:
```

One comment for the FE-Global-DIC code is that it always manipulate the deformed images and tries to transform them back to the reference image to compute their deformation fields which is based on the *Lagrangian* description. If user wants to track the deformation field in the *Eulerian* description, user can select the reference image as the second image, and select the deformed image as the first image and manipulate the reference image to transform to the current deformed image.

4.1.1 Load DIC images from certain folder

If we select method 0: Select images folder to load DIC images, you will be asked to select the folder path, and all the images within this folder will be loaded automatically. In the *cumulative* DIC mode, the first frame is set to be the fixed reference image by default, while all the subsequent frames in the image sequence are set to be deformed images whose deformations will be tracked in the Lagrangian description. In the *incremental* DIC mode, the reference image can be updated after every certain number ("DICpara.ImgSeqIncUnit") of frames. After loading DIC images, please jump to Section 4.2.

4.1.2 Load DIC images with prefix of image name

If we select method 1: Use prefix of image names to load DIC images, user also needs to provide prefix text words of their DIC image frames (and all these images need to be added to the MATLAB path as well). For example, if all the DIC images are named in the format as: "img_0001.tif", "img_0002.tif", ..., user should input "img_0*.tif" on the MATLAB command screen to load all the DIC images started with prefix "img_0" and in the "tif" format. After loading DIC images, please jump to Section 4.2.

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

4.1.3 Load DIC images manually

If we select method 2: Manually select images, user needs to load DIC images frame by frame manually, see Fig. 6. FE-Global-DIC code sets the first loaded image as reference image and

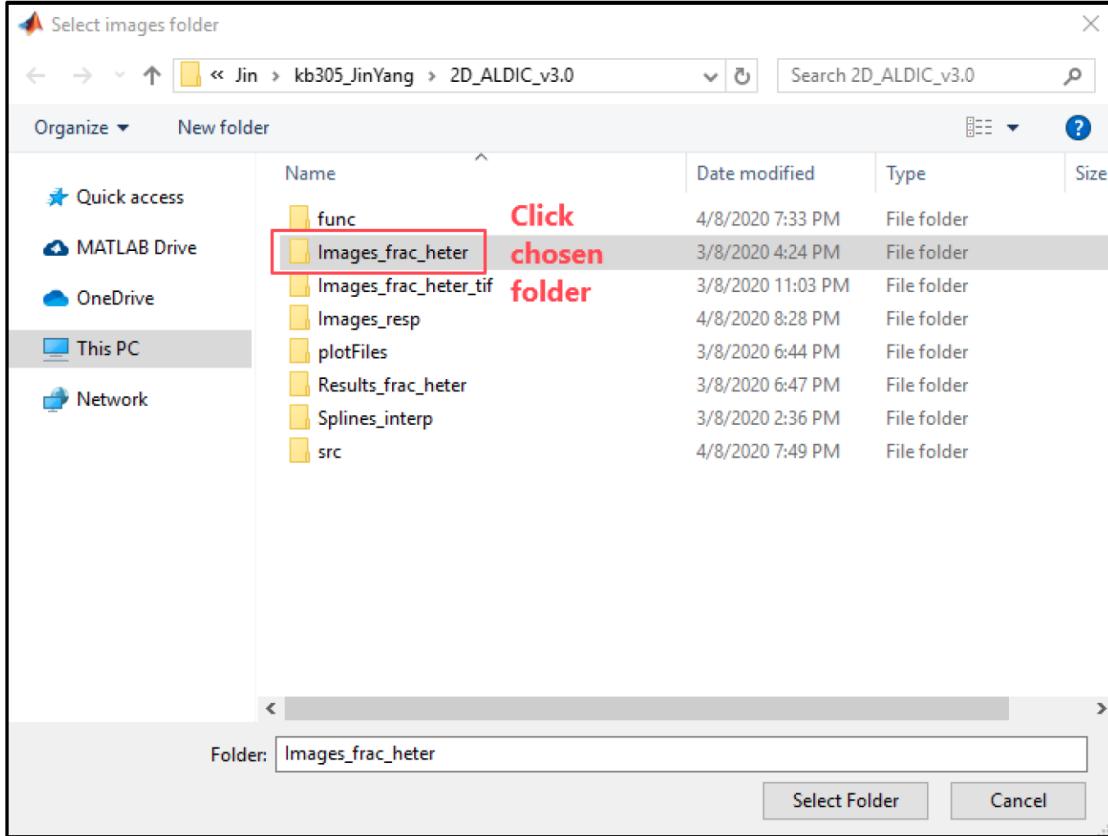


Figure 5: Load DIC images by selecting the folder including all DIC images. For example, after clicking chosen folder "Images_frac_heter", all the images in this folder will be loaded automatically.

other images as deformed images by default.

```

1 --- Please load first image ---
2 --- Please load next image ---
3 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. For example, we can select first reference image as "img_0000.tif", and select second image as "img_0570.tif".

4.2 Define region of interest (ROI)

Execute code Section 2, user can click both the top-left and bottom-right corner points on a popped-out image to define DIC region of interest (ROI). On the command window, it displays:

```

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

```

Then user first click a left-top point and then click a right-bottom point to define ROI directly on the DIC image, see Fig. 7.

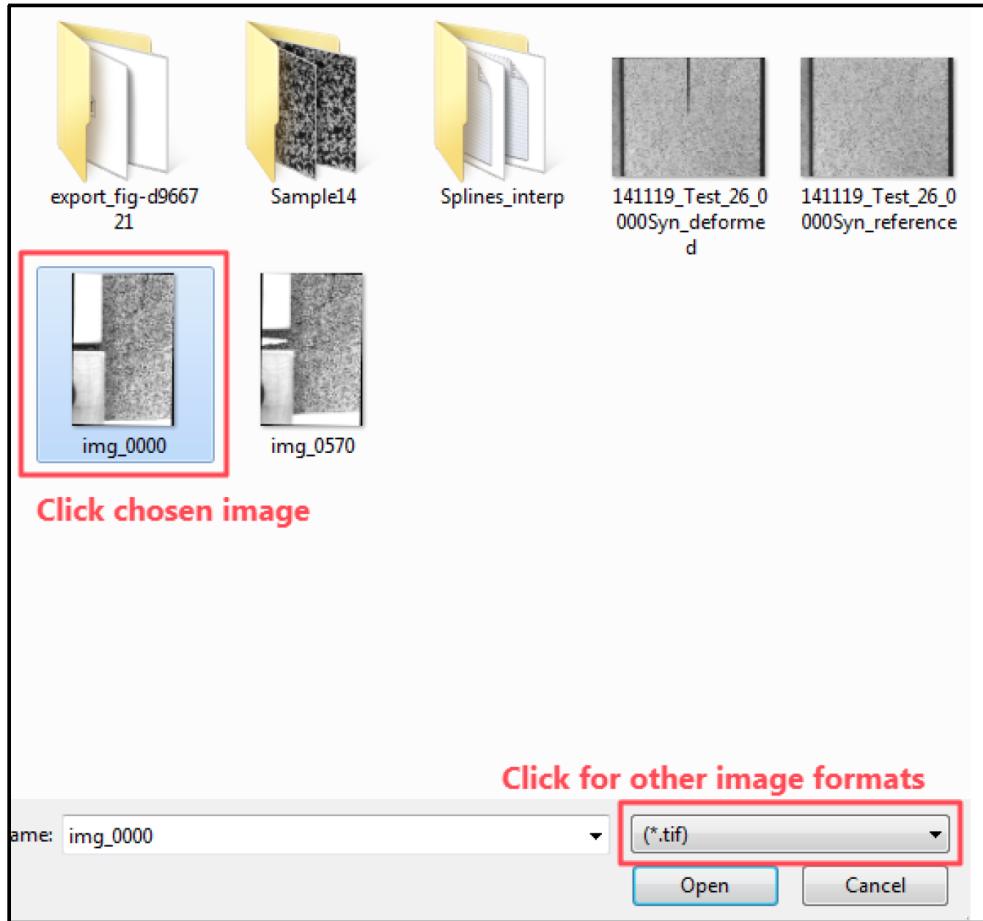


Figure 6: Manually load DIC images by clicking both reference and deformed images.

After clicking both the top-left and bottom-right corner points, the command window screen will display their coordinates in the unit of pixels. E.g., in the image shown in Fig. 7, I define a ROI with top-left and bottom-right corner points are:

```

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

```

Comment 1: If the top-left or bottom-right corner point is clicked out of the image, it will be adjusted to the nearest point on the original DIC image borders automatically.

Comment 2: If user prefers to putting in ROI coordinates instead of directly clicking corner points on the DIC image. Please uncomment line `% gridxROIRange = [gridxROIRange1, gridxROIRange2];` `% gridyROIRange = [gridyROIRange1, gridyROIRange2];` and modify values of "gridxROIRange" and "gridyROIRange" there.

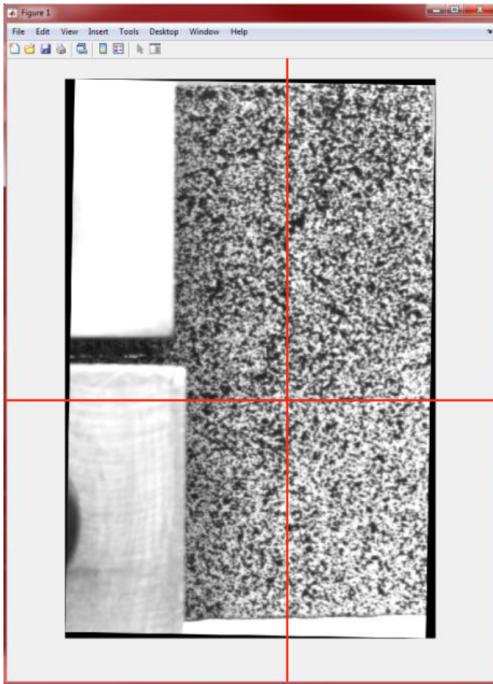


Figure 7: Manually click top-left and bottom-right corner points to define DIC region of interest (ROI).

4.3 Set up DIC parameters

Then user will be asked to decide the finite element size. Current version of this code uses uniform square Q4 FE-meshes.

```
1 --- What is the finite element size? (unit: px) ---
2 Input here: 10
```

4.4 Additional parameters setup when dealing with image sequence

If we upload an image sequence with more than two frames, as for each new 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 voxels, 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.

```
1 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?
2 0: Use last frame;
3 1: Redo initial guess.
```

```
4 Input here:
```

When post-process 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 has been updated. 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:

```
1 --- Choose cumulative or incremental mode ---
2     0: Cumulative(By default);
3     1: Incremental;
4 Input here: 1
```

If you choose to use incremental mode, you will further be inquired to input how often you would like to update the reference image:

```
1 Incremental mode: How many frames to update reference image once?
2 Input here: 10
```

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:

```
1 Update ROI at the same time of updating reference image?
2     0: Do not update ROI;
3     1: Manually(Recommended);
4     2: Automatically;
5 Input here: 1
```

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.

5 Code Section 3: Computing an initial guess of unknown deformation from FFT-based cross correlation

5.1 FFT-based methods to compute initial guess

In this section, the initial estimate of the unknown displacement is calculated by maximizing the (zero-normalized) cross correlation function C_{ZNCC} , see Fig. 8(a-b).

$$C_{ZNCC} = \frac{\int(f - \mu_f)(g - \mu_g)}{\sigma_f \sigma_g} \quad (1)$$

where μ_f, μ_g are average grayscale values of DIC images, f and g ; σ_f, σ_g are the standard deviation of image grayscale values. Above cross correlation can be computed fast using Fourier transform convolution theorem and fast Fourier transform (FFT). For example:

$$C_{CC} = \int f(\mathbf{x})g(\mathbf{x} + \mathbf{u})d\mathbf{x} = \mathcal{F}^{-1} \left[\overline{\mathcal{F}[f(\mathbf{x})]} \circ \mathcal{F}[g(\mathbf{x} + \mathbf{u})] \right] \quad (2)$$

where $\mathcal{F}[\cdot]$ denotes the Fourier transform, $\overline{(\cdot)}$ denotes the complex conjugate, and “ \circ ” denotes the Hadamard product (entry-wise product) and the absolute values are taken entry-wise as well.

The ALDIC code provides three ways to compute the initial guess for an unknown displacement field, as shown in Fig. 8(c-e), and following lines will appear on the screen:

```

1 --- Method to compute an initial guess of displacements ---
2   0: Multigrid search based on an image pyramid
3   1: Whole field search for all the subsets
4   2: Search near manually clicked seeds and then interpolate for the
      full-field
5 Input here:

```

Method 1 (whole field search, Fig. 8(c)) executes FFT-based cross correlation for all local subsets (subsets can overlap), while **Method 2** (Several seeds search, Fig. 8(d)) is to compute near several manually clicked local seeds. Both **Methods 1 & 2** will ask the user to enter the size of the search area. The user should try to search for a small value of the search area size and incrementally increase that value until it exceeds the maximum x- and y-displacements. For example, in the heterogeneous fracture example as shown in Fig. 10, the max magnitude of the displacements are around 60 px, thus the search area size can be chosen as an integer larger than 60. If the chosen FFT search area size is too small, for example, 30 or 50, the computed initial displacements will be noisy (see Fig. 9(a-b)). If this happens, the user needs to continually increase the size of the search area until a good initial guess is found. For example, search area size equals 60 in Fig. 9(c).

```

1 --- What is your initial guess search zone size (pixels)? ---
2 User should start to try a small integer value, and gradually increase the
3 value of the search zone size until it is larger than the magnitudes of
4 |disp u| and |disp v|.
5 Input here: 60
6 Finish initial guess search!

```

To further speed up the above FFT-based optimization process, we also provide a multigrid **Method 0** based on the constructed image pyramids (see Fig. 8(e)). There is no need to provide the search area size in this Method. For example, an initial guess of the above heterogeneous fracture deformation is solved and shown in Fig. 10.

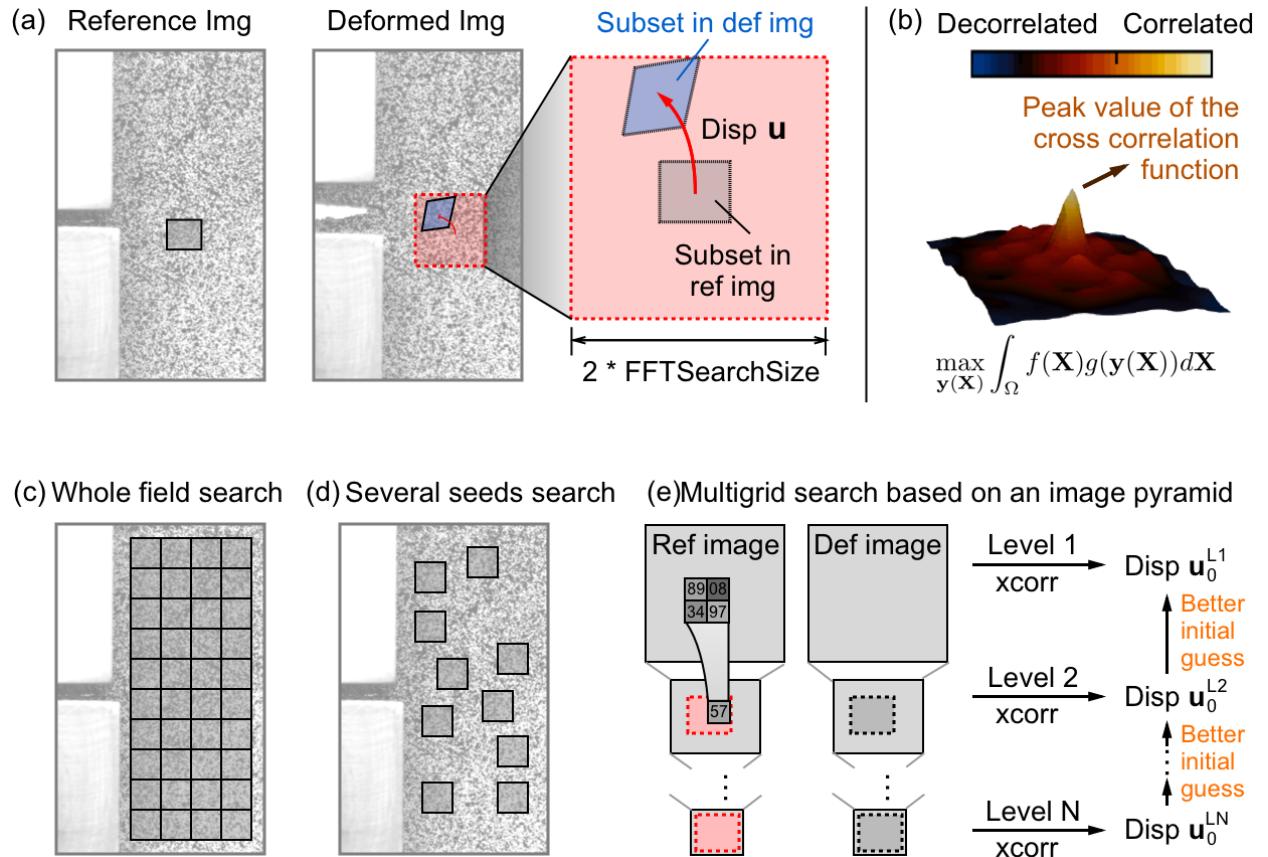


Figure 8: (a-b) The initial estimate of the unknown displacement of a local subset is calculated by maximizing the (zero-normalized) cross correlation function. (c-d) All local subsets or a few manually clicked seeds are analyzed using FFT-based cross correlation to compute the initial guesses of their deformations. (e) A multigrid method based on the constructed image pyramids is proposed to compute the initial estimate of full-field deformation.

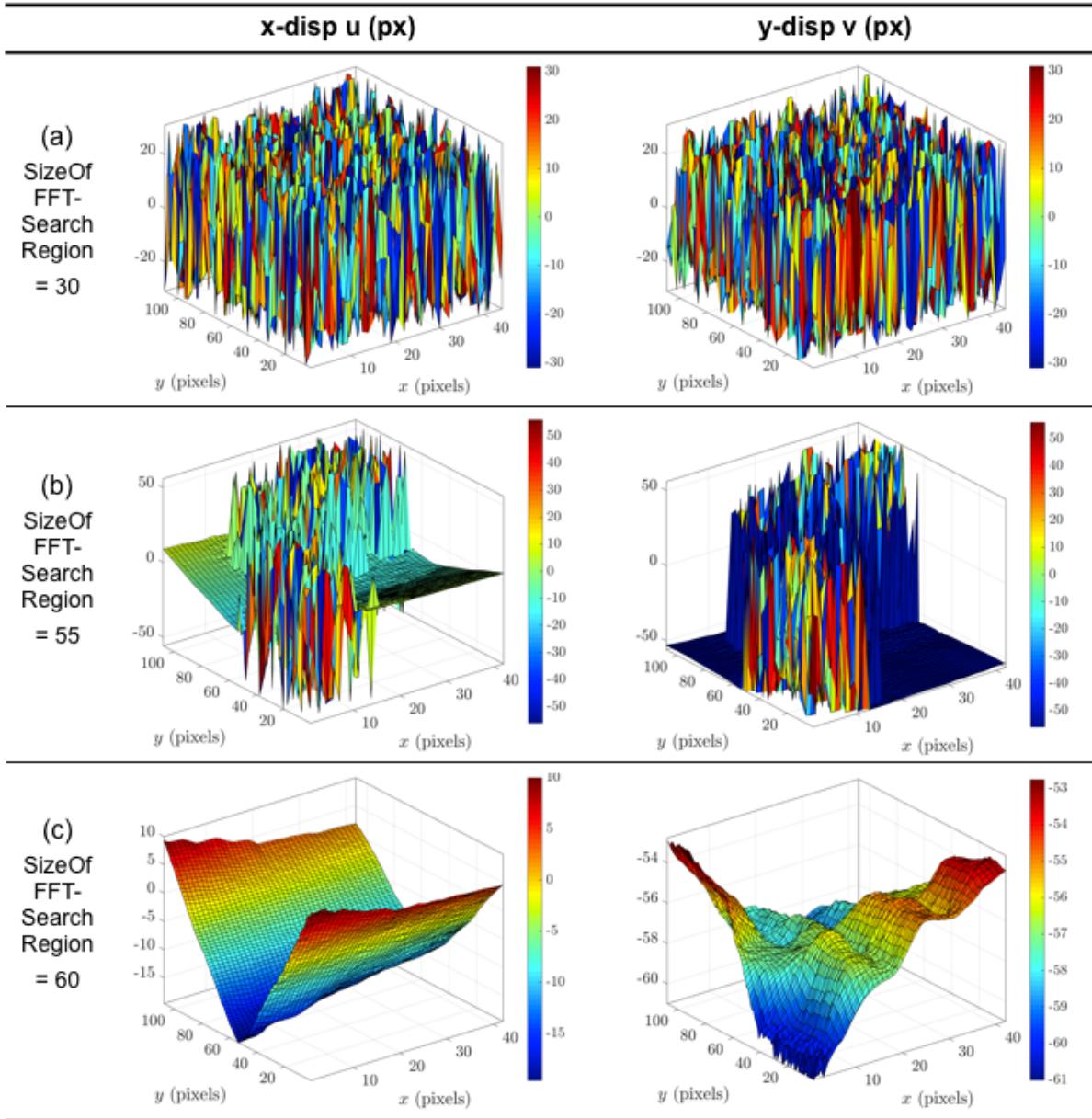


Figure 9: **Method 1** solved displacement fields by the FFT-based cross correlation method with different search area sizes. (a-b) If the chosen size of the search area is too small, computed initial displacements will be super noisy. (c) A good “search area size” should not be smaller than the magnitudes of displacement components ($|dispu|$, $|dispv|$).

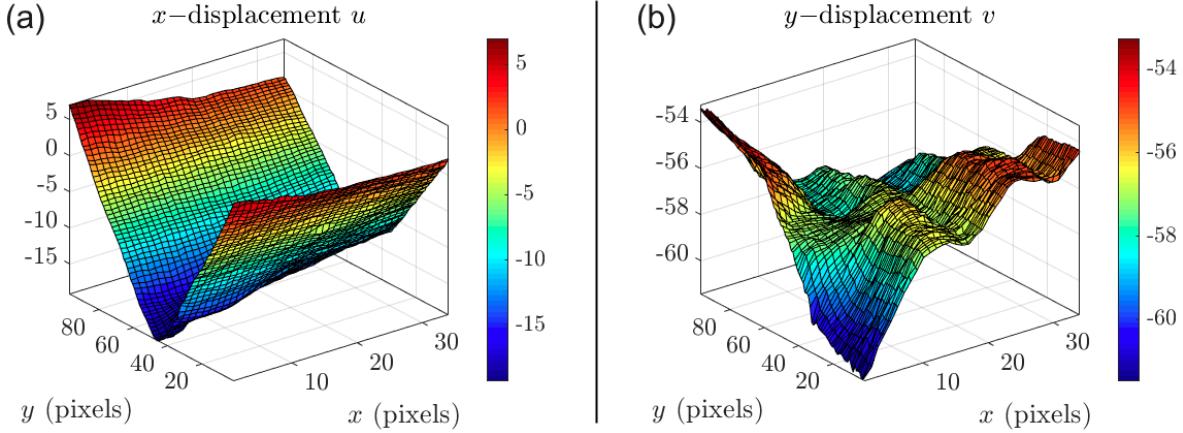


Figure 10: **Method 0** – Multigrid FFTbased cross correlation solver – computed initial estimate of the unknown deformation field.

5.2 Remove noise in computed initial guess

In practice, FFT-based cross correlation method solved initial guess ² may have large noise, the user can further remove these bad points by applying a median filter, setting a q-factor threshold, and setting upper and lower limits of displacements. Users can also continue to delete the bad points directly by clicking them in each image and then press the " Enter " key. However, this step requires some manual work. If noise elimination is necessary, the user can follow the following steps:

- 1 Do you clear bad points by setting upper/lower bounds once more?
- 2 (0-yes; 1-no)

If the user inputs " 0 ", then he/she will be asked to enter the upper and lower limits of the x- and y-displacements. If the user inputs " 1 ", this process will be skipped.

- 1 % ===== Find bad initial guess points manually by setting bounds =====
- 2 What is your upper bound for x-displacement?
- 3 What is your lower bound for x-displacement?
- 4 What is your upper bound for y-displacement?
- 5 What is your lower bound for y-displacement?
- 6 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 removing bad points directly by clicking them in the image.

- 1 % ===== Find bad initial guess points manually =====
- 2 'Do you clear bad points by directly pointing x-disp bad points?
- 3 (0-yes; 1-no);
- 4 'Do you clear bad points by directly pointing y-disp bad points?

²Some designed filters or iterative deformation method (IDM) can further improve the accuracy of initial guess, cf [9, 10, 11, 12]. These are beyond the scope of this code manual.

```
5 (0-yes; 1-no)';
```

Click all the bad points directly, and then press the " Enter " key.

At the end of this section, a finite element mesh is also automatically generated.

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

The image pixel grayscale gradient can also be calculated fast by using a finite difference operator and convolution operation.

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

6 Code Section 4: Finite-element-based Global DIC iterations

6.1 Finite element DIC method math formulation

In the finite element based global DIC method, we represent the global deformation using a global basis set, often based on a finite element formulation [13], i.e.,

$$\mathbf{y}(\mathbf{X}) = \mathbf{X} + \mathbf{u}(\mathbf{X}) = \mathbf{X} + \sum_p u_p \psi_p(\mathbf{X}) \quad (3)$$

where $\psi_p(\mathbf{X})$ are chosen global 4-node quadrilateral (Q4) finite element basis functions and u_p are the unknown degrees of freedom. To minimize the DIC correlation function, for example,

$$C_g = \int_{\Omega} \left| f(\mathbf{X}) - g(\mathbf{X} + \sum_p u_p \psi_p(\mathbf{X})) \right|^2 d\mathbf{X} \quad (4)$$

we can solve this problem iteratively by setting $\mathbf{u}_{k+1} = \mathbf{u}_k + \delta\mathbf{u}$ using the first order approximation

$$g(\mathbf{y}(\mathbf{X})) = g(\mathbf{X} + \mathbf{u}_k(\mathbf{X}) + \delta\mathbf{u}) \approx g(\mathbf{X} + \mathbf{u}_k(\mathbf{X})) + \nabla g \cdot \delta\mathbf{u}(\mathbf{X}) \quad (5)$$

such that

$$C_g \approx \int_{\Omega} \left| f(\mathbf{X}) - g(\mathbf{X} + \mathbf{u}_k(\mathbf{X})) - \left(\sum_p \delta u_p \psi_p(\mathbf{X}) \right) \cdot \nabla g(\mathbf{X}) \right|^2 d\mathbf{X} \quad (6)$$

This leads to a linear equation in $\delta\mathbf{u}$

$$M_{pq} \delta u_q = b_p \quad (7)$$

where

$$M_{pq} = \int_{\Omega} \psi_p^T(\mathbf{X}) (\nabla g(\mathbf{X})) (\nabla g(\mathbf{X}))^T \psi_q(\mathbf{X}) d\mathbf{X} \quad (8)$$

$$b_p = \int_{\Omega} (f(\mathbf{X}) - g(\mathbf{X} + \mathbf{u}_k(\mathbf{X}))) \psi_p^T(\mathbf{X}) \nabla g(\mathbf{X}) d\mathbf{X}. \quad (9)$$

Different from conventional finite element analysis where numerical integrals can be computed using several Gauss points, here, we use pixel-wise summation to approximate numerical integrals in equations (8-9).

6.2 Regularization technique

To speed up the convergence and reduce the noise, we could add regularization penalties onto the DIC correlation function (4).

$$C_{g-\text{regularized}} = \int_{\Omega} |f(\mathbf{X}) - g(\mathbf{X} + \mathbf{u}(\mathbf{X}))|^2 d\mathbf{X} + \alpha |\nabla \mathbf{u}(\mathbf{X})|^2 \quad (10)$$

where α is the coefficient of the added gradient regularizer term ³. For example, user can set α as a fixed value. In the heterogeneous fracture demo case, if we set $\alpha = 10$, set the iteration stopping tolerance as 1e-4, and execute this section, then we will have the following lines displayed on the MATLAB command screen where FE-Global-DIC iterates 11 steps and gets converged.

```

1 %% Section 4
2 fprintf('----- Section 4 Start ----- \n')
3 % %%%%%%%%%%%%%%
4 % Finite element based global DIC iterations
5 % %%%%%%%%%%%%%%
6 DICpara.tol = 1e-4; % iteration stopping threshold
7 alphaList = 10; % Set regularization coefficient, alpha, as 10

----- Section 4 Start -----
--- Global IC-GN iterations ---
3 normW = 0.41832 at iter 1; time cost = 47.4258s
4 normW = 0.040996 at iter 2; time cost = 38.4114s
5 normW = 0.014932 at iter 3; time cost = 39.9557s
6 normW = 0.0067409 at iter 4; time cost = 39.4569s
7 normW = 0.0032884 at iter 5; time cost = 38.5084s
8 normW = 0.0016662 at iter 6; time cost = 39.3641s
9 normW = 0.00086242 at iter 7; time cost = 38.8785s
10 normW = 0.0004522 at iter 8; time cost = 40.3063s
11 normW = 0.00023907 at iter 9; time cost = 38.8822s
12 normW = 0.00012706 at iter 10; time cost = 38.8817s
13 normW = 6.777e-05 at iter 11; time cost = 39.1988s
14 Elapsed time is 439.2666 seconds.
15 ----- Section 4 Done -----

```

6.3 Find the best coefficient of the regularizer

Theoretically, with a smaller value of α , the solved displacements and strains are noisier and there are less effects of the added regularization terms (see Fig. 12(a)). A large value of α may oversmooth the deformation fields, which are preferred by the added regularization penalties (see Fig. 12(c)). In practice, finding the best coefficient of the added regularizer term, α , needs user's experience or other optimization process ⁴. Here, we also provide the code to estimate the best

³There are other types of regularization penalties based on different prior assumptions of the actual deformation fields, such as elastic, curvature-based, fluidic, and hyperelastic penalties.

⁴One benefit of the ALDIC method [1, 14] is that there doesn't come across this "oversmooth" issue in the ALDIC method since there are no smooth-regularization terms added.

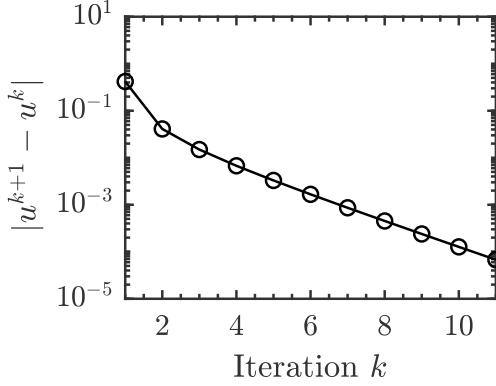


Figure 11: FE-Global-DIC converges after 11 iterations when $\alpha = 10$.

value of α from the modified L-curve method [15]. User needs to uncomment the following line, then the code will tune the best value of α automatically, see Fig. 13.

```

1 % ===== Tune regularization coefficient =====
2 % If you don't know the best alpha (coefficient), please run the following
3 % codes to tune the best value of the coefficient of the regularizer |grad
4 %| ^2:
5 %%%%%% Uncomment the following line to tune the best value of alpha %%%%%%
5 alphaList = [1e-3,1e-2,1e-1,1e0,1e1,1e2,1e3]*mean(DICpara.winstepsizes);

```

7 Code Section 5: Compute strains

7.1 Smooth displacement field if needed

Before computing strains, solved displacement fields can be further denoised if necessary. In most cases, FE-Global-DIC with regularization solved displacement fields are already denoised, so usually there is no need to further smooth solved displacement fields anymore.

```

1 ----- Section 5 Start -----
2 Do you want to smooth displacement? (0-yes; 1-no)1

```

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` ”.

7.2 Compute strain fields

In FE-Global-DIC algorithm, strain fields can be further computed at finite element Gauss points. Strain values at nodal points are extrapolated and averaged from the inside Gauss points. In

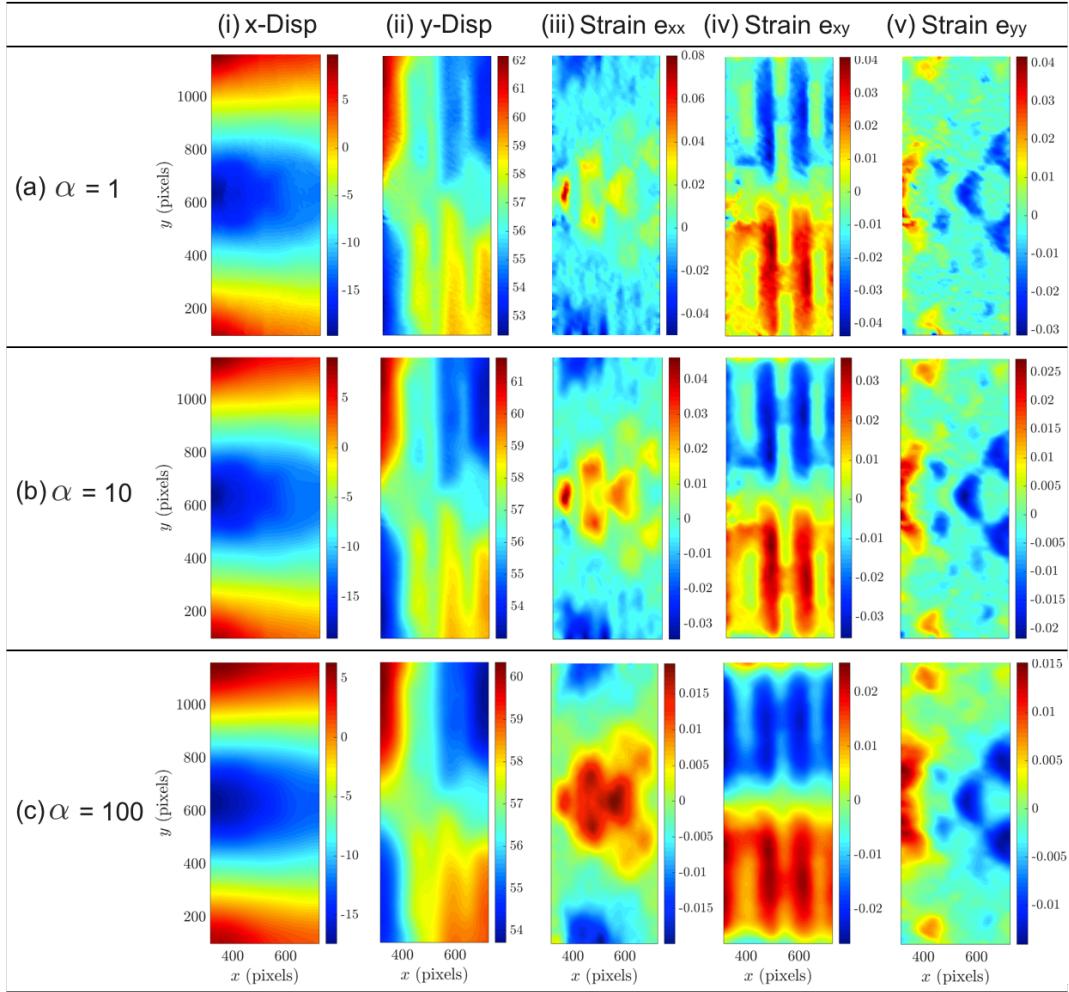


Figure 12: FE-Global-DIC method solved displacement and strain fields with different regularization coefficients, α . With a smaller value of α , the solved displacements and strains are noisier, while a large value of α may oversmooth the deformation fields.

addition, strain fields can also be computed from numerically differentiating solved displacement field. In summary, we implement three methods to compute strain fields.

- (0) Strain field can be computed from **finite element Gauss points**;
- (1) **Central finite difference** of solved displacement field;
- (2) **Plane fitting method** to differentiate solved displacement field. In this method, you will be asked to input half plane width to define the size of the fitted plane;

```

1 What method to use to compute strain?
2   0: Finite element (Recommended);
3   1: Finite difference;
4   2: Plane fitting;

```

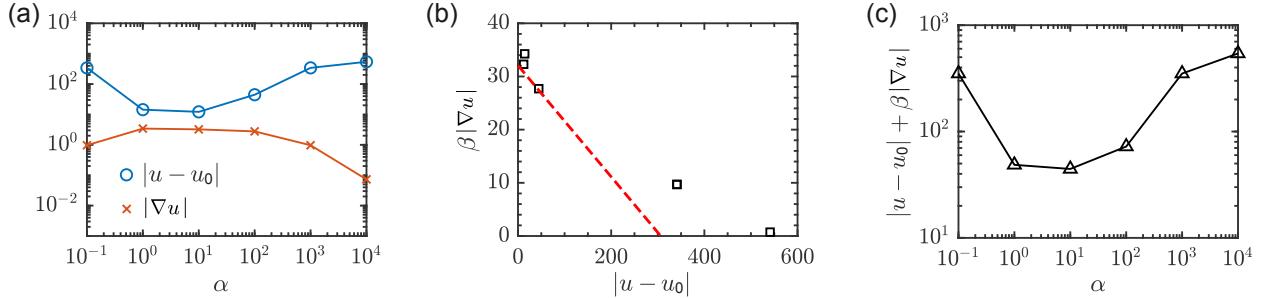


Figure 13: Finding the best coefficient of the regularization by the L-curve method. (a-b) Plots of terms $|\mathbf{u} - \mathbf{u}_0|$ and $|\nabla \mathbf{u}|$. Red dashed line in (b) points out the best α in the L-curve method, which is equivalent to the plot in (c) for the “ $|\mathbf{u} - \mathbf{u}_0| + \beta |\nabla \mathbf{u}|$ ”, where the minimizer corresponds to the best α . Here, β is an empirical number here I set as the finite element size ($\beta = 10$ in the heterogeneous fracture example, which is also the chosen finite element size). To further improve the accuracy of α , I apply a quadratic polynomial to interpolate the data points near the best triangle point in (c), and to further update α with the minimizer of the local quadratic polynomial function.

```
5 Input here: 0
```

If user inputs “ 2 ”, MATLAB command window will display following lines for continuing to define the plane size:

```
1 What is your half window size: % Input half window size for plane fitting
```

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

```
1 Infinitesimal stran or finite strain?
2     0: Infinitesimal stran;
3     1: Eulerian stran;
4     2: Green-Lagrangian stran;
5     3: Others: code by yourself;
6 Input here: 0
```

7.3 Plot and save results

Users can visualize solved displacement and strain field and plot them overlaid with original DIC images, see Fig. 14. Finally, don't forget to save results for future use.

```
1 Save figures into different format:
2 1: jpg(Choose transparency 0~1)
3 2: pdf(Choose transparency = 0)
4 3: Others: Edit codes in ./plotFiles/SaveFigFiles.m
5 Input here:
```

Current version code, overlaying original DIC image can only be saved in the “jpg” format.

```

1 Define transparency for overlaying original images:
2 Input a real number between 0(Only original images)
3 and 1(Only deformation results).
4 Input here(e.g. 0.5): 0.5

```

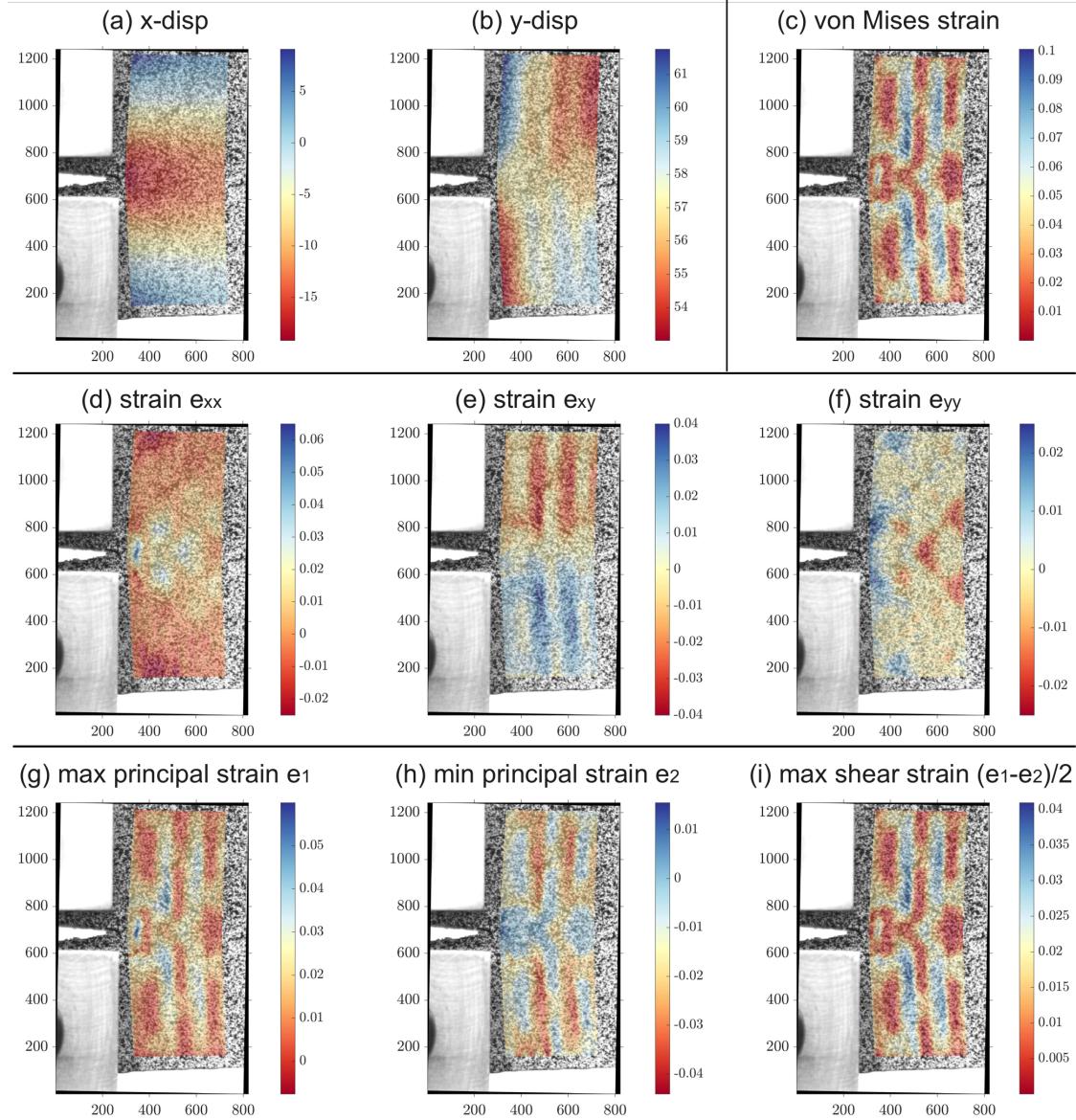


Figure 14: Final 2D FE-Global-DIC code tracked deformation fields with original DIC images overlaid as the background. (a) x-displacement; (b) y-displacement; (c) equivalent von Mises strain; (d-f) infinitesimal strains ϵ_{xx} , ϵ_{xy} , and ϵ_{yy} ; (g-h) maximum and minimum principal strains; (i) max shear strain.

Finally, all the results will be saved in a Matlab matfile, see Table 3.

```

1 %% Save data again including strain solve method

```

Table 3: Summary of DIC results

Variable	Description
DICpara	FE-Global-DIC parameters, see Table 1
DICmesh	FE-Global-DIC finite element mesh details, see Table 2
ResultDisp	Solved displacements
ResultDefGrad	Solved “deformation gradient minus identity” \mathbf{F}
ResultStrain	Solved strains from FE-Global-DIC code Section 8
ResultFEMesh	Stored FE-mesh due to reference frame update in incremental mode
timeICGN	Computation time in FE-Global-DIC iterations
normOrW	Norm of displacement update vector during FE-Global-DIC iterations

```

2 results_name = ['results_FE_globalDIC_', imgname, '_st', num2str(DICpara.
    winstepsize), '_alpha', num2str(DICpara.alpha), '.mat'];
3 save(results_name, 'file_name', 'DICpara', 'DICmesh', 'ResultDisp', ,
    ResultDefGrad', 'ResultStrain', 'ResultFEMesh', ...
4     'normOfW', 'timeICGN');
```

8 Code Section 6: Compute stress

If the material constitutive relation of the specimen is known, the relevant stress field in DIC experiment can be obtained from the calculated strain field.

For example, the constitutive equation in **plane stress** case reads as (Voigt's notation):

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{bmatrix} = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} \quad (11)$$

where E is Young's modulus, ν is Poisson's ratio, γ_{xy} is the engineering shear strain equals two times of infinitesimal shear strain ε_{xy} .

Similarly, the constitutive equation in **plane strain** case reads as (Voigt's notation):

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{bmatrix} = \frac{E(1 - \nu)}{(1 + \nu)(1 - 2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} \quad (12)$$

```
1 ----- Section 6 Start -----
2 Material model to compute Cauchy stress fields:
3   1: Linear elasticity -- Plane stress
4   2: Linear elasticity -- Plane strain
5   3: Others: User needs to edit codes in ./plotFiles/Plotstress.m
6 Input here: 1
7 -----
8 Define Linear elasticity parameters
9 Young's modulus (unit: Pa):
10 Input here (e.g., 69e9): 69e9
11 Poisson's ratio:
12 Input here (e.g., 0.3): 0.3
13 -----
14 Current image frame #: 2/2
15 ----- Section 6 Done -----
```

9 Summary of the MATLAB command window screen outputs in the heterogeneous fracture example: “main_FE_GlobalDIC.m”

This simulation is performed in MATLAB (R2020a, 64-bit) and on a workstation with an Intel i5-4670X CPU with a base clock of 3.40 GHz (4 threads), 16.0 GB memory, and runs under Windows 10.

9.1 Code Section 1

```
1 ----- Section 1 Start -----
2 Building with 'MinGW64 Compiler (C++)'.
3 MEX completed successfully.
4 ----- Section 1 Done -----
```

9.2 Code Section 2

```
1 ----- Section 2 Start -----
2 Choose method to load images:
3   0: Select images folder;
4   1: Use prefix of image names;
5   2: Manually select images.
6 Input here: 2
7 --- Please load first image ---
8 --- Please load next image ---
9 Do you want to load more deformed images? (0-Yes; 1-No)1
10
11 --- Define ROI corner points at the top-left and the bottom-right ---
12 Coordinates of top-left corner point are (344.750,125.750)
13 Coordinates of bottom-right corner point are (736.250,1076.750)
14
15 --- What is the finite element size (unit: px) ? ---
16 Input an integer number: 10
17 ----- Section 2 Done -----
```

9.3 Code Section 3

```
1 Current image frame #: 2/2
2
3 ----- Section 3 Start -----
4
5 --- Method to compute an initial guess of displacements ---
6   0: Multigrid search based on an image pyramid
7   1: Whole field search for all the subsets
8   2: Search near manually clicked seeds and then interpolate for the
      full-field
9 Input here: 0
10
11 Finish setting up mesh and assigning initial value!
```

```

12 --- Start to compute image gradients ---
13 --- Computing image gradients done ---
14 ----- Section 3 Done -----

```

9.4 Code Section 4

```

1 ----- Section 4 Start -----
2 --- Global IC-GN iterations ---
3 normW = 0.12493 at iter 1; time cost = 10.0768s
4 normW = 0.0042001 at iter 2; time cost = 9.0387s
5 normW = 0.00039037 at iter 3; time cost = 9.1134s
6 Elapsed time is 28.2283 seconds.
7 ----- Section 4 Done -----

```

9.5 Code Section 5

```

1 ----- Section 5 Start -----
2 Convert units from pixels to the physical world.
3 %%%%%%%%%%%%%%
4 Results in Section 5 (ResultStrain), Section 6
5 (ResultStress), and all the plots will be converted
6 to the physical world unit instead of the pixel unit.
7 %%%%%%%%%%%%%%
8 If you want to keep the pixel unit, enter '1'.
9 Input here (e.g., mm/px, um/px): 1
10 -----
11 Do you want to smooth displacement? (0-yes; 1-no)
12 Input here: 1
13 -----
14 What method to use to compute strain?
15     0: Finite element (Recommend);
16     1: Finite difference ;
17     2: Plane fitting;
18 Input here: 0
19 -----
20 Infinitesimal strain or finite strain?
21     0: Infinitesimal strain;
22     1: Eulerian strain;
23     2: Green-Lagrangian strain;
24     3: Others: code by yourself;
25 Input here: 0
26 -----
27 Over which image(s) you want to plot the results?
28     0: First image only (if you are using only two images);
29     1: Second and next images;
30 Input here: 1
31 -----
32 Save figures into the format:
33     1: jpg(Choose transparency 0~1)
34     2: pdf(Choose transparency = 1)
35     3: Others: Edit codes in ./plotFiles/SaveFigFiles.m

```

```

36 Input here: 1
37 -----
38 Define transparency for overlaying original images:
39 Input a real number between 0(Only original images)
40 and 0.99 (Non-transparent deformation results).
41 Input here(e.g., 0.5): .5
42 -----
43 Current image frame #: 2/2
44 ----- Section 5 Done -----

```

9.6 Code Section 6

```

1 ----- Section 6 Start -----
2 Material model to compute Cauchy stress fields:
3   1: Linear elasticity -- Plane stress
4   2: Linear elasticity -- Plane strain
5   3: Others: User needs to edit codes in ./plotFiles/Plotstress.m
6 Input here: 1
7 -----
8 Define Linear elasticity parameters
9 Young's modulus (unit: Pa):
10 Input here (e.g., 69e9): 69e9
11 Poisson's ratio:
12 Input here (e.g., 0.3): 0.3
13 -----
14 Current image frame #: 2/2
15 ----- Section 6 Done -----

```

10 Frequently Asked Questions (FAQs)

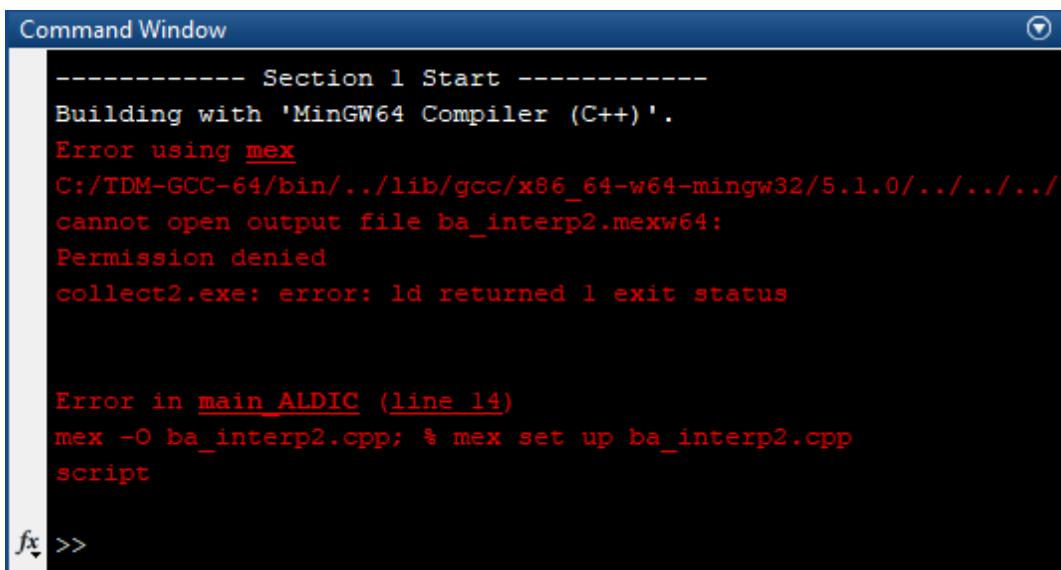
10.1 About MATLAB mex set up

10.1.1 Where do I install a TDM-gcc compiler?

This is a general question for all MATLAB users to apply mex C/C++ compilers where more details can be found online, such as [7, 8]. Users need to install suitable mex C/C++ compilers based on their OS and MATLAB versions. In my experience, the mex C/C++ compiler usually is already installed on MAC system. For Windows users, I usually install the TDM-gcc compiler on a first level main disk, such as “C:, D:, E:”, and higher level disks (like “C/FirstLevel/SecondLevel/...”) may not work. If you have better suggestions, I appreciate you could let me know.

10.1.2 Mex permission denied

MATLAB command window reports an error message when mex permission is denied, see Fig. 15. This error appears because a mex has already been set. To fix this bug, please update the first code section `mex -O ba_interp2.cpp;` to `try mex -O ba_interp2.cpp; catch; end`.



The screenshot shows a MATLAB Command Window. The text output is as follows:

```
----- 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/../../...
cannot open output file ba_interp2.mexw64:
Permission denied
collect2.exe: error: ld returned 1 exit status

Error in main_ALDIC (line 14)
mex -O ba_interp2.cpp; % mex set up ba_interp2.cpp
script

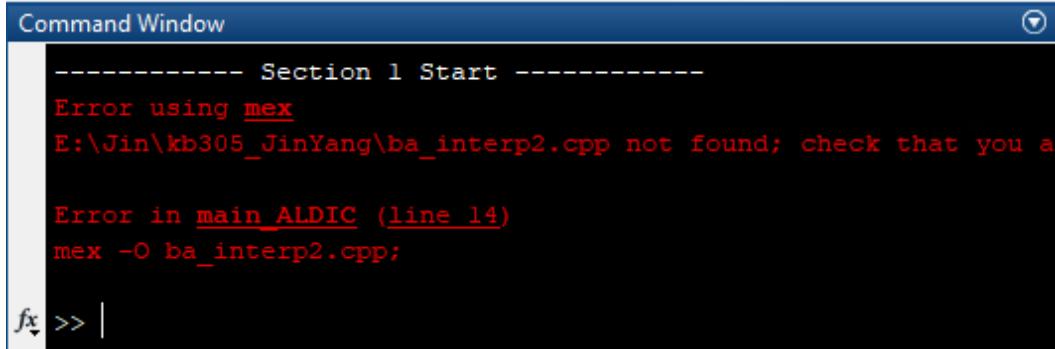
fx >>
```

Figure 15: MATLAB command window reports an error message when mex permission is denied.

10.1.3 File “ba_interp2.cpp” is not found

MATLAB command window reports an error message when the “ba_interp2.cpp” file is not found, see Fig. 16. This error happens because “ba_interp2.cpp” file does not exist in MATLAB “Current Folder”. To fix this bug, please enter the “2D_FE_Global_DIC_v*” folder and then re-run the

“main_FE_Global_DIC.m” code Section 1. Alternatively, the user could also copy and paste a “ba_interp2.cpp” file to your current folder.



The image shows a screenshot of a MATLAB Command Window titled "Command Window". The window displays the following text:

```
----- Section 1 Start -----
Error using mex
E:\Jin\kb305_JinYang\ba_interp2.cpp not found; check that you a

Error in main_ALDIC (line 14)
mex -O ba_interp2.cpp;

fx >> |
```

Figure 16: MATLAB command window reports an error message when the “ba_interp2.cpp” file can not be found.

10.2 About the algorithm

10.2.1 MATLAB reports error “Undefined function or variable”

Please make sure you have already added all the subfolders ('./src','./plotfiles', ...) and the DIC image folder on the MATLAB path.

10.2.2 Which image to load as the first image?

One comment about this finite element based global code is that it is set by default to manipulate the deformed images and attempt to transform deformed images back to the reference image and compute their deformation fields, which is based on the *Lagrangian* description. If the user wants to track the deformation field in the *Eulerian* description, he/she can select the reference image as the second image, and select the deformed image as the first image and manipulate the reference image to transform to the current deformed image.

10.2.3 How to compute other types of strains?

(i) What the finite element based global DIC code solved are displacements ($\mathbf{U} = \mathbf{x}(\mathbf{X}) - \mathbf{X}$) and deformation gradients ($\mathbf{F} = \nabla_{\mathbf{X}}\mathbf{U} = \nabla_{\mathbf{X}}\mathbf{x} - \mathbf{I}$), where \mathbf{x} and \mathbf{X} are the coordinates in deformed and reference configurations, respectively. \mathbf{U} is stored in `ResultDisp`, and \mathbf{F} is stored in `ResultDefGrad`.

In this FE-based global DIC, length of the displacement vector – \mathbf{U} – is two times of the row number of the FE-mesh coordinates (ResultFEMesh{1}.coordinatesFEM), and \mathbf{U} vector is assembled by the nodal displacements $[u^1, v^1, u^2, v^2, \dots, u^N, v^N]^T$ with nodal indices as $[1, 2, \dots, N]$. The length

of \mathbf{F} vector is four times of the row number of FE-mesh coordinates and is assembled in the order of $[F_{11}^1, F_{21}^1, F_{12}^1, F_{22}^1, F_{11}^2, F_{21}^2, F_{12}^2, F_{22}^2, \dots, F_{11}^N, F_{21}^N, F_{12}^N, F_{22}^N]^T$.

(ii) With the FE-based global DIC solved displacements \mathbf{U} and deformation gradients \mathbf{F} , we calculate strains in code Section 5. For example, if strains are small ($< 5\%$), infinitesimal strain is a good option, where $e_{xx} = F_{11}$, $e_{yy} = F_{22}$, $e_{xy} = 0.5(F_{12} + F_{21})$. For engineering shear strains, $E_{xx} = F_{11}$, $E_{yy} = F_{22}$, $E_{xy} = (F_{12} + F_{21})$. With computed \mathbf{F} or $\mathbf{FStraintemp}$, other types of finite strain measurements can be easily computed based on user's choice.

(iii) Comment about difference between \mathbf{F} and $\mathbf{FStraintemp}$. In code Section 5, you will be also be asked:

```

1 What method to use to compute strain?
2 0: Finite element (Recommended);
3 1: Finite difference;
4 2: Plane fitting;
```

If you choose the method 0, $\mathbf{FStraintemp}$ is computed from the Gauss points of the finite elements; If you choose the method 1 or method 2, the $\mathbf{FStraintemp}$ is re-computed from the computed \mathbf{u} . The size of $\mathbf{FStraintemp}$ is cropped by \mathbf{Rad} , since the strains near the edges are less accurate. The coordinates of $\mathbf{FStraintemp}$ are:

```
x0(1+Rad:M-Rad,1+Rad:N-Rad), y0(1+Rad:M-Rad,1+Rad:N-Rad).
```

Acknowledgements

I am grateful to Dr. Louisa Avellar for sharing her unpublished images of fracture. I also want to thank Prof. Can C. Aydiner, Dr. Alex K. Landauer and Jialiang Tao, and all the ALDIC and ALDVC users for lots of helpful feedback and 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] J Yang and K Bhattacharya. Augmented Lagrangian Digital Image Correlation. *Experimental Mechanics*, 59:187–205, 2019.
- [2] J Yang and K Bhattacharya. Combining image compression with digital image correlation. *Experimental Mechanics*, 59:629–642, 2019.
- [3] *Augmented Lagrangian Digital Image Correlation (2D_ALDIC) Code*. <https://www.mathworks.com/matlabcentral/fileexchange/70499-augmented-lagrangian-digital-image-correlation-and-tracking>.
- [4] https://www.researchgate.net/publication/329456141_Augmented_Lagrangian_Digital_Image_Correlation.
- [5] *FE-Global-DIC code*. <https://www.mathworks.com/matlabcentral/fileexchange/82873-2d-finite-element-global-digital-image-correlation-fe-dic>.
- [6] *2D Image Interpolation with ba_interp2*. https://www.mathworks.com/matlabcentral/fileexchange/20342-image-interpolation-ba_interp2.
- [7] *MATLAB Support for MinGW-w64 C/C++ Compiler*. <https://www.mathworks.com/matlabcentral/fileexchange/52848-matlab-support-for-mingw-w64-c-c-compiler>.
- [8] *MathWorks: MinGW-w64 Compiler*. https://www.mathworks.com/help/matlab/matlab_external/install-mingw-support-package.html.
- [9] E Bar-Kochba, J Toyjanova, E Andrews, K-S Kim, and C Franck. A fast iterative digital volume correlation algorithm for large deformations. *Experimental Mechanics*, 55:261–274, 2015.
- [10] AK Landauer, M Patel, DL Henann, and C Franck. A q-factor-based digital image correlation algorithm (qDIC) for resolving finite deformations with degenerate speckle patterns. *Experimental Mechanics*, 58:815–830, 2018.
- [11] *FIDVC code*. <https://github.com/FranckLab/FIDVC>.
- [12] *qFIDVC code*. <https://github.com/FranckLab/qFIDVC>.
- [13] G Besnard, H Leclerc, F Hild, S Roux, and N Swiergiel. Analysis of image series through global digital image correlation. *The Journal of Strain Analysis for Engineering Design*, 47:214–228, 2012.

- [14] J Yang and K Bhattacharya. Fast Adaptive Mesh Augmented Lagrangian Digital Image Correlation. *Experimental Mechanics*, 61:719–735, 2021.
- [15] PC Hansen. The L-curve and its use in the numerical treatment of inverse problems. 1999.