Finite Element Global Digital Image Correlation (FE-Global-DIC) Code Manual (v1.0)

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 November 15, 2020

Contents

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], 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: [2]). 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 [3]).

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.

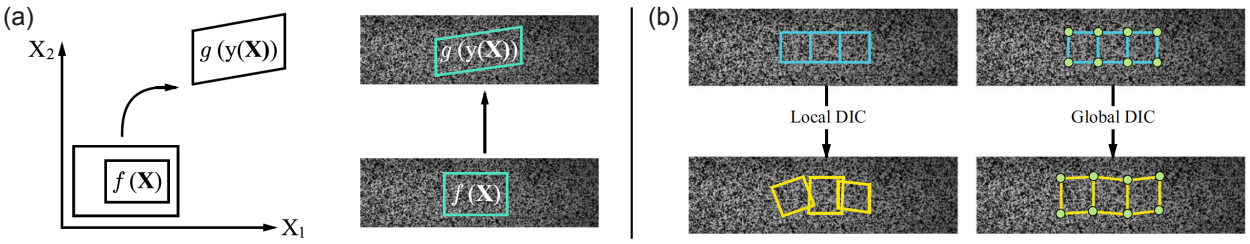


Figure 1: (a) Schematic showing a DIC reference image f(**X**), with a general speckle pattern, deforming into the deformed image g(**y**(**X**)) under some mapping **y**. **X** and **y** coordinates are in the reference and deformed images, respectively. (b) A schematic comparison between the local 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 [4]. 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.

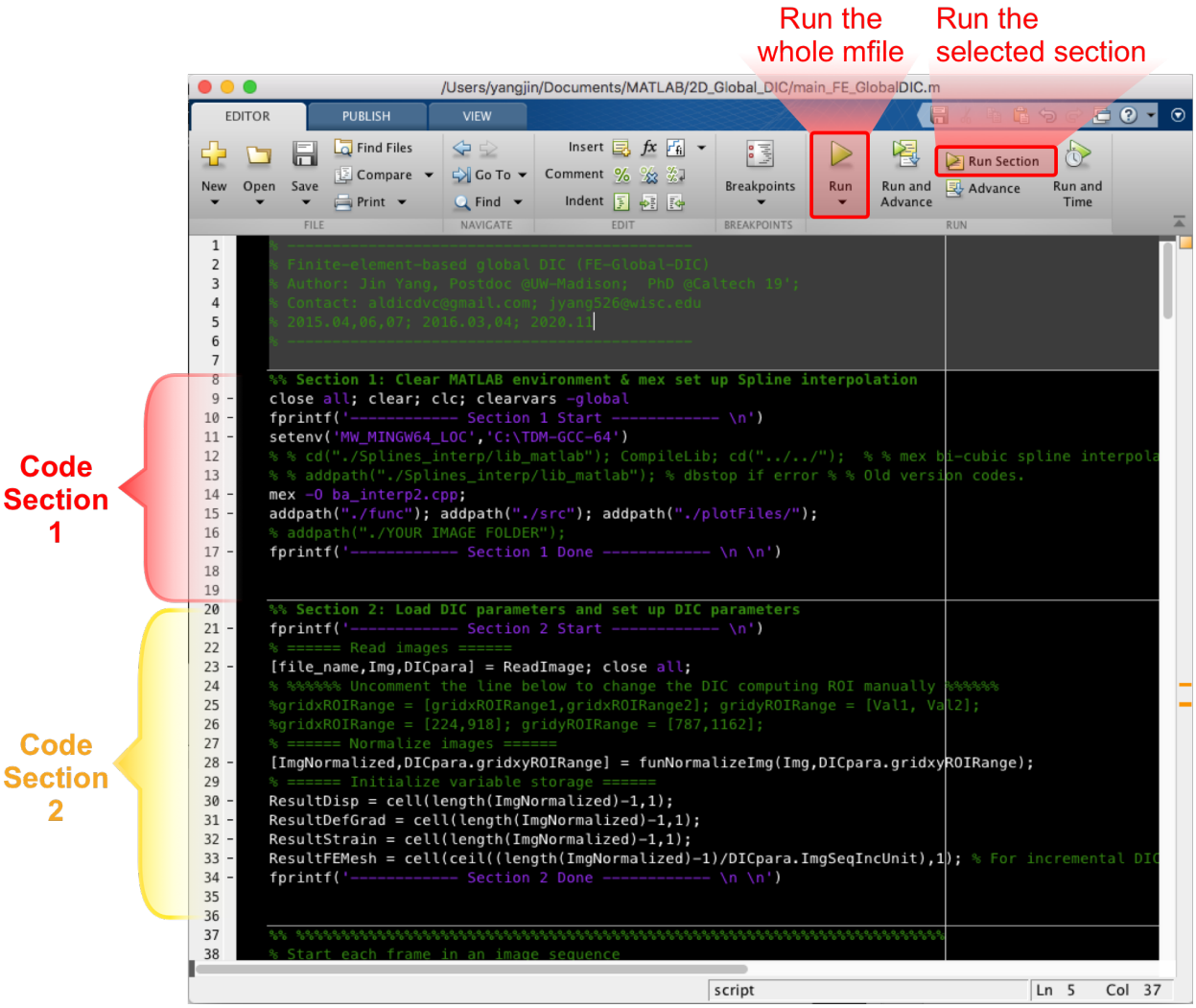


Figure 2: Main file of the FE-Global-DIC code: "main_FE_GlobalDIC.m". Each section can be executed in order by clicking the "Run Section" button.

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

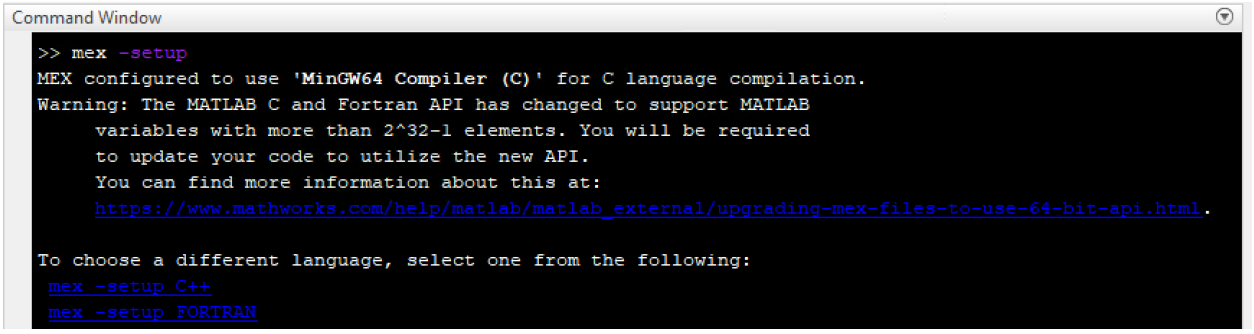


Figure 3: Message display on the command window when a mex C/C++ compiler is stalled successfully.

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" [5].

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.

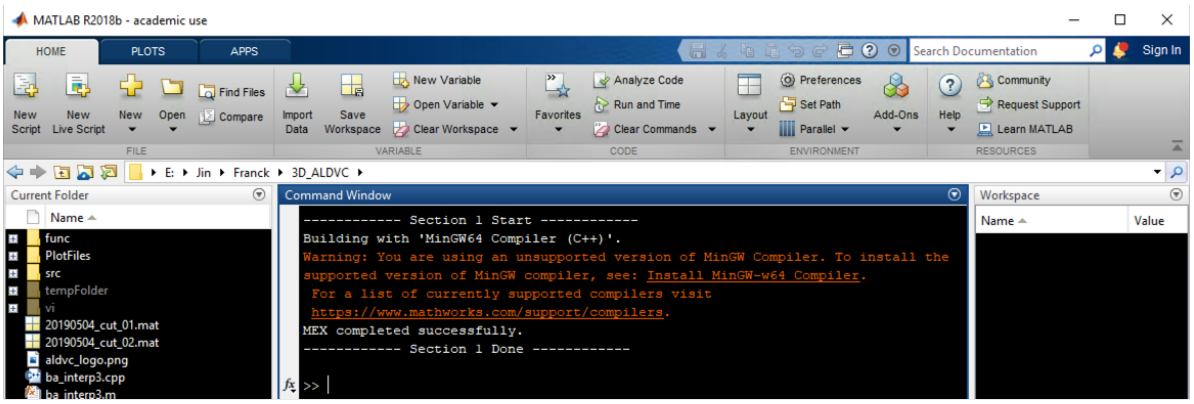
3.2 Install mex C/C++ compiler

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 [6, 7].

- 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.

[1] 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.

(a)



(b)

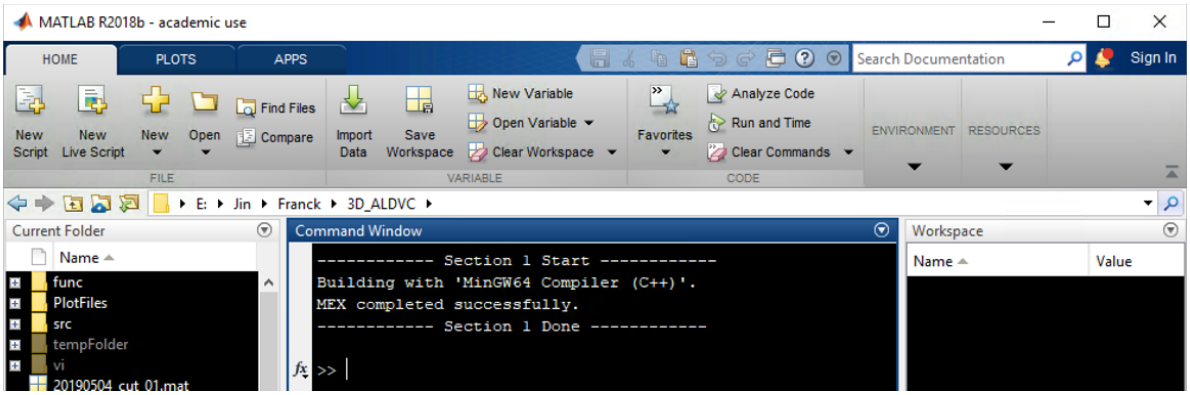


Figure 4: FE-Global-DIC code Section 1 is executed successfully and a message similar to (a) or (b) will display on the command window.

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.
ImgSeqIncROIUpdateOrNot	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.
NewFFTSearch	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.

```
------------ Section 2 Start ------------
Choose method to load images:
0: Select images folder;
1: Use prefix of image names;
2: Manually select images.
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", $\cdots$, use 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.

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

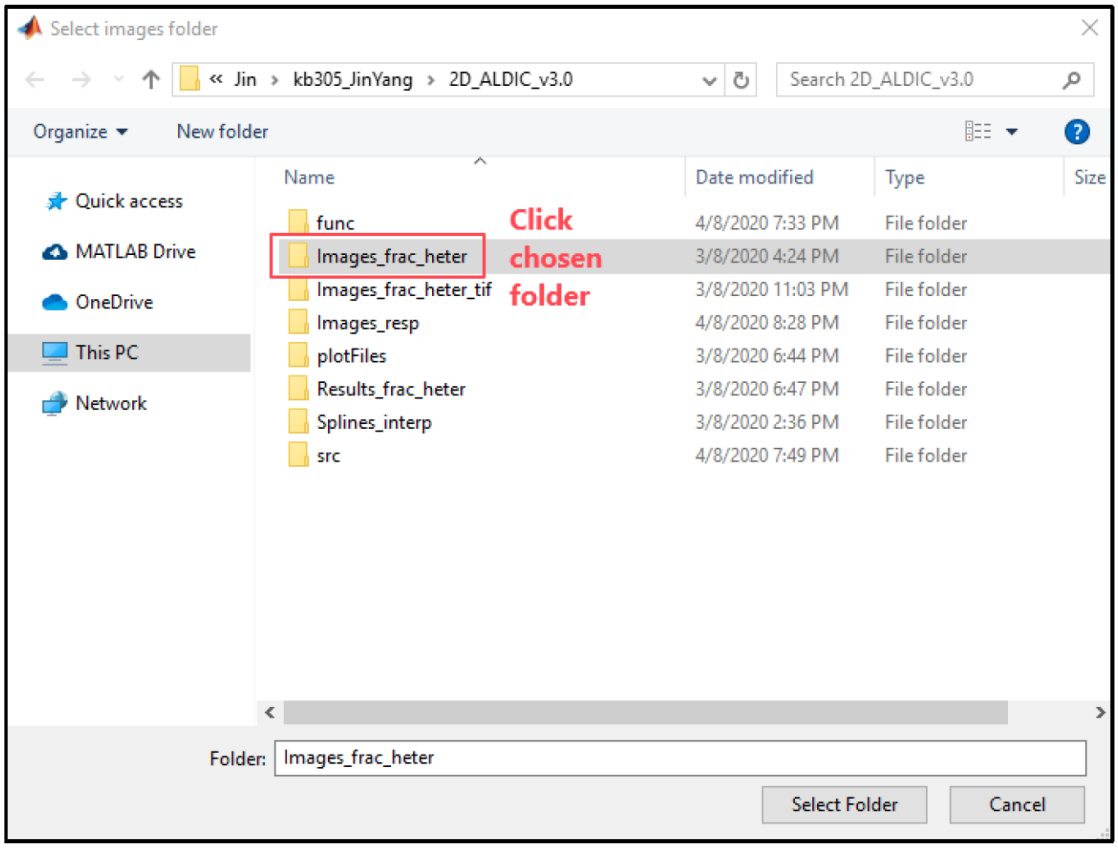


Figure 5: Load DIC iamges by selecting the folder including all DIC images. For example, after clicking chosen folder "Images_frac_heter", all the images in this folder will be loaded automatically.

other images as deformed images by default.

```
--- Please load first image ---
--- Please load next 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. 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:

```
--- 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.

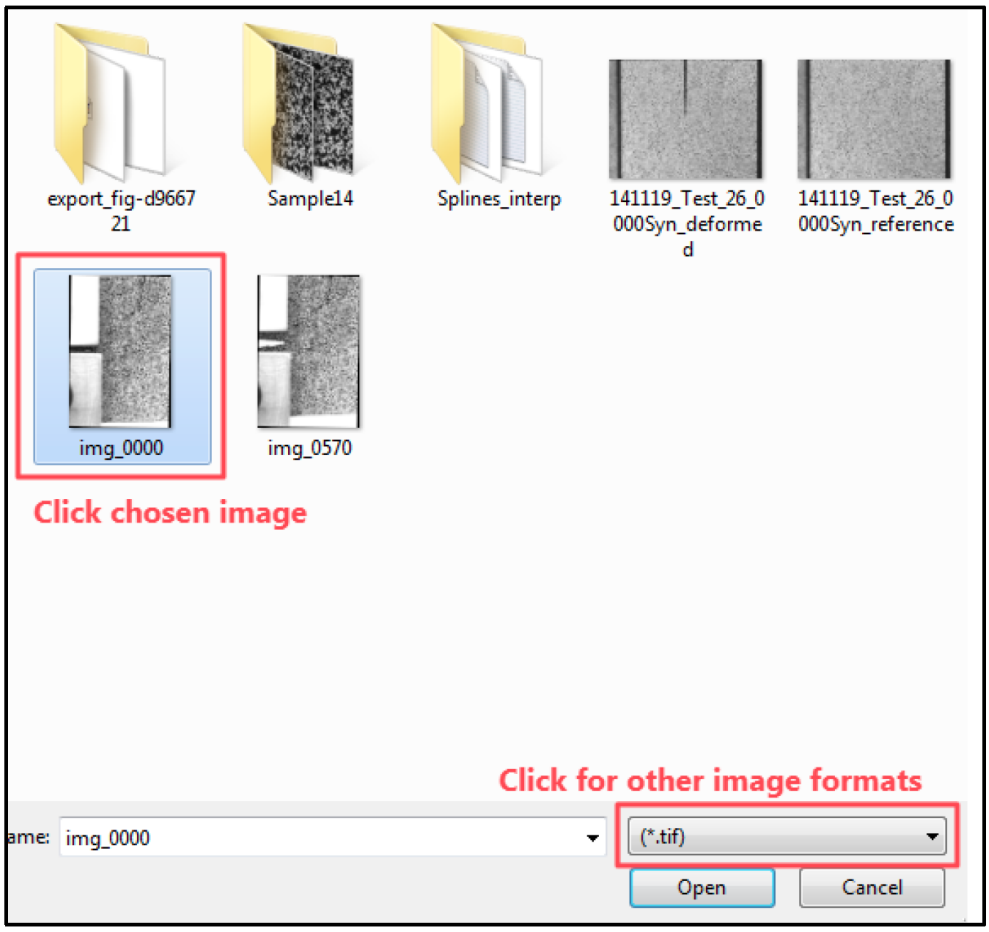


Figure 6: Manually load DIC images by clicking both reference and deformed images.

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:

```
Coordinates of top-left corner point are (322.786,74.730)
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.

```
--- What is the finite element size? (unit: px) ---
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.

```
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:
```

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:

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

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

```
Update ROI at the same time of updating reference image?
    0: Do not update ROI;
    1: Manually(Recommended);
    2: Automatically;
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: Compute initial guess from FFT-based cross correlation

5.1 FFT-based methods to compute initial guess

In this section, an initial guess of the unknown deformation is computed using fast Fourier transform (FFT) based method to maximize zero normalized cross correlation function C_{ZNCC}.

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

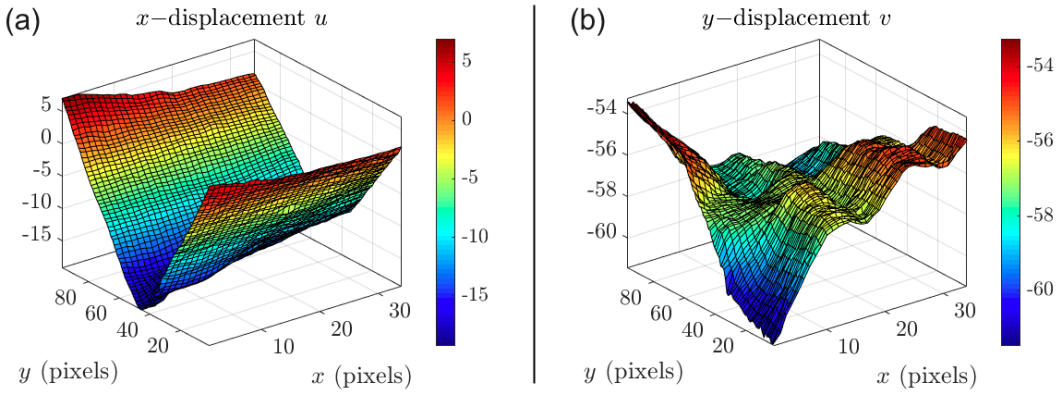


Figure 8: Solved initial guess for displacement fields by FFT-based method to maximize zero normalized cross correlation function.

where $\mathcal{F}(\cdot)$ denotes 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. $\mu_{\mathsf{f}}, \mu_{\mathsf{g}}$ are average grayscale values of DIC images f and g; $\sigma_{\mathsf{f}}, \sigma_{\mathsf{g}}$ are the standard deviation of image grayscale values.

To speed up the above FFT-based optimization process, we apply a multiscale scheme based on a constructed image pyramid. For example, an initial guess for above heterogeneous fracture example is solved in Fig. 8.

5.2 Remove noise in computed initial guess

In practice, FFT-based method solved initial guess[2] may have large noise, user can further remove these bad points by applying a median filter, setting a q-factor threshold and setting both upper and lower bounds of displacements. User can also continue to remove bad points by directly clicking them on each image frame and then press " `Enter` " key. However, this step needs some manual work. So this step is **not recommended** until it's very necessary for the overall quality improvement of final DIC results. If this noise removal is necessary, user could follow these steps and guide lines will display on MATLAB command window automatically:

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

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

```
% ======= Find bad initial guess points manually by setting bounds =======
What is your upper bound for x-displacement?
```

[2]Some designed filters or iterative deformation method (IDM) can further improve the accuracy of initial guess, cf [8, 9, 10, 11]. These are beyond the scope of this code manual. Additionally, accuracy of computed initial guess will further be improved after executing ALDIC ADMM iterations (code Sections 4-6).

```
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 clicking them directly.

```
% ======= 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.

At the end of this section, a finite element mesh will be generated automatically.

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

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

```
--- Start to compute image gradients ---
--- 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 [12], i.e.,

$$\mathbf{y}(\mathbf{X}) = \mathbf{X} + \mathbf{u}(\mathbf{X}) = \mathbf{X} + \sum_p u_p \psi_p(\mathbf{X}) \tag{2}$$

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,

$$C_g = \int_\Omega \left| f(\mathbf{X}) - g(\mathbf{X} + \sum_p u_p \psi_p(\mathbf{X})) \right|^2 d\mathbf{X} \tag{3}$$

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

$$g(\mathbf{y}(\mathbf{X})) = g(\mathbf{X} + \mathbf{u}_k(\mathbf{X}) + \delta\mathbf{u}) \approx g(\mathbf{X} + \mathbf{u}_k(\mathbf{X})) + \nabla g \cdot \delta\mathbf{u}(\mathbf{X}) \tag{4}$$

such that

$$C_g \approx \int_\Omega \left| f(\mathbf{X}) - g(\mathbf{X} + \mathbf{u}_k(\mathbf{X})) - \left(\sum_p \delta u_p \psi_p(\mathbf{X}) \right) \cdot \nabla g(\mathbf{X}) \right|^2 d\mathbf{X} \tag{5}$$

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

$$M_{pq}\delta u_q = b_p \tag{6}$$

where

$$M_{pq} = \int_\Omega \psi_p^T(\mathbf{X})(\nabla g(\mathbf{X}))(\nabla g(\mathbf{X}))^T\psi_q(\mathbf{X})d\mathbf{X} \tag{7}$$

$$b_p = \int_\Omega (f(\mathbf{X}) - g(\mathbf{X}+\mathbf{u}_k(\mathbf{X})))\psi_p^T(\mathbf{X})\nabla g(\mathbf{X})d\mathbf{X}. \tag{8}$$

Different from conventional finite element analysis where numerical integrals can computed using several Gauss points, here, we use pixel-wise summation to approximate numerical integrals in equations (7-8).

6.2 Regularization technique

To speed up the convergence and reduce the noise, we could add regularization penalties onto the DIC correlation function (3).

$$C_{g-regularized} = \int_\Omega |f(\mathbf{X}) - g(\mathbf{X}+\mathbf{u}(\mathbf{X}))|^2 d\mathbf{X} + \alpha|\nabla\mathbf{u}(\mathbf{X})|^2 \tag{9}$$

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 α = 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.

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

```
------------ Section 4 Start ------------
--- Global IC-GN iterations ---
normW = 0.41832   at iter 1;  time cost = 47.4258s
normW = 0.040996    at iter 2;    time cost = 38.4114s
normW = 0.014932    at iter 3;    time cost = 39.9557s
normW = 0.0067409   at iter 4;    time cost = 39.4569s
normW = 0.0032884   at iter 5;    time cost = 38.5084s
normW = 0.0016662   at iter 6;    time cost = 39.3641s
normW = 0.00086242  at iter 7;    time cost = 38.8785s
normW = 0.0004522   at iter 8;    time cost = 40.3063s
normW = 0.00023907  at iter 9;    time cost = 38.8822s
```

[3] There 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.

```
normW = 0.00012706   at iter 10;    time cost = 38.8817s
normW = 6.777e-05    at iter 11;    time cost = 39.1988s
Elapsed time is 439.2666 seconds.
------------ Section 4 Done ------------
```

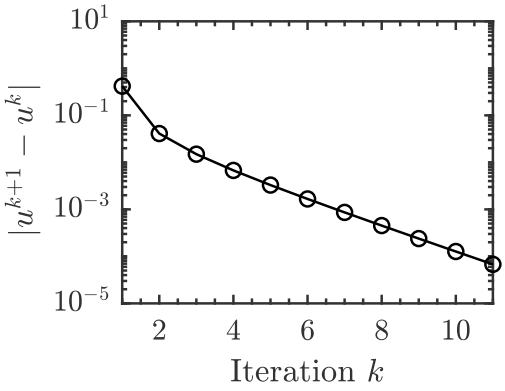


Figure 9: FE-Global-DIC converges after 11 iterations when α = 10.

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. 10(a)). A large value of α may oversmooth the deformation fields, which are preferred by the added regularization penalties (see Fig. 10(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 value of α from the modified L-curve method [13]. User needs to uncomment the following line, then the code will tune the best value of α automatically, see Fig. 11.

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

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,

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

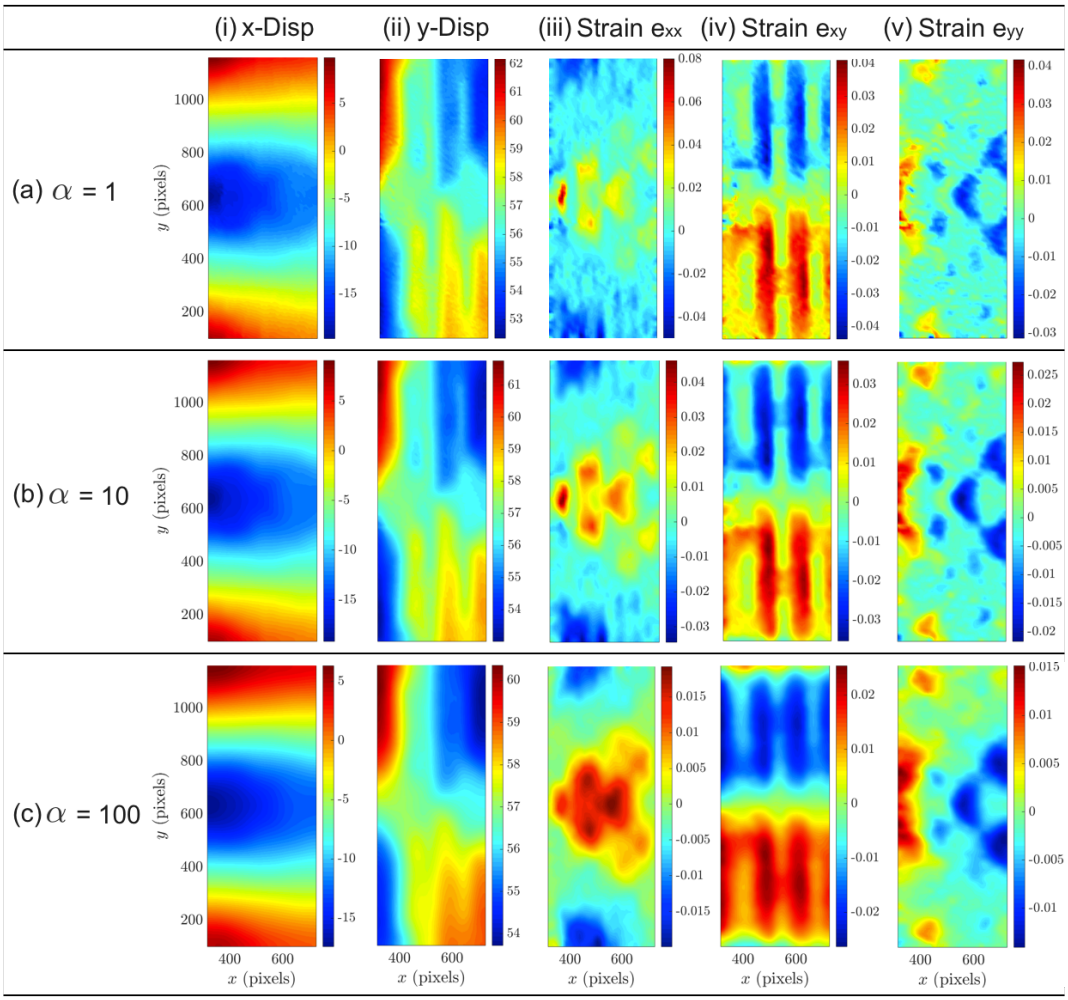


Figure 10: FE-Global-DIC method solved displacement and strain fields with different regularization coefficients, α. With a smaller value of α, the solved displacements and strains are noisier, while a large value of α may oversmooth the deformation fields.

so usually there is no need to further smooth solved displacement fields anymore.

```
------------ Section 5 Start ------------
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

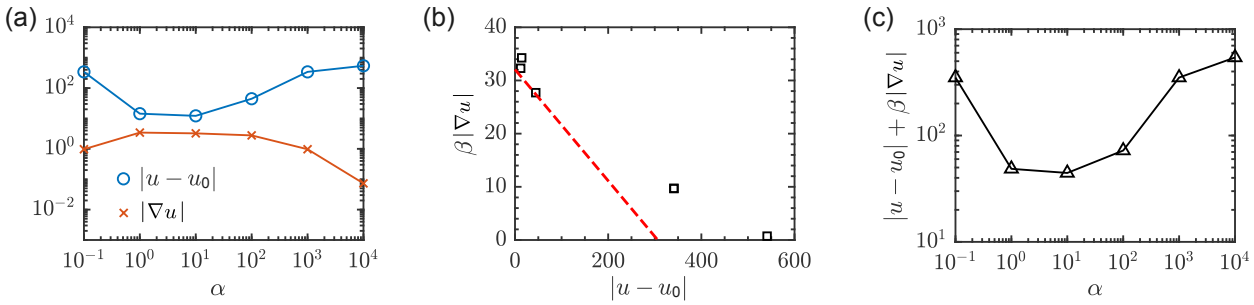


Figure 11: Finding the best coefficient of the regularization by the L-curve method. (a-b) Plots of terms $|\mathbf{u}-\mathbf{u}_0|$ and $|\nabla\mathbf{u}|$. Red dashed line in (b) points out the best α in the L-curve method, which is equivalent to the plot in (c) for the "$|\mathbf{u}-\mathbf{u}_0|+\beta|\nabla\mathbf{u}|$", where the minimizer corresponds to the best α. Here, β is an empirical number here I set as the finite element size (β = 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.

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;

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

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

```
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.

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

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" **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

7.3 Plot and save results

Users can visualize solved displacement and strain field and plot them overlaid with original DIC images, see Fig. 12. Finally, don't forget to save results for future use.

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

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

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

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

```
%% Save data again including strain solve method
results_name = ['results_FE_globalDIC_',imgname,'_st',num2str(DICpara.
    winstepsize),'_alpha',num2str(DICpara.alpha),'.mat'];
save(results_name, 'file_name','DICpara','DICmesh','ResultDisp','
    ResultDefGrad','ResultStrain','ResultFEMesh',...
      'normOfW','timeICGN');
```

8 Frequently Asked Questions (FAQs)

8.1 About MATLAB mex set up

[1] Where do I install TDM-gcc compiler?

This is a general question for MATLAB users to apply mex C/C++ compilers where more details can be found online, e.g., [6, 7]. User needs to install suitable mex C/C++ compiler based on

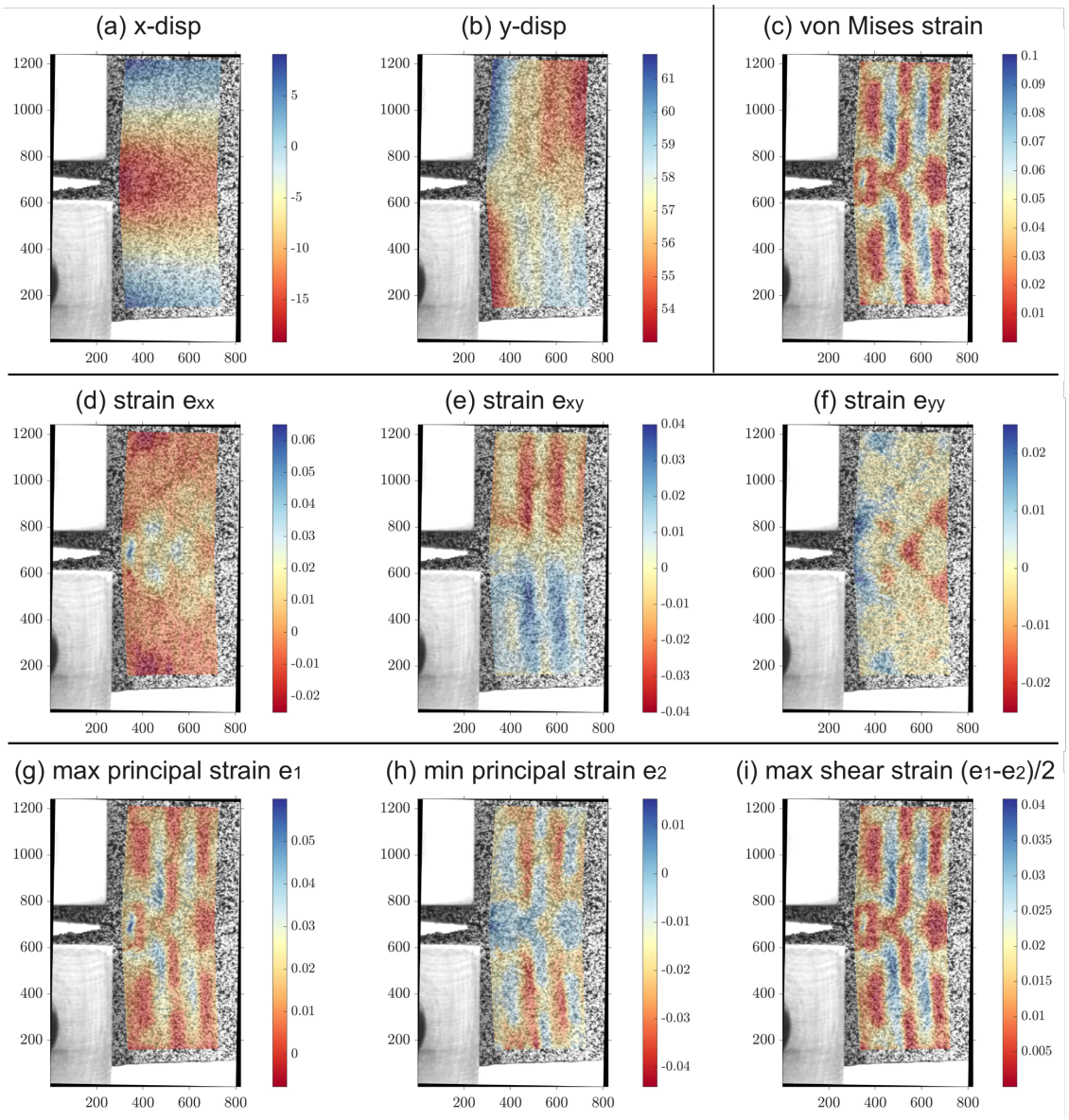


Figure 12: Final 2D FE-Global-DIC code tracked deformation fields with original DIC images overlaid as the background. (a) x-displacement; (b) y-displacement; (c) equivalent von Mises strain; (d-f) infinitesimal strains e_{xx}, e_{xy}, and e_{yy}; (g-h) maximum and minimum principal strains; (i) max shear strain.

their OS and MATLAB versions. Based on my experience, mex C/C++ compiler usually is already installed on MAC system. For Windows users, I usually install the TDM-gcc compiler to the first level main disks like "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.

[2] What if I don't want to use mex compiler or "ba_interp2"? Can I still use the code?

Sure! You can still use this MATLAB code! You need to convert the "ba_interp2" to other MATLAB interpolation functions. This change needs to be done in the "./func/funGlobalICGN.m" code line# 102 to change the "interpmethod". For example, you can replace the function "ba_interp2" with "interp2", where interp2 is the MATLAB built interpolation function. You can also try other interpolation schemes too.

```
tempg = ba_interp2(Img2, ptOfyAll+tempVMat, ptOfxAll+tempUMat, 'cubic'); %
    Deformed g(x+u)
```

Based on my experience, "ba_interp2" runs much faster than the MATLAB built cubic interpolation script. So if "ba_interp2" still can be used, maybe that will be the best solution.

8.2 About computing strains

[3] How to compute other types of strains?

(i) FE-Global-DIC solved 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 coordinates in deformed and reference configurations, are stored in `ResultDisp` and `ResultDefGrad` , respectively.

In 2D DIC, the length of $\mathbf{U}$ vector is two times of row # of coordinates (in ResultFEMesh1.coordinatesFEM), and the $\mathbf{U}$ vector is assembled by the nodal displacements $[u^1, v^1, u^2, v^2, \cdots, u^N, v^N]^T$ of nodes $[1, 2, \cdots, N]$. The length of $\mathbf{F}$ vector is four times of row # of coordinates and assembled in the order of $[F_{11}^1, F_{21}^1, F_{12}^1, F_{22}^1, F_{11}^2, F_{21}^2, F_{12}^2, F_{22}^2, \cdots, F_{11}^N, F_{21}^N, F_{12}^N, F_{22}^N]^T$.

(ii) With these solved displacements $\mathbf{U}$ and deformation gradients $\mathbf{F}$, we compute strains in code Section 8. 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 `FStraintemp` , other types of finite strain measurements can be easily computed based on user's choice.

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

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

If you choose method 0, `FStraintemp` is exactly the same with computed $\mathbf{F}$; If you choose method 1-2, the `FStraintemp` is re-computed from the computed u. The size of `FStraintemp` is cropped by "Rad" since the strains near the edges are less accurate. The coordinates of `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. Aydıner, 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] *2D ALDIC code*. https://www.mathworks.com/matlabcentral/fileexchange/70499-augmented-lagrangian-digital-image-correlation-and-tracking.

[3] https://www.researchgate.net/publication/329456141_Augmented_Lagrangian_Digital_Image_Correlation.

[4] *FE-Global-DIC code*. https://www.mathworks.com/matlabcentral/fileexchange/82873-2d-finite-element-global-digital-image-correlation-fe-dic.

[5] *2D Image Interpolation with ba_interp2*. https://www.mathworks.com/matlabcentral/fileexchange/20342-image-interpolation-ba_interp2.

[6] *MATLAB Support for MinGW-w64 C/C++ Compiler*. https://www.mathworks.com/matlabcentral/fileexchange/52848-matlab-support-for-mingw-w64-c-c-compiler.

[7] *MathWorks: MinGW-w64 Compiler*. https://www.mathworks.com/help/matlab/matlab_external/install-mingw-support-package.html.

[8] 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.

[9] 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.

[10] *FIDVC code*. https://github.com/FranckLab/FIDVC.

[11] *qFIDVC code*. https://github.com/FranckLab/qFIDVC.

[12] 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.

[13] PC Hansen. The L-curve and its use in the numerical treatment of inverse problems. 1999.