GridStrain

Release 2.0 – November 2022

User's guide

Giordano Teza
Department of Physics and Astronomy, Alma Mater Studiorum University of Bologna
Viale Berti Pichat, 6/2, I-40127 Bologna, Italy
giordano.teza@unibo.it, giordano.teza@gmail.com

Arianna Pesci
Istituto Nazionale di Geofisica e Vulcanologia, Bologna, Italy
Via Creti, 1, I-40128 Bologna, Italy
arianna.pesci@bo.ingv.it

Table of contents

		page
1	Introduction …….…….	3
2	Toolbox installation and program running ……..…….	5
	***2D version* (`GridStrain and StrainZero`)**	
3	Options ……..	6
	3.1 `GridStrain` saved general options ……..……..……	6
	3.2 Initial options, data importation, visualization and management ……..	7
4	Grid creation and/or modification ……...…….....	14
5	Definition and/or change of the scale factor …….....	16
6	Strain computation on the grid nodes and significance evaluation …….....	17
7	Data visualization ….……..……..…….....	20
8	Complementary functions and options: ……...……..……..	27
	8.1 Generation of a background image from a DTM ……...	27
	8.2 Data export to ASCII ESRI `.asc` format …….	28
9	Strain rate computation without a grid: `StrainZero` ……	30
	***3D version* (`grid_strain3`)**	
10	Data importation in the 3D case ……..…….	34
11	Strain computation in the 3D case and visualizations ……..……	38
11	References …….	40

1. Introduction

`GridStrain` is the upgraded version of `grid_strain` MATLAB™ toolbox described in and Pesci and Teza (2007), which is conceived for easy and quick calculation of the 2D deformation field (or the 2D deformation rate field) in a region, starting from the displacements (or the velocities) of a series of observational points adequately distributed in this region. The obtained deformation pattern is shown as a set of the principal components of the strain (strain rate) calculated on the nodes of a regular grid whose limits and steps can be chosen by the user. See Teza et al., (2008) and Pesci and Teza (2007) for a general description of the implemented approach and some examples of its application on GPS-based, regional scale analysis of a strain field. The computations are performed in a modified least square approach (MLS), as described in Shen et al. (1996). The upgraded version is described in Teza et al. (2022) together with an application.

The main changes with respect to the previous releases (i.e. the various versions of grid_strain toolbox) are:

- The introduction of a struct variable `GSS`, which can be updated as different `GridStrain` stages are executed, whose fields are all the data and results of computations. Although `GSS` is used like an object, it is not an object and, therefore, no methods are associated to it;
- Upgraded interactive interfaces. Menu boxes are removed and replaced by user interfaces and dialog boxes;
- Useless options are removed;
- Upgraded visualization functions;
- Upgraded data saving options. In particular, if ASCII data are to be exported, ArcGIS-like `.asc` files can be saved.

The 3D version of `GridStrain/grid_strain` is `grid_strain3`. The calculations are performed in this case on the points of a digital terrain model (DTM). See Teza *et al.* (2008a) for further information. The results of an application of both toolboxes to two landslides are presented in Teza *et al.* (2008b).

The toolboxes, which are sets of functions called by the main programs (`GridStrain.m` and `grid_strain3.m` respectively), also provide significance evaluations. This because a result related to a grid node (a DTM point in the 3D case) can be really used only if the observational points are well distributed around it. In particular, three levels of significance are defined: high, mid and low, according to the user's choice (see below).

The typical and simplest sequence to follow is:

1) Data logging, visualization and management;
2) Grid creation and/or modification;
3) Definition and/or change of the smoothing factor *d0*;
4) For each grid point, evaluation of:
   - strain rate tensor;
   - trace of the strain rate tensor, leading to rate of change in area and rate of normalized shear;
   - second invariant of the train;
   - geometric significance of the results;
5) Data visualization.

The function `StrainZero` (computation of 2D strain on a point without grid) is included in `GridStrain` toolbox.

These toolboxes are free. The authors require that the use of this software be intended for scientific use only (no commercial use). If a publication whose results were obtained by means of this software is accepted for the publication, the toolboxes `GridStrain/grid_strain3` and their authors must be cited.

Disclaimer: The authors of these toolboxes accept no responsibility for damages resulting from the use of these products and makes no warranty or representation, either express or implied, including but not limited to, any implied warranty of merchantability or fitness for a particular purpose. This software is provided "AS IS", and the user assumes all risks when using it.

The toolboxes surely run under MATLAB™ 2018a or later releases (Some actions aimed at allowing use with much older versions of MATLAB were however implemented. For example, whether a figure is an object or not is automatically verified). For any question, or also suggestion, please contact the authors (the Email addresses are shown in the cover of this user's guide).

2. Toolbox installation and program running

The toolbox `GridStrain` is contained in zip file `GridStrain.zip`. The files should be extracted and saved in a MATLAB directory whose name should be `GridStrain`. For example, if the MATLAB work directory is:

`C:\Users\john.doe\Documents\MATLAB\`,

the toolbox can be saved and accessed with this path:

`C:\Users\john.doe\Documents\MATLAB\GridStrain .`

Please note that, if MATLAB operates under an Unix environment, "/"must be used instead of "\".
If the directory `GridStrain` is saved in the default directory on which MATLAB command window operates, the directory change must carried out before the toolbox run. This change is obtained simply typing on the MATLAB command window (MCW):

```
cd GridStrain
```

The program runs by typing

```
GridStrain
```

on this MCW. All the necessary functions are automatically called by the principal program. The toolbox runs on different platforms (Windows, Linux).

The `grid_strain3` toolbox is contained in the directory `grid_strain3`, compressed in the zip file `grid_strain3.zip`.

In order to call the program whichever is the current MCW directory, a startup file can be written by the user. For example, if the `GridStrain` toolbox is placed in the directory '`C:\Users\john.doe\Documents\MATLAB\GridStrain`', `grid_strain3` is placed in the corresponding directory and Windows is the operating system, the rows

```
addpath 'C:\Users\john.doe\Documents\MATLAB\GridStrain;
addpath 'C:\Users\john.doe\Documents\MATLAB\grid_strain3'
```

should be added to the `startup.m` script. If a file named `startup.m` is placed in the MATLAB work directory, it is automatically executed at each MATLAB start. In this way, all the defined search paths are automatically added at each MATLAB start and the directory changes to use `GridStrain` and `grid_strain3` are unnecessary. The functionalities of these toolboxes do not depend on user's choice about the startup. More information about `addpath` function in a startup file can be found in http://www.mathworks.it/it/help/matlab/ref/ addpath.html.

Another option is the use of the Set Path dialog box, which appears by typing

```
pathtool
```

on the MCW or by selecting `Set Path` in `Home menu` of MATLAB desktop. The button `Add Folders` allows the choice of the folder and other buttons allow the choice of the folder order for the search of the files. Please see http://www.mathworks.it/it/help/matlab/matlab_env/using-the-matlab-search-path.html to have more information about the `Set Path` dialog box.

All provided functions are MATLAB `.m` files that can the opened and, if necessary, modified by the user. In this way, an expert user can modify, for example, the data saving options.

3. Options

3.1 `GridStrain` general saved options

The `GridStrain` general saved options are managed by means of `GridStrainOptions` function.
Two possible syntaxes are allowed:

```
GridStrainOptions
Options=GridStrainOptions
```

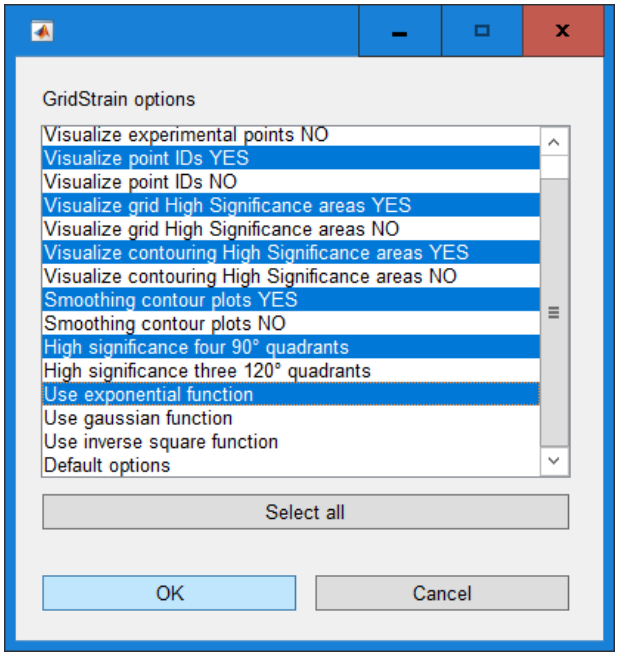

Figure 3.1 Interactive management of `GridStrain` options

If no output is considered, a modal box (Fig. 3.1) allows the management of main options related to `GridStrain` and save the file `GSOptions.mat`.
If an output is considered, no modal box is shown. In this case, if the file `GSOptions.mat` exists, the options related to this file are used. If the file `GSOptions.mat` does not exist, the default choices are used.
Options:

Option	Possible values	Default value
`Options.VisualizePoints`	`true / false`	`true`
`Options.VisualizePointIDs`	`true / false`	`true`
`Options.VisualizeGridHSA`	`true / false`	`true`
`Options.VisualizeContourHSA`	`true / false`	`true`
`Options.Function`	`'exp'/'gaussian'` `'inverse square'`	`'exp'`

where

`Options.VisualizePoints` is referred to the visualization of experimental points in strain plotting (i.e. this option does not apply in plotting for data loading and possible selection and removal of some stations);

`Options.VisualizePointIDs` is referred to the visualization of point IDs (numeric or station name, depending on input file) in both dataload and strain plotting (in the case of strain plotting, this option acts under the condition that `Options.VisualizePoints` is true);

`Options.VisualizeGridHSA` is referred to the grid visualization in case of plotting in high significance areas (or high and mean significance areas);

`Options.VisualizeContourHSA` is referred to the visualization of contour plots of strain in in high significance areas (or high and mean significance areas);

`Options.Function` is referred to the choice of the function, i.e. `'exp'` for $\exp(-d/d0)$, `'gaussian'` for $\exp(-(d/d0)^2)$ and `'inverse square'` for $1/[1+(d/d0)^2]$.

If the user interactively selects two incompatible choices (e.g. both true and false for `VisualizePoints`), the corresponding default value is used. In the no-choice case, all default values are used.

3.2 Initial options, data importation, visualization and management

When the program starts an interactive window appears and the user can import a new data file or use previously saved data.

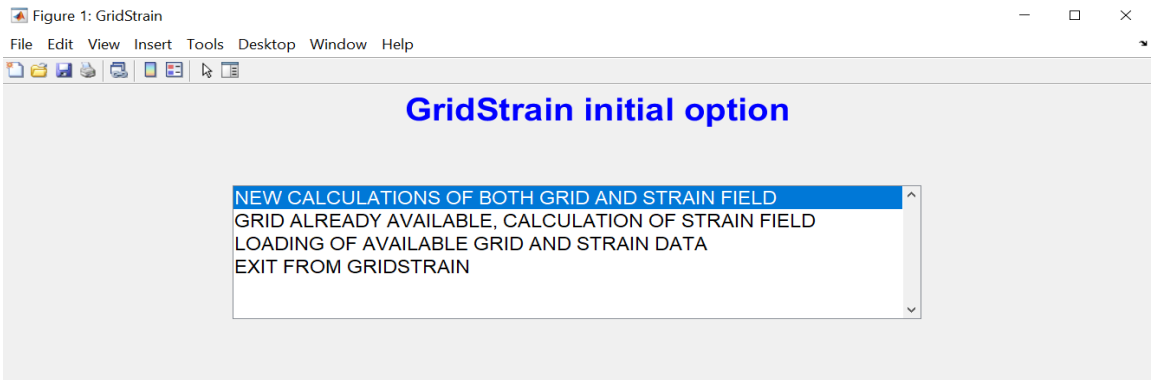

Figure 3.2. Top: the first menu interface. Bottom: dialog box for ASCII filename choice.

After the choice of the initial option, if an option different from "`EXIT FROM GRIDSTRAIN`" is chosen, a second interactive window allows the choice of the general saving option (Fig. 3.3). In this way, the user can choose one of the following options:

- Automatic data saving with a defined name of the generated files. In this case, the filenames are automatically generated. If, for example, `CommonPart` is the string chosen by means of a input dialog box which appears if the selected option is "`AUTOMATIC DATA SAVING WITH A DEFINED NAME`", the automatically generated files are `CommonPart2DG.mat` (grid data) and `CommonPart2DSFxxx.mat`, where xxx is the chosen scale factor (e.g. 10000 if the scale factor is 10000 m, i.e. 10 km).
- Data saving with filenames interactively managed.
- No saving. In this case, the saving function is always excluded.

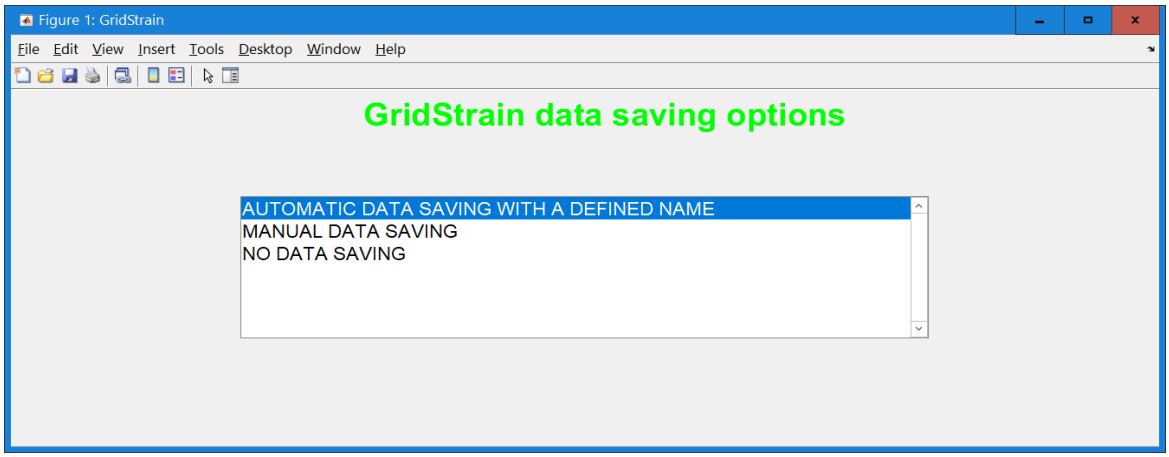

Figure 3.3. Data saving options.

After the choice of the data saving option, if the chosen initial option is "`NEW CALCULATION OF BOTH GRID AND STRAIN FIELD`", a combo box allows the choice of the input file name (Fig. 3.4), that must have ASCII format. The ASCII file `tutorialData2.txt` is added to the toolbox for tutorial purposes.

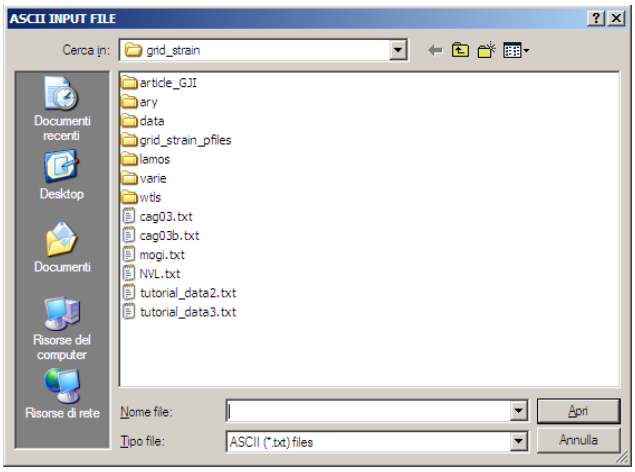

Figure 3.4 Interactive choice of the input ASCII file

There are some options for the input file. It can be a 10-column numerical matrix. The columns represent the data (with this order):

1	*ns*	station identifier (an integer number)
2	*e*	east coordinate, expressed in m
3	*n*	north coordinate, m
4	*ve*	east velocity (or east displacement) expressed in mm/y (or mm)
5	*vn*	north velocity (or north displacement) mm/y (or mm)
6	*eve*	root mean square (rms) error on *ve*, mm/y (or mm)
7	*evn*	rms error on *vn*, mm/y (or mm)
8	*a*	length of major half-axis of error ellipse, mm/y (or mm), *optional*
9	*b*	length of minor half-axis of error ellipse, mm/y (or mm), *optional*
10	*theta*	azimuth of major axis of error ellipse, degrees, *optional*

The last three columns are optional. If the file contains 7 columns only, the error ellipses are assumed to have the axes parallel to east and north direction respectively, with half-axes length simply expressed by *eve* and *evn*. If the number of columns is smaller than 7, the data load cannot

be performed. In this case, the choice of other filenames is possible. Although the coordinates are conceived as expressed in m and the displacements in mm, a check is performed in order to detect if the displacements are expressed in m instead. If this fact is suspected, a menu box in shown to allow the confirmation or exclusion of this suspect (the user is always guided in each choice).

Another option on the input ASCII file is:

1	*statName*	station name (four characters string)
2	*e*	east coordinate, expressed in m
3	*n*	north coordinate, m
4	*ve*	east velocity (or east displacement) expressed in mm/y (or mm)
5	*vn*	north velocity (or north displacement) mm/y (or mm)
6	*eve*	rms error on *ve*, mm/y (or mm)
7	*evn*	rms error on *vn*, mm/y (or mm)
8	*a*	length of major half-axis of error ellipse, mm/y (or mm), *optional*
9	*b*	length of minor half-axis of error ellipse, mm/y (or mm), *optional*
10	*theta*	azimuth of major axis of error ellipse, degrees, *optional*

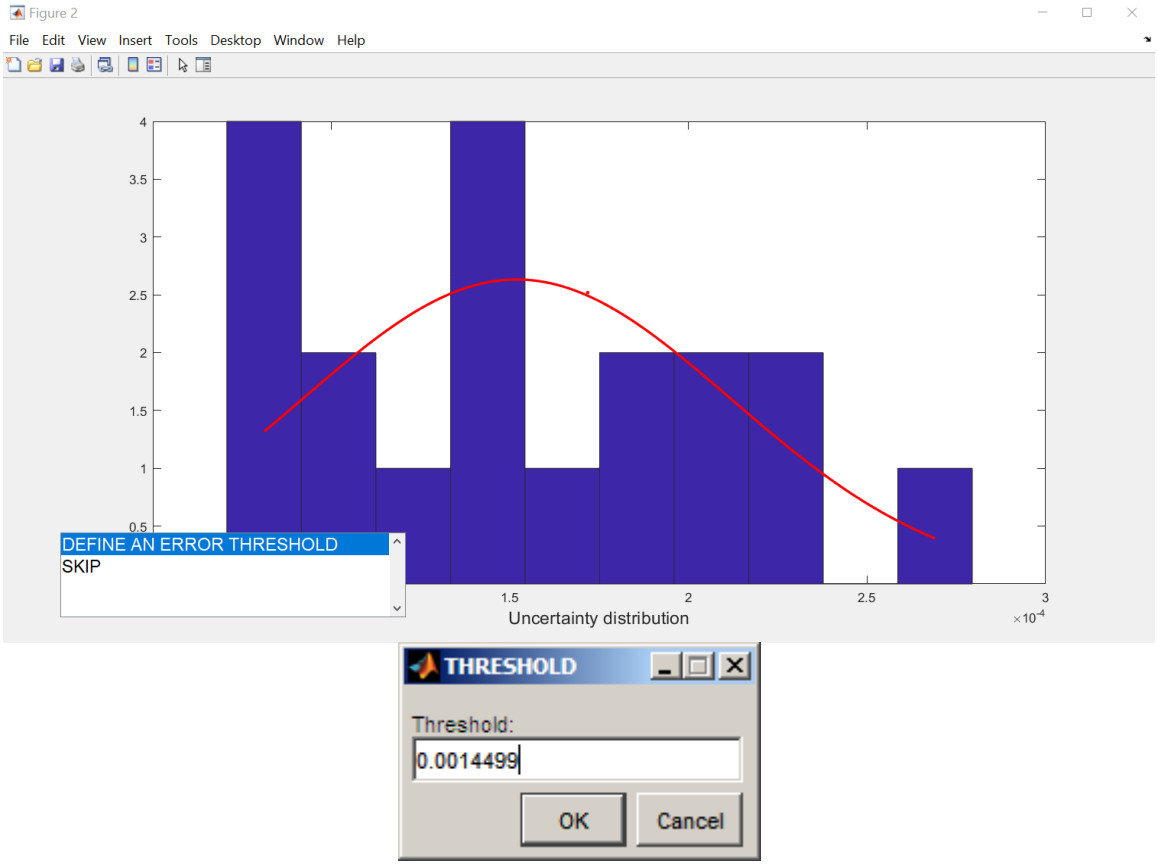

Figure 3.5 Upper panel: distribution of the input displacements (or velocity) uncertainties, shown for a possible error-based thresholding. The Gaussian model of the distribution is also shown (obviously, in the specific case such a model is inadequate because a strong tail exists. Each result provided by `GridStrain` requires an interpretation. No abilities in Computer Science are necessary, but a good knowledge of the studied phenomenon, as well as of basic statistics, is required). Lower panel: threshold definition (this dialog box appears if "`DEFINE AN ERROR THRESHOLD`" is selected)

Obviously, also in this case the columns can be 7 instead of 10. As the input file is loaded, the distribution of displacements (or velocity) errors is shown (Fig. 3.5) together with a user interface

(UI) control for the choice of an optional error threshold (Fig. 3.5, top). In this way, the user can define a lower threshold for the uncertainties of the experimental points (EPs) by acting on an input dialog box (Fig. 3.5, down). This threshold is referred to the squared sum of the *x*- and *y*-component errors. In particular, if the user defines a threshold, for the EPs whose squared sum of the displacement (or velocity) uncertainties are lower than this threshold, the uncertainties for both the components are now equal to such a threshold. The uncertainties related to the EPs whose squared sum of the *x*- and *y*-component errors are higher than the threshold are the initial values instead. Finally, a menu box allows the conformation of the threshold, the choice of a different threshold, or also the exclusion of the threshold. The corresponding Gaussian distribution of probability density is also shown. It should be noted the fact that, if large part of the stations have very similar uncertainties, the distribution can be strongly peaked with a low-significance tail.

The next step is the visualization of the displacement (or velocity) vectors, which are plotted together with their error ellipses (Fig. 3.6). If the optional error thresholding was carried out (see above), the error ellipses are the ones defined by such a thresholding. The user can introduce a background image by acting on a UI control. In order to avoid license issues related to Google Earth, the background image should be already available and manageable in this way:

- The image is stored as a JPEG or a TIFF file with a filename like, e.g., `'MyBGImage.jpg'`;
- a MATLAB .mat file having the name `'MyBGImage.mat'`, i.e. the same name of the image file but extension `.mat`, having a variable `ALI` whose value is the vector `[Xdatamin, Xdatamax, Ydatamin, Ydatamax]` with the coordinates of the vertices of the area represented by the image, expressed in the same reference frame of the input velocity data.

If a background image is used, the axis limits are the ones of the input velocities, regardless to the limits related to the field `ALI` (see Figures 3.7 and 3.8). The filename of the input MATLAB file or of the image file can be interactively managed in the standard way by means of a combo box (in the second case, the name of the file carrying the struct variable must be the same of the image file, clearly unless file extensions).

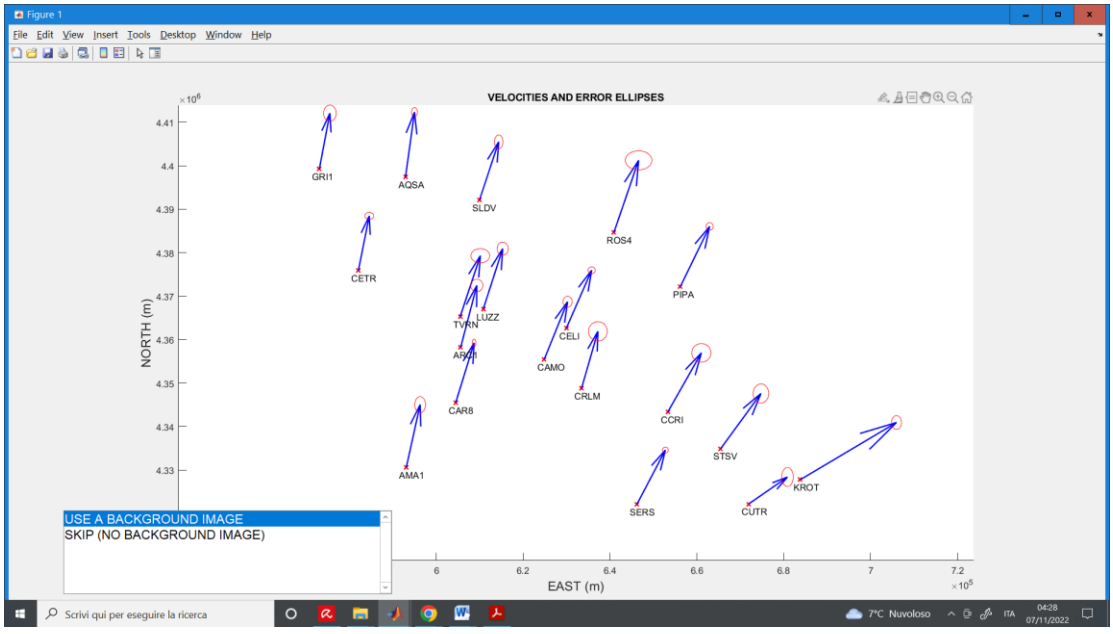

Figure 3.6 Velocity field and possible choice of a background image

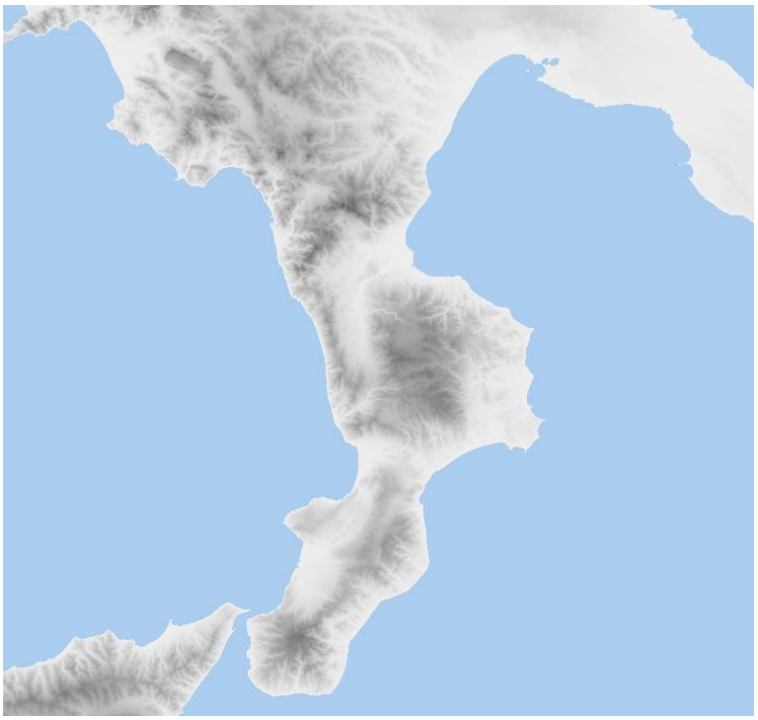

Figure 3.7 Example of background image, whose area is significantly higher than the one related to the velocity vectors (please see Figs. 3.6 and 3.8)

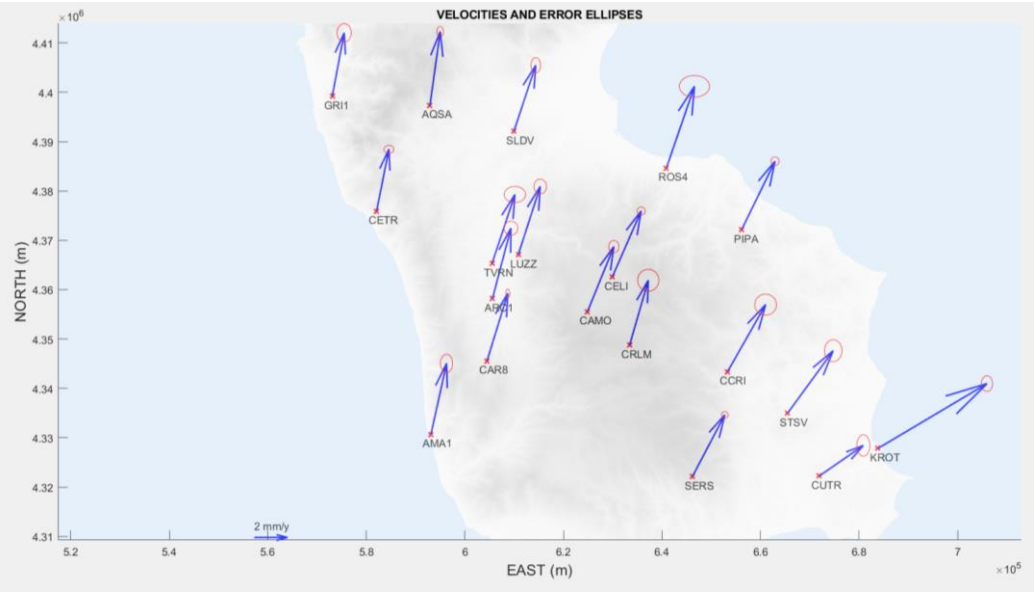

Figure 3.8 Velocities (with corresponding error ellipses) drawn on the background image

A complementary function allows the direct generation of a background image (included the struct variable above described) starting from an ASCII `.asc` digital elevation model (DEM) in ESRI format exported from a GIS software (please see the description of `gs_refDTMImage`, Chapter 8).

The input data are managed by means of `gs_dataload` function, which is automatically called by `GridStrain`. The function can be used autonomously. The valid syntaxes are

```
GSS=gs_dataload
GSS=gs_dataload(filena)
```

where `filena` is the input ASCII file. If `filena` is undefined or empty, the filename can be managed in an interactive way. Please type `help gs_dataload` on MCW for more information.

Regardless to the fact that `gs_dataload` is called by `GridStrain` or is used in autonomous way, the struct variable `GSS` is generated. The generation of `GSS` variable is the main innovation with respect to the `grid_strain` releases. In this first `GridStrain` stage, the `GSS` fields are:

`StationID` (empty if `statName` is the first column in the input file)
`StationName` (empty if `ns` is the first column in the input file)
`East`
`North`
`EastVelocity`
`NorthVelocity`
`EastVelocityError`
`NorthVelocityError`
`ErrorEllipsea` (if the input file has 7 columns, it is equal to `EastVelocityError`)
`ErrorEllipseb` (if the input file has 7 columns, it is equal to `NorthVelocityError`)
`ErrorEllipsetheta` (if the input file has 7 columns, it is a vector of zeros)
`RefImageStruct` (empty if no a background image is used)

The last field, i.e. `RefImageStruct`, if non-empty, is a struct variable with two subfields, i.e. `RefImageStruct.Image` (name of the background image file, including path) and `RefImageStruct.Vertices` (name of the vertices file, including path, see above).

Note that the output velocities and related uncertainties are always expressed in m/y to allow their direct use in strain computation, regardless to the unit used (m/y or mm/y) in the input ASCII file, and this despite the fact that the preferred input velocities are in mm/y. Clearly, if displacements are used, the `GSS` output data are expressed is mm.

Regardless to the presence of the optional background image, the next step is the optional removal of one or more EPs. An interactive image allows the choice between "`EXCLUSION OF ONE OR MORE POINTS FROM CALCULATIONS`" and "`NO POINT EXCLUSION`" (Fig. 3.9). If the first option is chosen, the point to be removed can be chosen by means of mouse (the EP closest to the selected position on the screen is the selected one). The selected EP is highlighted in a figure and an UI allows the selection confirmation (Fig. 3.10). This operation can be repeated more times, until at least two data points are available.

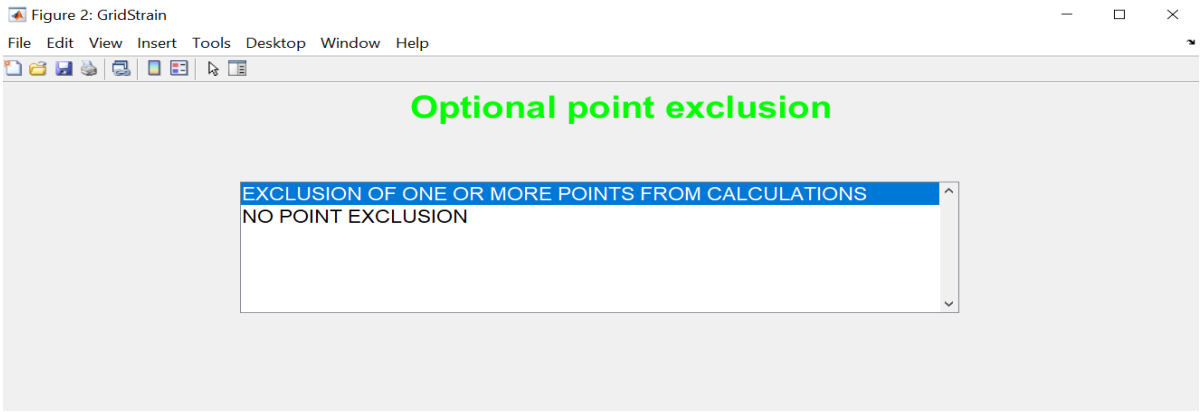

Figure 3.9 Optional point exclusion

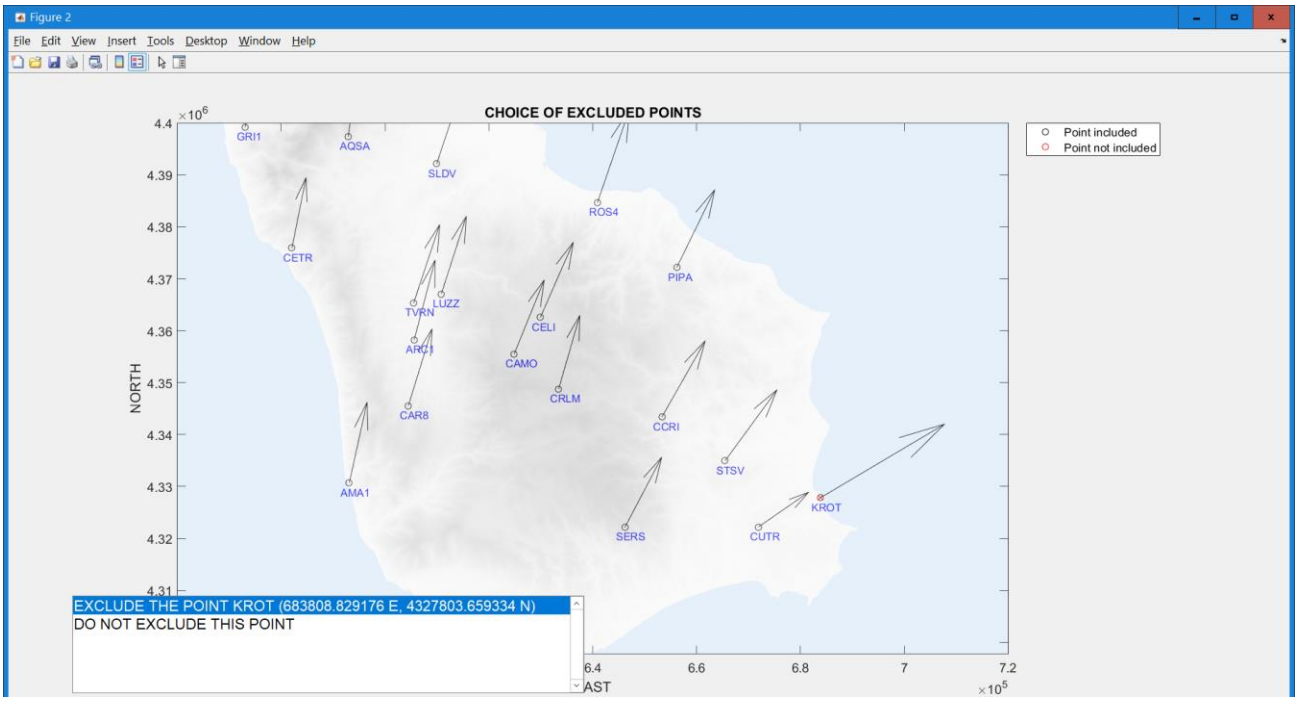

Figure 3.10 Details on selected point and removal confirmation.

The optional exclusion of one or more EPs is carried out by means of `gs_nopoints` function, which is automatically called by `GridStrain`. This function can also be used in autonomous way. The valid syntax is:

```
GSSOut=gs_nopoints(GSS)
```

where the input struct variable `GSS` is as in `gs_dataload`. This function shows the points and the correspondent arrows and allows the exclusion of one or more points by means of magnification of the corresponding errors. The field `IncludedPoints`, whose value is a logical vector, is added to `GSS` (the name of the output variable also is `GSS` if `gs_nopoints` is called by `GridStrain`, an `GSSOut` if `gs_nopoints` is called by means of the above described command line). In particular, `npe=GSSOut.IncludedPoints` is such that the point `[e(m) n(m)]` is included if `npe(m)` is true and `[e(m) n(m)]` is excluded if `npe(m)` is false instead. If the field `IncludedPoints` already exists in GSS, the points for which `npei=GSS.IncludedPoints` is false are considered to be already excluded.

4. Grid creation and/or modification

The program automatically computes the grid length based on station baselines (in general, on spacing of the experimental points). The standard deviation of all the inter-distances between point pairs is computed and proposed as default value. A first grid is shown together with the data about the grid (Fig. 4.1). Together the grid, both the accepted and refused points in the previous step are shown. The user can interactively accept or refuse the proposed grid. If the grid is not accepted, both the grid step and grid size can be modified. In particular, the grid step can be changed by means of an input dialog box. The grid vertices can be interactively changed by either selecting the new lower left and the upper right corners on the figure or by means of an input dialog box. An example of edited grid is shown in Fig. 4.2.

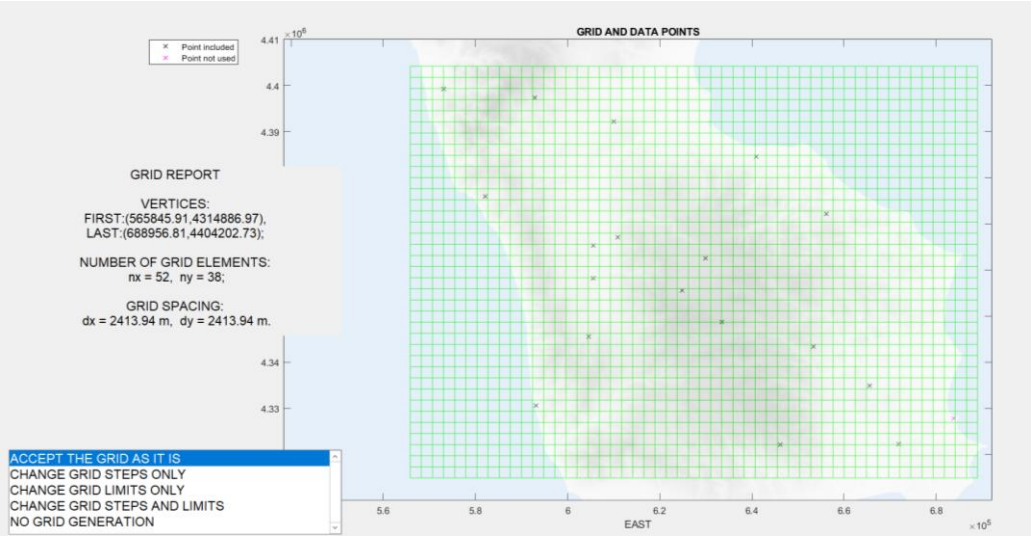

Figure 4.1 Example of default grid

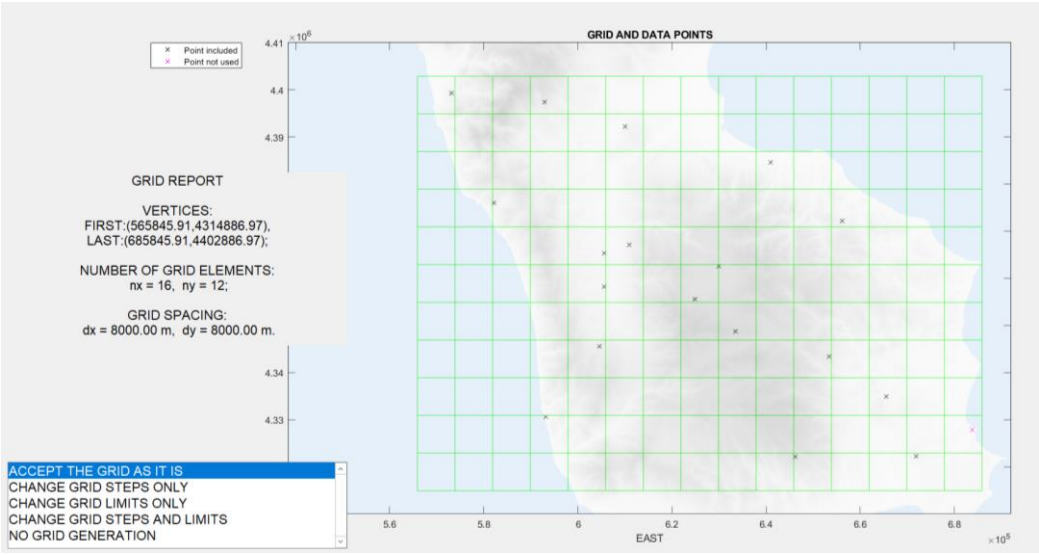

Figure 4.2 Example of edited grid

Among the initial options (Fig. 3.1), the user can choose the option "`GRID ALREADY AVAILABLE, CALCULATION OF STRAIN FIELD`". In this case, the name of file that contains the grid parameters, generated in previous session of `GridStrain`, is managed in interactive way. Note that the user, in this case, can to remove other EPs (with the same interactive procedure described in Chapter 3), as well as to reintroduce one or more points that were excluded in the previous session with a similar procedure. Once that the grid configuration is accepted, the program execution is the same for both the first two options.

The initial grid generation is carried out by means of `gs_gridgene` function, which is automatically called by `GridStrain`. The same main program `GridStrain` calls the complementary functions `gs_nopoints` (exclusion of other EPs) and `gs_yepoints` (reintroduction of already excluded EPs). If the grid is already available, `GridStrain` calls `gs_gridmod` to allow possible grid changes.

The function `gs_gridgene` can also be used in autonomous way with the syntax

```
GSSOut=gs_gridgene(GSS)
```

The autonomous use of `gs_gridmod` is also possible and can be carried out with a similar syntax.

Regardless to the way in which `gs_gridgene` is called, these new fields are added to the input `GSS` variable, leading to the output `GSS` variable, whose name is `GSS` if `gs_gridgene` is called by `GridStrain`, `GSSOut` if `gs_gridgene` is called with the above described command line:

`GridReport`	(string, see below)
`XGridStep`	
`YGridStep`	
`XGridNumber`	
`YGridNumber`	
`DownLeftCorner`	(vector of down left corner coordinates)
`TopRightCorner`	(vector of top right corner coordinates)

An example of `GridReport` is the string

```
GRID REPORT

     VERTICES:
      FIRST:(565845.91,4314886.97), LAST:(685845.91,4402886.97);

     NUMBER OF GRID ELEMENTS:
      nx = 16,  ny = 12;

     GRID SPACING:
      dx = 8000.00 m,  dy = 8000.00 m.
```

If the automatic data saving was chosen, as the grid is generated and accepted by means of `gs_gridgene` or modified and accepted by means of `gs_gridmod` automatically called by `GridStrain` (and possible EP exclusion or reintroduction, always managed by means of `GridStrain`, is completed), the `GSS` variable is saved in the file `CommonPart2DG.mat` (see data saving options, chapter 3). If the manual data saving was chosen, the output filename can be interactively managed. If the data saving was disabled, no data saving is carried out.

5. Definition and/or change of the scale factor

Once the grid satisfies the user preferences, the program automatically defines the smoothing parameter, or scale factor, for the modification of the least square weighting matrix in an MLS approach, as in Shen *et al.* (1996). This default value is defined as three time the grid spacing, but the user can modify it by means of an input dialog box (fig. 5.1). The scale factor characterizes the strain calculation. In particular, if data points are widely dense distributed, the local strain can be estimated at each node of the grid (or also in a defined point) using a weighting strategy to automatically lower contribution of far stations from the node. All available data are involved in computation but errors are rescaled using an appropriate function which increases with distance. Following Shen and Jackson (2000), the weigh function $\exp(-d/d0)$ can be used, where d is the distance between the node and the grid point and *d0* is the smoothing parameter (see 3.1 about other available scaling functions, i.e. Gaussial and inverse square). Except for the weighting function, which introduces a scale factor in the weight matrix used in the least square computations, the approach is exactly the same described e.g. in Livieratos (1980).

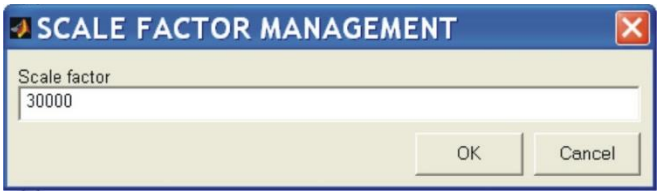

Figure 5.1. Definition of scale factor for weighting (default: three times the grid spacing)

6. Strain computation on the grid nodes and significance evaluation

The MLS computation lead, for each grid node, to the displacement gradient **L,** which is a tensor and can be written as $\boldsymbol{L} = \boldsymbol{E} + \boldsymbol{\Omega}$, where $\boldsymbol{E} = \varepsilon_{ij} = (\partial_i u_j + \partial_j u_i)/2$ is the strain tensor, $\boldsymbol{\Omega} = \omega_{ij} = (\partial_i u_j - \partial_j u_i)/2$ is the rotational part of **L**, u_i is the displacement u_x (u_y) for $i = 1$ ($i = 2$), and ∂_i is the partial derivative operator $\partial/\partial x$ ($\partial/\partial y$) for $i = 1$ ($i = 2$). The strain tensor is symmetric and represents the internal deformation, whereas $\boldsymbol{\Omega}$ is antisymmetric and represents a rigid body motion. The formulas are

$$\boldsymbol{L} = \begin{bmatrix} \frac{\partial}{\partial x}u_x & \frac{\partial}{\partial y}u_x \\ \frac{\partial}{\partial y}u_y & \frac{\partial}{\partial y}u_y \end{bmatrix} = \boldsymbol{E} + \boldsymbol{\Omega}, \tag{6.1}$$

$$\boldsymbol{E} = \begin{bmatrix} \frac{\partial}{\partial x}u_x & \frac{1}{2}\left(\frac{\partial}{\partial y}u_x + \frac{\partial}{\partial x}u_y\right) \\ \frac{1}{2}\left(\frac{\partial}{\partial y}u_x + \frac{\partial}{\partial x}u_y\right) & \frac{\partial}{\partial y}u_y \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix}, \tag{6.2}$$

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & \frac{1}{2}\left(\frac{\partial}{\partial y}u_x - \frac{\partial}{\partial x}u_y\right) \\ -\frac{1}{2}\left(\frac{\partial}{\partial y}u_x + \frac{\partial}{\partial x}u_y\right) & 0 \end{bmatrix} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}. \tag{6.3}$$

Since the strain tensor **E** is real and symmetric, it can be diagonalized, i.e. there exists an invertible matrix **V** wherein $\boldsymbol{E_d} = \boldsymbol{V}^{-1}\boldsymbol{E}\boldsymbol{V}$, where $\boldsymbol{E_d}$ is a diagonal matrix. The eigenvalues $e_{\max}$ and $e_{\min}$ are the maximum and minimum principal strain respectively. The maximum (minimum) strain is the change of length per unit-length in the direction of maximum (minimum) extension, positive for extensions. In the reference frame having origin in the grid node and whose unit vectors are the orthonormal eigenvectors, no shear deformation is present and variations occur along the principal axes only.

The `GridStrain` toolbox computes, for each grid node, the eigenvalues $e_{\max}$ and $e_{\min}$, the principal directions (i.e. the corresponding normalized eigenvectors), the displacement and the rigid body rotation, i.e. ω in (6.3). Once a scale factor is available, the program starts the strain rate estimation over each grid node. The redundant equation system is estimated according to a standard linear least square approach where the weight function is introduced to provide an error scaling as described in Chapter 5.

In addition, the geometric significance of the results is evaluated in order to recognize the grid nodes where the results are really representative of the local strain for the chosen scale factor. The corresponding flag field is generated. In particular, to each grid node is associated a flag indicating if:

- The grid node has high significance (flag 2). In this case, in each of the three 120°-width regions in which the plane can be subdivided there exists at least a data point whose distance from the grid point is lower than (or equal to) the scale factor. The spatial distribution of the data points around the grid point is in this case in good.
- The grid point has mid significance (flag 1). In this case, in two of the three 120°-width regions in which the plane can be subdivided, there exists at least a data point whose distance between the grid point is lower than (or equal to) the scale factor. The spatial distribution of the data points around the grid point is in this case near to validity, but the estimate value of the strain is not completely representative of the compressive and/or tensional state of the area (insufficient data points, but situation not completely bad).
- The grid point has low significance (flag 0). In this case, the result cannot be used.

The computations are carried out by `gs_itera` function, which is automatically called by `GridStrain`. The function `gs_itera` can laso be used in autonomous way; in this case, the syntax is

```
GSSOut=gs_itera(GSS,d0)
```

where `GSS` is a `GSS` struct variable with the information about the grid (see Chapter 4) and `d0` is the scale factor. This function automatically calls the sub-function `gs_strain` for each grid node (the use of `gs_strain` as autonomous function is not recommended).

Regardless to the fact that `gs_itera` is automatically called by `GridStrain` or manually called by the used with the command shown above, these new fields are added to the resulting `GSS` struct variable:

`MaxStrainMatrix`	maximum strain matrix
`MinStrainMatrix`	minimum strain matrix
`AzimuthStrainMatrix`	matrix of azimuth, i.e. angle between north direction and maximum strain vector
`MaxStrainMatrixError`	error on maximum strain
`MinStrainMatrixError`	error on minimum strain
`NumberMatrix`	if `numb=GSS.NumberMatrix`, `numb(m,l)` is the number of experimental points whose distance from the grid node `[X(m,l),Y(m,l)]` is smaller than or equal to `d0`
`ComputationFlag`	is `1` in the grid nodes where the strain is actually computed, `0` otherwise
`SignificanceFlag`	Flag on significance. Let `FLS` be `GSS.SignificanceFlag`; - It is `FLS(m,l)=2` if all the three sectors with 120° angular width around the grid node `[X(m,l),Y(m,l)]` contain at least one experimental point whose distance by this grid node is lower than or equal to `d0` (`HIGH SIGNIFICANCE GRID POINT CASE`). - It is `FLS(m,l)=1` if two sectors contain at least one point as above (`MID SIGNIFICANCE GRID POINT CASE`). - It is `FLS(m,l)=0` otherwise (`LOW SIGNIFICANCE GRID POINT CASE`). Obviously, `FLS` is 0 in the grid nodes in which `FLC=0`, where `FLC=GSS.ComputationFlag`.
`UTranslation`	matrix of velocity/translation East components
`VTranslation`	matrix of velocity/translation North components
`UTranslationError`	error on `UTranslation`
`VTranslationError`	error on `VTranslation`
`Rotation`	for each grid node, it is the non-zero component of the rotation antisymmetric matrix

`RotationError`	error on `Rotation`

If, before the strain calculation, `GSS` already has the fields related to strain rate data because a previous session of `GridStrain` (or `gs_itera`) was carried out, the corresponding values are overwritten without any warning message.

As the computations for a defined scale factor are completed, if user did not excluded the data saving, the output `GSS` variable is saved. In case of automatic data saving, if the chosen filename common is, for example, `MyStrainData`, and the scale factor is 30 km, the name of the generated `GSS` file is `MyStrainData2DSF30000.mat`.

7. Data visualization

The next `GidStrain` step is data visualization. An interactive figure (Fig. 7.1) allows the choice of visualization; the data can be shown in all grid nodes, in high significance grid nodes only and in high and mean significance grid nodes.

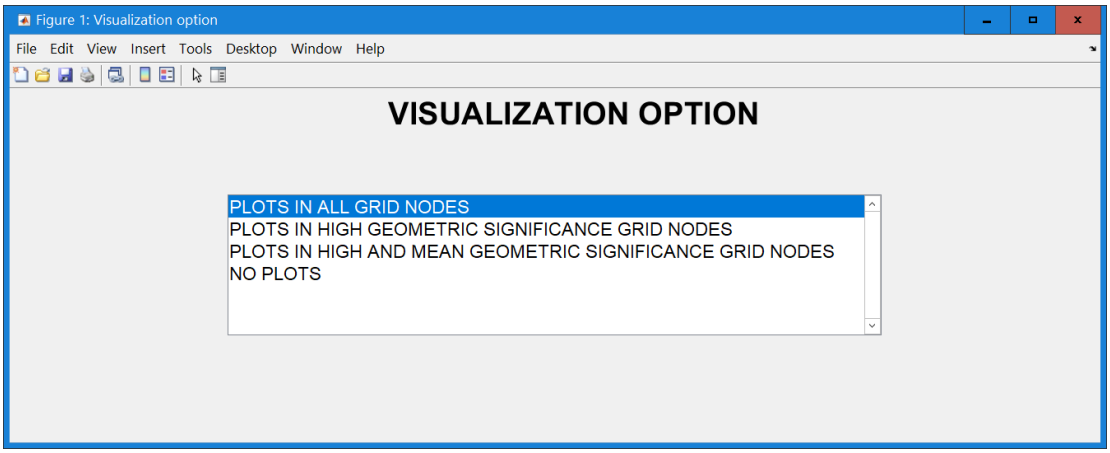

Figure 7.1 Choice of visualization option.

If an option different from "`NO PLOTS`" is chosen, the fist plot is the contour map of significance areas. An example is shown in Fig. 7.2. A continuous line represents the boundary of the high significance area.

If a background image was selected (see Chapter 3), it is shown in all plots. Moreover, in each plot the scale factor used in the computations is shown in the figure title.

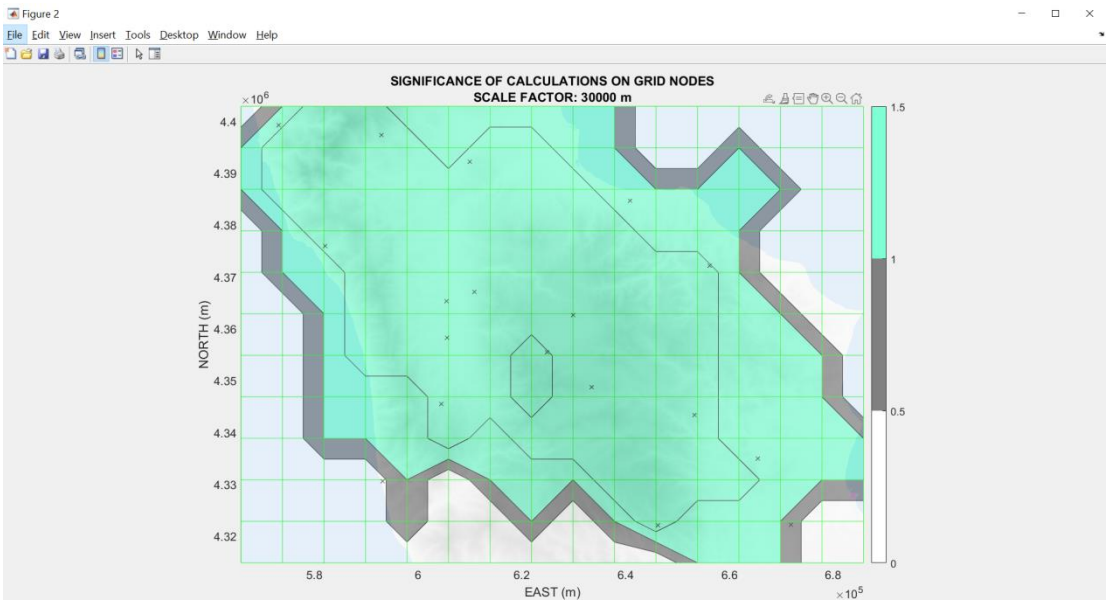

Figure 7.2 Geometric significance plot

These plots are shown:

1. Strain rate field. The eigenvectors are shown with their directions and two colors: if an eigenvalue is positive (extension) the color is blue, whereas the color is red in the case in which the eigenvalue is negative (compression). The arrow lengths are proportional to the absolute values of the corresponding eigenvalues (strictly speaking, an eigenvector is a unit vector). For representation purposes, the vector scale is such that the longest vector has length equal to the grid step. This plot also allows the selection of a grid node and the

presentation of the corresponding results (see below about the grid node data that can be shown) as well as an area selection. If the user choose the visualization of strain rate in both high and mean significance areas, the eigenvectors in the mean significance areas are shown with dashed lines (Figs. 7.3-7.5).

2. Contour plot of rate of change in area. For each grid node, it is the trace of the strain tensor, i.e. $e_{\max} + e_{\min}$, where $e_{\max}$ and $e_{\min}$ are the maximum and minimum eigenvalue respectively (Fig. 7.6). This plot allows the characterization of the kinematics areas.

3. Contour plot of the prevailing eigenvalues (Fig. 7.7), i.e., for each grid node, the eigenvalue having the highest absolute value.

4. Contour plot of the normalized shear rate, i.e. of engineering shear normalized to the change in area. It is $(e_{\max} - e_{\min})/(e_{\max} + e_{\min})$. Note that values near zero characterize strain tensors whose components are comparable in modulus, while values near ±1 are characterized by the predominance of a component on the other (Fig. 7.8). This plot allows the characterization of boundaries between different kinematic areas.

5. Contour plot of the second invariant of the strain rate, i.e. $\sqrt{\dot{\varepsilon}_{xx}^2 + \dot{\varepsilon}_{yy}^2 + 2\dot{\varepsilon}_{xy}^2} = \sqrt{\dot{\varepsilon}_{\max}^2 + \dot{\varepsilon}_{\min}^2}$, which represents the magnitude of total strain rate (Fig. 7.9).

6. Contour plots of the relative error on maximum and minimum strain, drawn on the same figure (Fig. 7.10). In order to avoid saturation effects due to areas characterized by very high errors, the contour plots of $\log_{10}(\sigma_{e\max}/|e_{\max}|)$ and $\log_{10}(\sigma_{e\min}/|e_{\min}|)$ in the high and mid significance areas are shown.

7. Plots of strain rate field (i.e. eigenvectors) overlapped on contour plots of prevailing eigenvalues in the high significance areas and in the high and mean significance areas, drawn in the same figure (Fig. 7.11).

8. Vectors of local rigid-body translation field (Fig. 7.12). Please note that the interpretation of this plot is not straightforward, and is scale factor dependent. For example, a very high scale factor is chosen, the result is the velocity field in uniform strain conditions.

Note that the figures 1-5 and 7 (i.e. Figs. 7.3-7.10 and 7.12 shown below) are drawn according to the user's choice about the strain on the grid nodes, i.e. all grid nodes, high significance nodes, high and mean significance nodes. Figure 6 (i.e. Fig. 7.11 shown below) is the same whichever is the option chosen about the data representation instead.

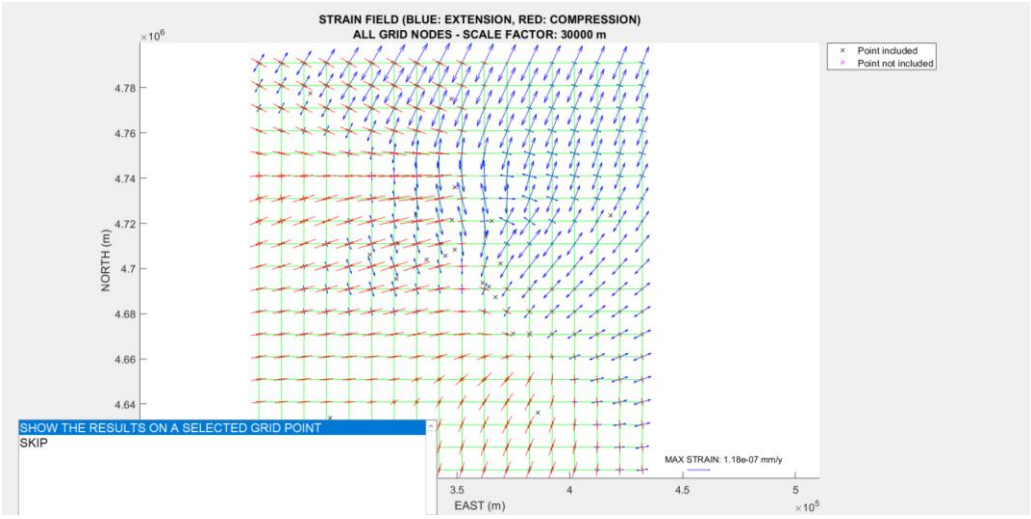

Figure 7.3 Example of strain rate field representation in all grid nodes. Note the different colors and orientations of the vectors.

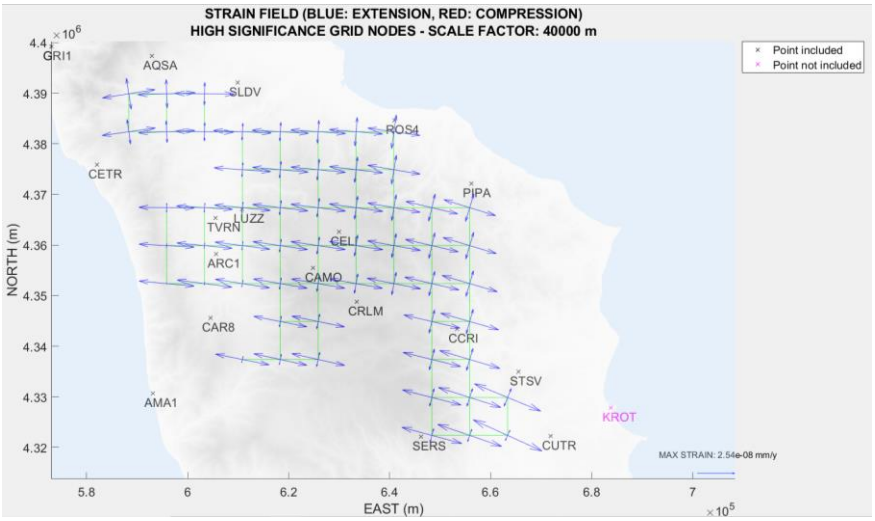

Figure 7.4 Example of strain rate field representation in high significance grid nodes.

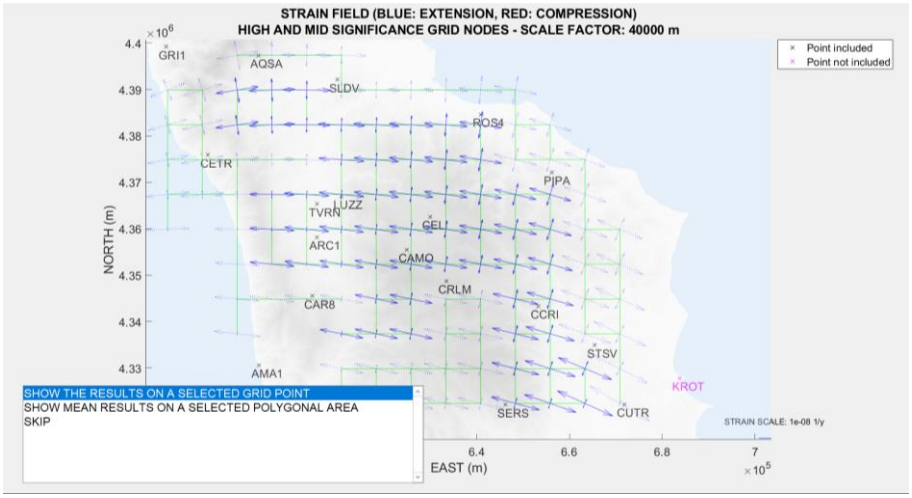

Figure 7.5 Example of strain field representation in high (solid lines) and mean (dashed lines) significance grid nodes.

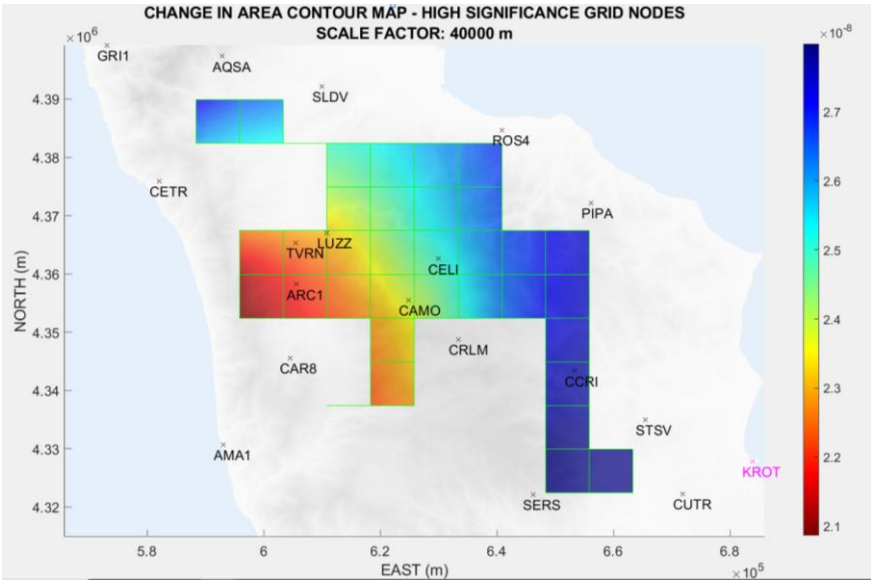

Figure 7.6 Example of rate of change in area contour map

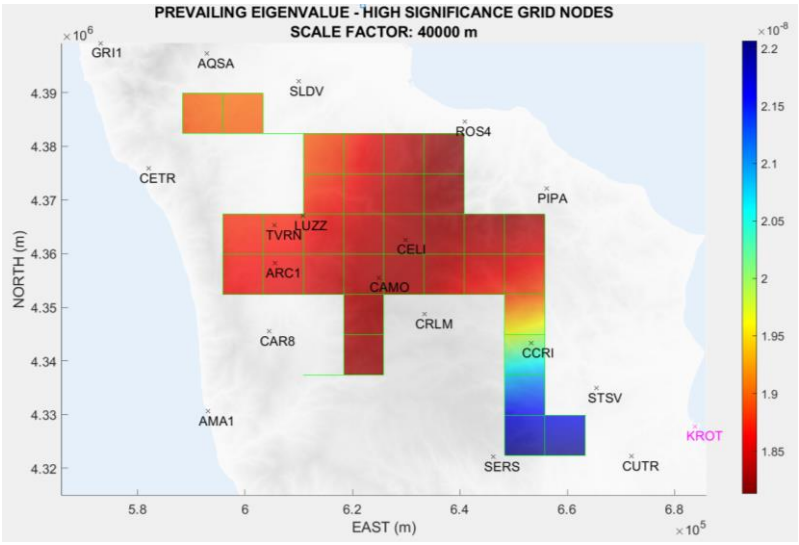

Fig. 7.7 Contour plot of prevailing eigenvalue.

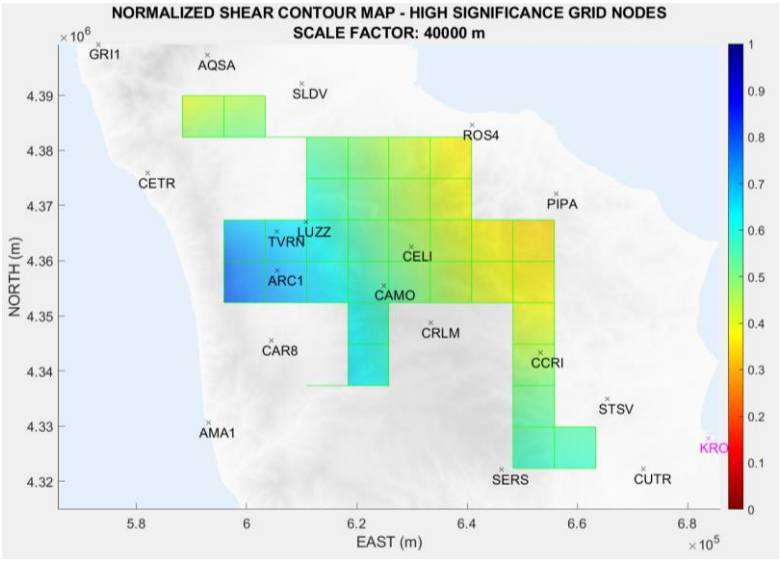

Figure 7.8 Contour plot of rate of engineering shear normalized to the change in area.

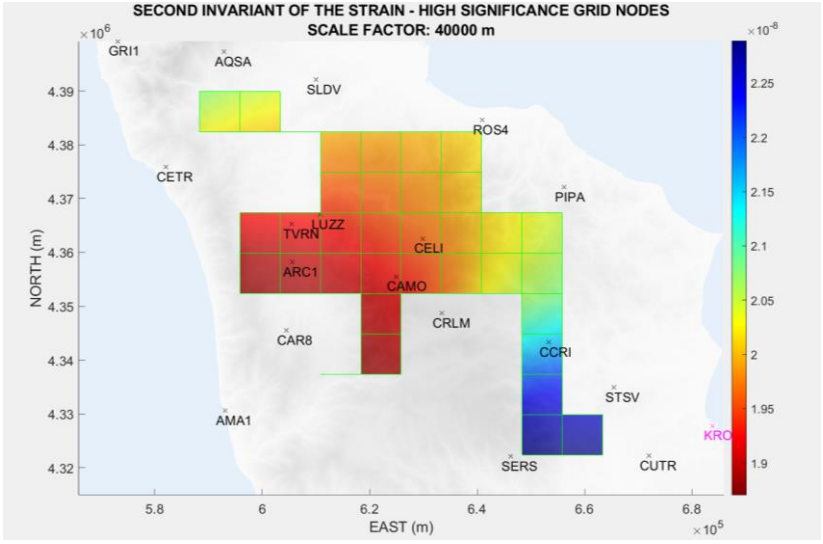

Figure 7.9 Contour plot of second invariant of the strain rate.

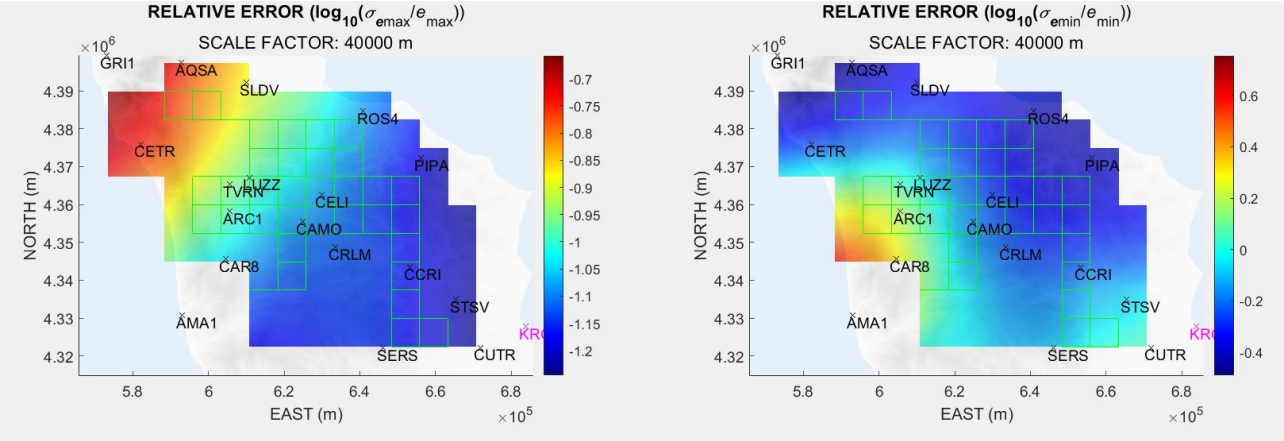

Figure 7.10 Contour plots of relative errors on strain computation, expressed as logarithms.

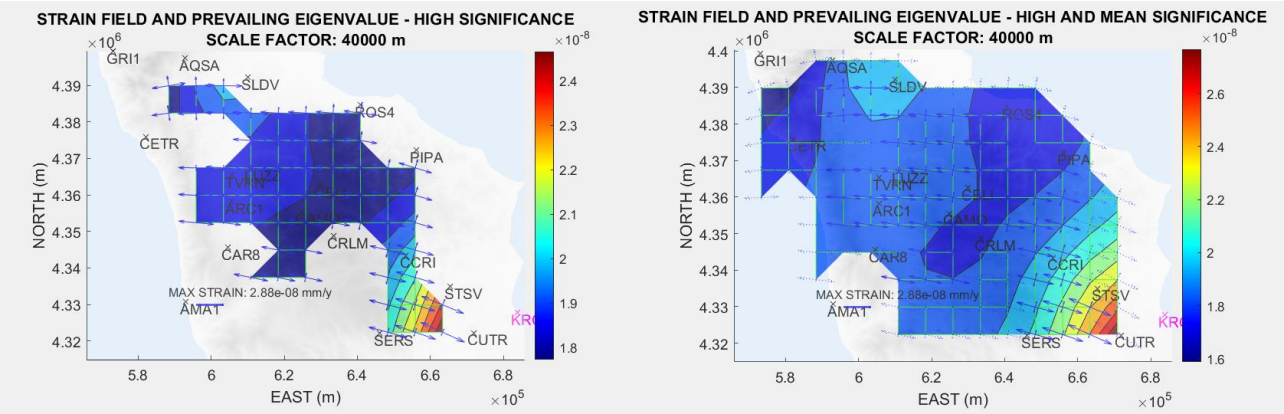

Figure 7.11 Strain rate fields in high (left panel) and high-mean significance areas (right panel) overlapped to the contour maps of prevailing eigenvalue.

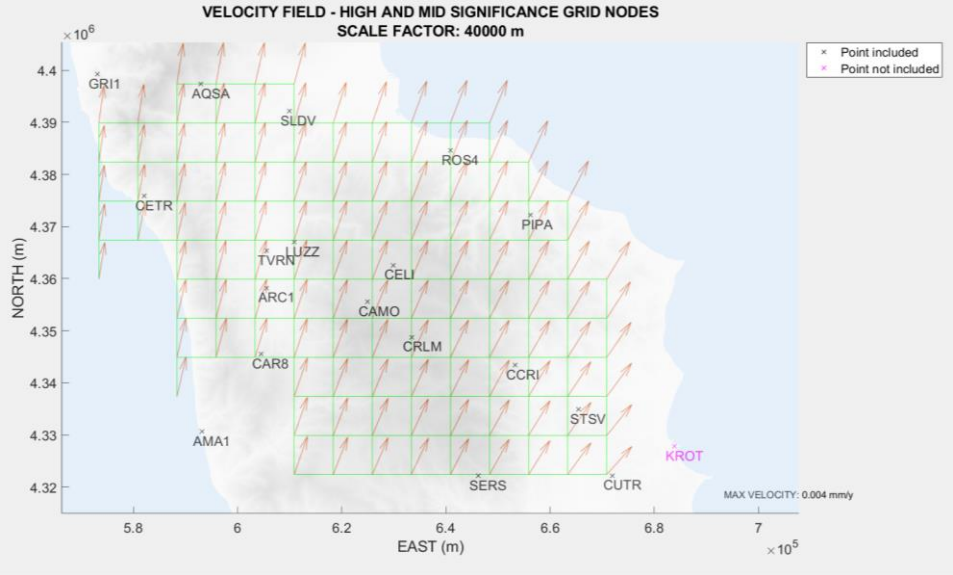

Figure 7.12 Velocity field estimated from strain field.

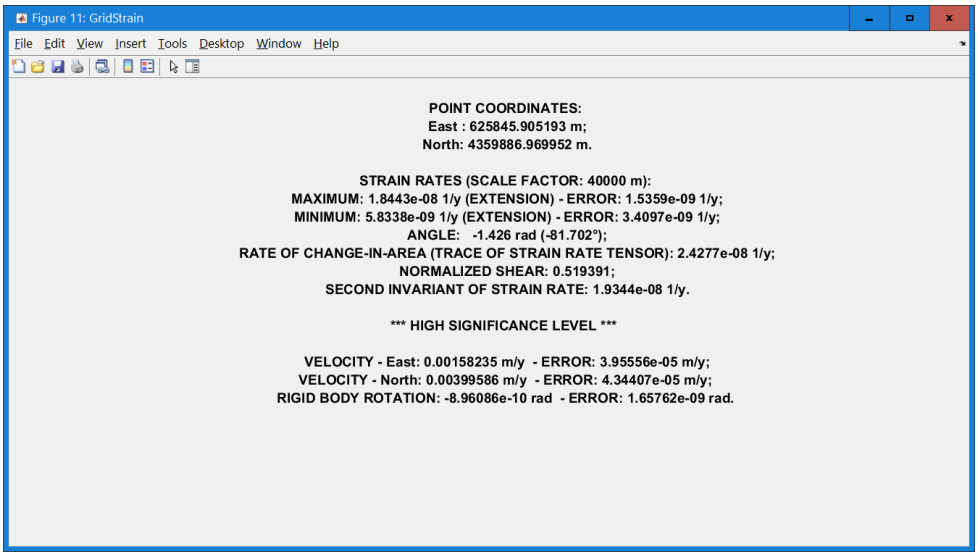

Figure 7.13 Example of data about a selected grid node.

As mentioned above, the program allows, at this stage, the interactive selection of one or more points and the visualization of the corresponding strain rate by acting on the figure which represents the strain rate field. No more computations are carried out; the data about the nearest grid node to the selected point are shown. If the user wants to have the data about a well-defined point, regardless to the grid used for `GridStrain` computations, must use the `StrainZero` program instead of `GridStrain` (see Chapter 9). The results of this selection are contemporarily shown in a menu box and in the command window (note that no commands are typed using this command window after the program initialization; it is used for visualization only). An example of visualization of these results is shown in Fig. 7.13 (this visualization is repeated on the MCW). The shown data are: coordinates of the nearest grid point, strain eigenvalues, angle for the definition of the direction of the first strain eigenvector, rate of change-in-area, rate of normalized shear, geometric significance of the results, scale factor, and rigid-body kinematic parameters (velocity and rotation).

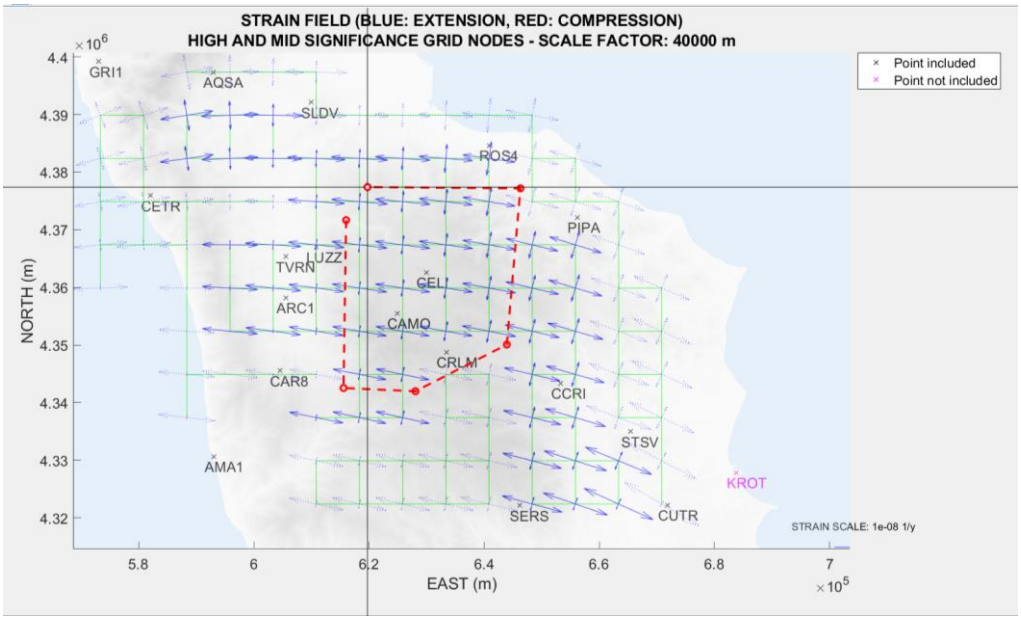

Figure 7.14 Polygonal area selection.

The user can also interactively select an area and visualize the mean values in this area (option SHOW MEAN RESULTS IN A SELECTED POLYGONAL AREA, Fig. 7.5). An

example of interactive selection, carried out by means of mouse (right button) is shown in Fig. 7.14. To complete the selection and acquire the polygon vertices, the used should use the return button on keyword (please note that the last acquired vertex is the one selected with the last righ-selected by mouse. An example of visualization of the corresponding results is shown in Fig. 7.15.

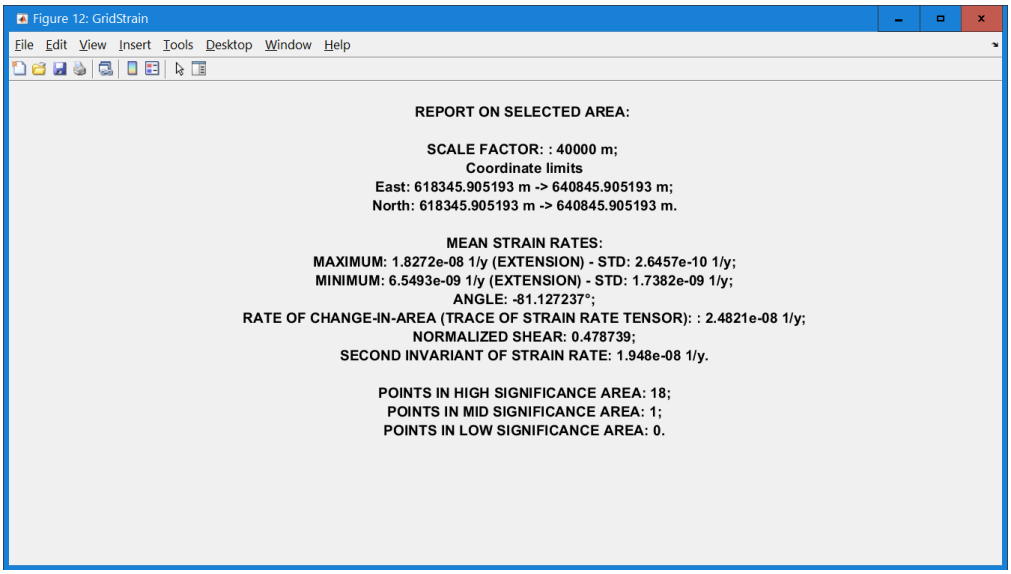

Figure 7.15 Example of results of polygonal area selection.

The visualization of strain data on a selected grid point and/or a selected area can be repeated as many as the user wants.

The data visualization is carried out by means of `gs_strashow` function, which is automatically called by `GridStrain`. This function can also be used in autonomous way. The valid syntaxes are

```
gs_strashow(GSS)
gs_strashow(GSS,visScale)
```

The input argument `visScale` defines the scale of representation. If it is undefined or empty, it is

```
visScale=min(dx,dy)/max(mean(abs(EMAX(:))),mean(abs(EMIN(:)))),
```

where `dx`, `dy`, `EMAX` and `EMIN` are taken from `GSS`.

As the visualization managed by means of `GridStrain` is completed, the user can choose a different scale factor and repeat the calculations. Please note that this option is also available if `GridStrain` is used to display existing strain rate data. If the data visualization is carried out by calling `gs_strashow` with a command line, no other scale factors can be considered in the same session.

8. Complementary functions and options

Some complementary functions are available in order to allow the implementation of some activities:

- Generation of a background image from a DTM;
- Data exportation in ESRI `.asc` ASCII format.

8.1 Generation of a background image from a DTM

A background image can be used for all the `GridStrain` visualization. As described in Chapter 3, the background image should be already available and manageable by means of two files:

1. A JPEG or a TIFF image file with a filename like, e.g., `'MyBGImage.jpg'`;
2. A MATLAB .mat file having the same name of the image file but extension `.mat`, (e.g. `'MyBGImage.mat'`) with the field `ALI` whose value is the vector `[Xdatamin, Xdatamax, Ydatamin, Ydatamax]` with the coordinates of the vertices of the area represented by the image, expressed in the same reference frame of the input velocity data.

If a DTM file in ESRI `.asc` format (see 8.3 for more information about this format) or in GeoTiff `.tiff` format[1] provided by a GIS package is available, a complementary function allows the generation of these two files. This function is `gs_genDTMImage`; the syntax is:

```
ALI=gs_GenDTMImage(filenaDTM,OtherNoData,OutName)
```

The input arguments are:

`filenaDTM`	name of input ESRI `.asc` file or GeoTiff `.tiff` file. If it is undefined or empty, the filename can be interactively managed;
`OtherNoData`	possible second no-data value (the file already has a no-data value. However, the generation of a DTM by adding different sources could lead to a second no-data value in elements of the DTM for which there are no sources). If it is undefined or empty, no a second no-data value is considered;
`OutName`	name of the output files. If it is undefined or empty, the default name `'refDTMIm'` is used. If `OutName` includes the extension, it is used for the image file (allowed extensions: `'.jpg'`, `'.jpeg'`, `'.tif'`, `'.tiff'`). If `OutName` does not include the extension, `'.tif'` is used. In each case, the second generated file has the same name of the image one (except for the extension);
`OptSea`	option about sea. If it is true or is the string `'light'` or `'dark'`, the sea (DTM level zero) is colored in blue. In particular, if `OptSea` is `'light'` or `'dark'`, the sea is in light blue or dark blue respectively. If `OptSea` is simply true, `'light'` is used. If `OptSea` is undefined, empty or false, the DTM is drawn as a grey-level image using all valid data.

[1] Please note that the output files are an image (jpeg or tiff) and a MATLAB .mat file with the georeferencing data. In particular, if the output image is a tiff file, it is a standard RGB image. If the input file is a GeoTiff image, even if the extension is the same, its nature is completely different; it is a DTM, i.e. a matrix with the parameters necessary to georeferencing the grid nodes and reconstruct the elevations. It is therefore important to avoid confusion between two types of different objects (images and DTMs) even if they can be represented by files with the same extension.

The output variable is `ALI=[Xdatamin, Xdatamax, Ydatamin, Ydatamax]`, taken from the header of the input file.

Depending on the options chosen by the user for generating the output image, this function can make use of the external function `customcolormap` by Víctor Martínez-Cagigal, Biomedical Engineering Group (University of Valladolid), Spain (download page: https://it.mathworks.com/matlabcentral/fileexchange/69470-custom-colormap), which is added to `GridStrain` toolbox.

The user can customize the function `gs_GenDTMImage` or write a script using this function as a reference in order to solve his/her specific problem of generating a background image, for example in order to represent the sea level with a specific color (e.g. light blue `[171,205,239]`). The `GridStrain` developer is available to provide assistance in the eventual creation of a customized function or script, provided that the objective of the toolbox use is scientific research (email: giordano.teza@gmail.com, giordano.teza@unibo.it) .

8.2 Data export in ESRI ASCII `.asc` format

The GSS struct variable can be saved in a MATLAB `.mat` file. As above mentioned, the data saving can be also carried out in automatic way. However, the user may want to export the output data in ASCII format. In order to allow this, the `gs_saveAsc` function is available. The corresponding syntax is

```
gs_saveAsc(GSS,CommonPart,optVec)
```

This function allows the saving of data related to a GSS struct variable. Each generated file is an .asc ASCII file in ESRI DTM style, with a six rows header and a data matrix. The header is:

```
ncols           xxx
nrows           xxx
xllcorner       xxx
yllcorner       xxx
cellsize        xxx
NODATA_value    xxx
```

where

- `ncol` and `nrows` are the integer number of grid lines along X-axis and Y-axis respectively (taken from the matrices stored in GSS);
- `xllcorner` and `yllcorner`) are the X-coordinate and Y-coordinate of the grid origin, taken from `GSS.DownLeftCorner`;
- `cellsize` is the cell size, taken from `GSS.DownLeftCorner` and `GSS.TopRightCorner` (note that a square grid is required);
- `NODATA_value` is the value assigned to `NaN` matrix elements (default value: `-9999`).

If the string `CommonPart` is assigned, the data saving procedure managed by `gs_saveAsc` is completely automatic. In this case, each file has name

```
(CommonPart)(TypeofData)(ScaleFactor).asc
```

For example, if `CommonPart` is `'MyArea'`, `TypeofData` is `'MaxStrain'` and the scale factor is 40000 m, the generated file is

```
MyAreaMaxStrain40000.asc.
```

If `CommonPart` is undefined or empty, each data saving is interactively managed.

The input argument `optVec` is a vector with integer numbers in the range `1:14`.

If `optVec = 1:14`, all files are generated. If `OptVec=[n1 n2 ... nr]`, with `n1`, `n2`, `nr` natural numbers in the range 1:14, the file related to these indices are generated (see below the list).
If `optVec` is undefined or empty, a list dialog box allow the choice of the files to be generated (Fig. 8.2). A multiple choice is allowed; all files can also be generated.

List of files that can be generated:

1. Maximum strain (rate) field
2. Error on maximum strain (rate) field
3. Minimum strain (rate) field
4. Error on minimum strain (rate) field
5. Azimuth strain (rate) field
6. Number matrix
7. Computation flag
8. Significance flag
9. U translation (velocity) field
10. Error on U translation (velocity) field
11. V translation (velocity) field
12. Error on V translation (velocity) field
13. Rotation field
14. Error on rotation field

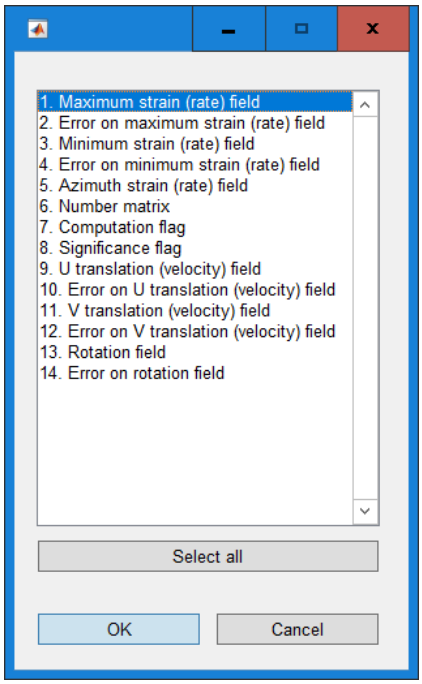

Figure 8.2 Choice of ASCII files to be generated

9. Strain rate computation without a grid: `StrainZero`.

It is a simplified version of `GridStrain`. For a data points file structured exactly as in the case of `GridStrain`, it calculates the strain in one or more points. The provided results are like the ones provided by `GridStrain`, but the generation of grid and the iteration on it are not performed. In addition, the user can choose a scale factor or exclude it (in the latter case, an infinite scale factor is used). Note that this program should be used in those cases where the user wish to have the strain data on a well-defined point and the nearest grid point is too far from it to allow a meaningful representation.

Unlike `GridStrain`, which is a MATLAB script, `StrainZero` is a MATLAB function. A function has input and output parameters, whereas the variables on which a script operates are the ones hard-coded into this script and the variables taken from external files called from the script. Moreover, all variables created in the script are added to the current workspace. `The StrainZero` syntax is

```
StrainZero(GSS)
```

The input argument is a `GSS` variable provided in a `GridStrain` session, see Chapters 3and 4 of this User's Guide (since `GridStrain` is a script, if ran, `GSS` is available on the current workspace). If the user is not interested in evaluating the strain rate field in a regular grid, he can select "`NO GRID GENERATION`" in the UI to confirm / modify the grid parameters (Fig. 4.1). No more than data about positions, velocities, velocity errors (including, if available, error ellipses) and (optionally) excluded points are required in `GSS`. All other data in `GSS`, if available, are ignored by `StrainZero`. If `GSS` is undefined or empty, the name of a file with such a variable can be interactively managed.

As `StrainZero` runs, the user can define the scale factor, or use an infinite scale factor (Fig. 9.1). The second option corresponds to the strain computation under uniform strain hypothesis.

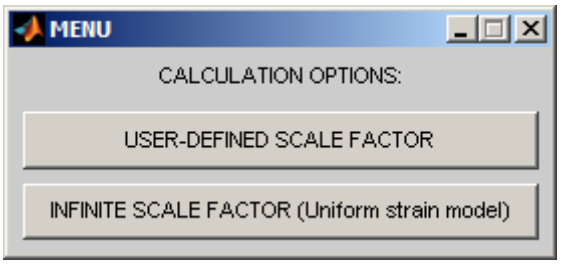

Figure 9.1. Choice of the scale factor

The strain can be computed either on the available stations (i.e. on the points whose coordinates are in the ASCII input data file, also called experimental points), or in interactively-selected points. Moreover, if at least a point was excluded from the computations, another option is available; for the experimental points, the strain can be computed either on all the experimental points on or the kept experimental points only. A menu box allows the choice of the general option on strain computation (Fig. 9.2).

In the case of computation on user-defined points, three options are available for the point selection (Fig. 9.3, left): by coordinates (Fig. 9.3, right), by click on plot, or on the center of mass, i.e. the point whose coordinates are the weighted means of the station coordinates, where the weights are the inverse squares of the uncertainties.

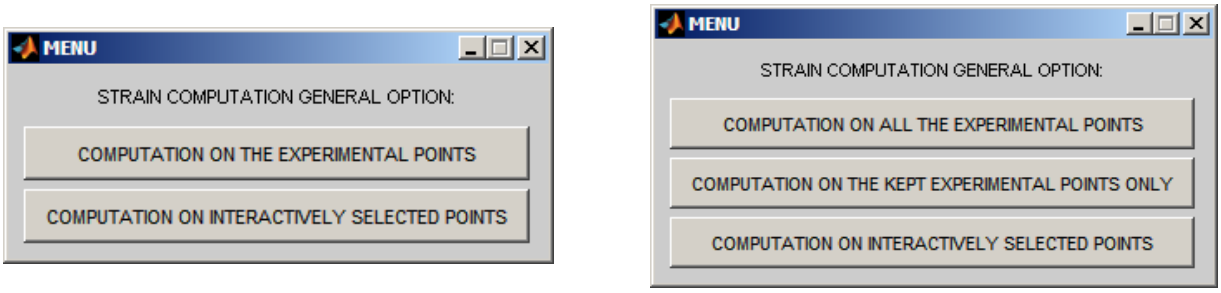

Figure 9.2. General options on strain computation: left, menu in the cases where no excluded points exist. Right: menu in the cases where at least a point has been excluded.

Fig. 9.4 shows the results of the `StrainZero` application to the sample file in the case of the choice of the computation on the EPs only. Please note that, if an EP is excluded from the computations, the strain in this point is computed, obviously without the effect of the corresponding experimental vector. The result in such an EP can be used to estimate the corresponding expected kinematics without the conditions that led to its exclusion from the strain computation, in particular, the modeled velocity vector can be compared with the experimental one (see below, in particular Fig. 9.7).

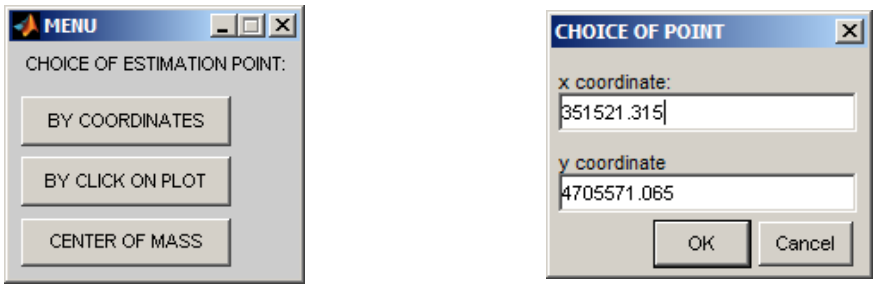

Figure 9.3. Choice of the estimation point in the case of interactive choice (left) and example of choice of the point in the case of selection of the option "by coordinates" (right).

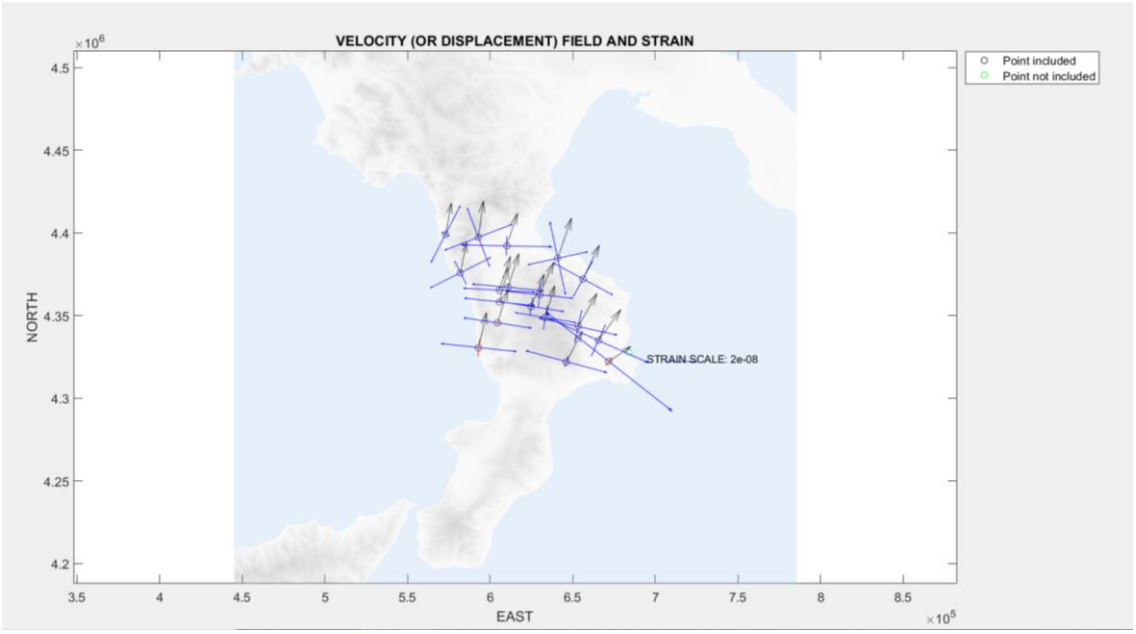

Figure 9.4 Example of computation in the experimental points based on the use of `StrainZero`.

Figs. 9.5 and 9.6 show the results of strain computation on a selected point and the corresponding report respectively (note that the point selection can be repeated how many times the User wants). Since the velocity field estimated in the experimental points can be compared with the input velocity data, the corresponding chi-square χ^2 is evaluated (see Fig. 9.7 for an example of velocity comparison. Please note that the EP 32 seems to be an outlier. This impression is well founded because this EP actually is affected by local motion due to a landslide). If at least a point is excluded from the computation, such an evaluation can be carried out either on the effectively used points (suggested option) or on all the experimental points. If no points are excluded, the χ^2 is automatically provided if the option "`COMPUTATION ON THE EXPERIMENTAL POINTS`" is chosen. Besides χ^2, the reduced chi-square χ^2_ν is also computed, where the number of degrees of freedom, ν, is 2N-6 (N is the number of kept experimental points and 6 is the number of estimated parameters). The chi-square can be very interesting in those cases where an uniform strain is modeled, i.e. an infinite scale factor is used. The corresponding report, shown on the MCW, is

```
CHI-SQUARE COMPUTATION (SCALE FACTOR: Inf):
 COMPLETE: 234.09;
 REDUCED: 5.85;
 DOFs: 40
```

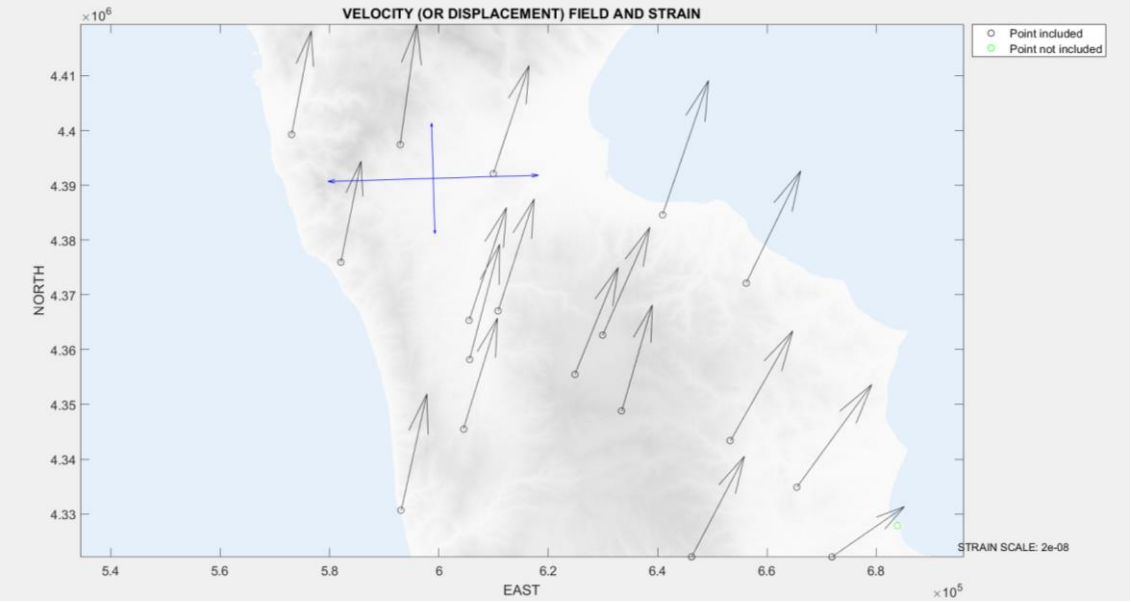

Figure 9.5 Example of strain computation on a point by using `StrainZero`.

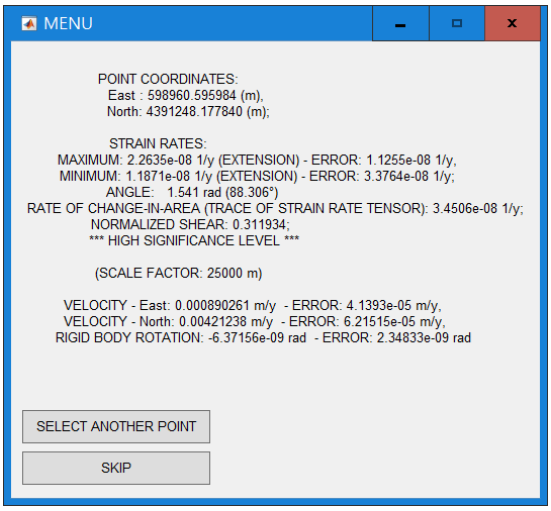

Figure 9.6 Example of report on strain computation on a single point by means of `StrainZero`. Please note that the number of digits shown is overabundant. To actually use the data in a publication, one must consider that, for example, the maximum strain is 2.3±1.1 y^{-1}.

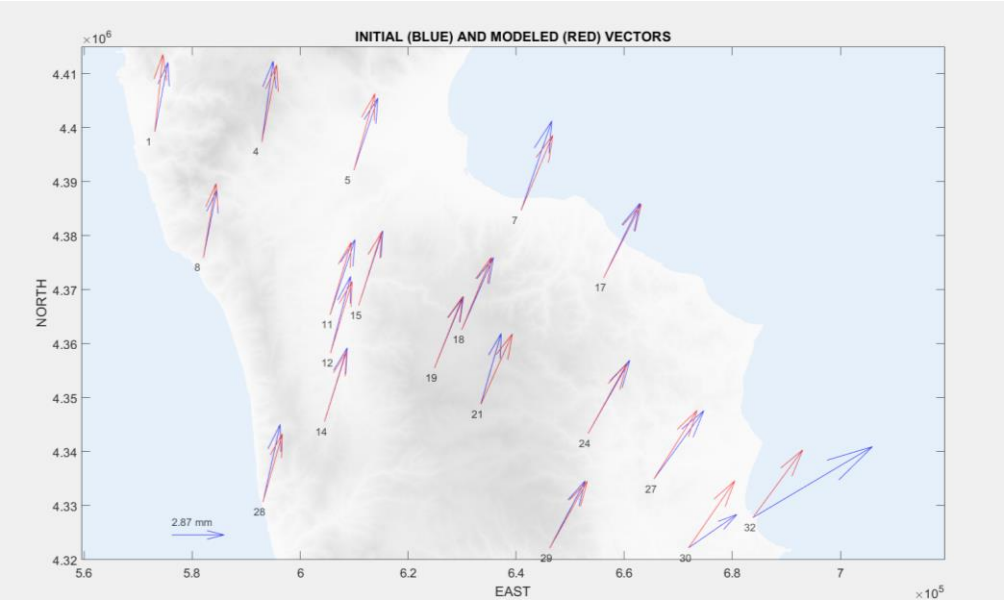

Figure 9.7 Example of experimental (blue vectors) and modeled (red vectors) velocities

The `StrainZero` function also allows the saving of the obtained results as an ASCII file. In particular, if the options "`COMPUTATION ON THE EXPERIMENTAL POINTS`" (if no exclusion is performed) or "`COMPUTATION ON ALL THE EXPERIMENTAL POINTS`" (if at least a point has been excluded from the computations), the output file contains a matrix having the same number of rows of the input ASCII file and these 21 columns:

ns e n ve vn eve evn a b theta Emax Emin eEmax eEmin Phi u0 v0 eu0 ev0 omeg eomeg

where *ns e n ve vn eve evn a b theta* are defined as in Chapter 3 (please note that *a*, *b* and *theta* are defined even if they are unavailable in the input file[2]), *Emax*, *Emin*, *eEmax* and *eEmin* are the principal strains and the corresponding errors, *Phi* is the angle for the eigenvector direction characterization, *u0*, *v0*, *eu0* and *ev0* are the rigid body translations and the corresponding errors, and *omeg*, *eomeg* are the rigid body rotation and the corresponding error respectively. If at least a point has been excluded from the computations and the option "Computation on the kept experimental points only" has been chosen, the file has 21 columns and the rows corresponding to the kept points.

If the computations are carried out on user defined points, a matrix having a number of rows equal to the number of selected points, and 14 columns is saved. The columns are in this case

k xp yp Emax Emin eEmax eEmin Phi u0 v0 eu0 ev0 omeg eomeg

where *k* is the row identifier (from 1 to the number of selected points), *xp* and *yp* are the coordinates of each selected point, and the other variables are as above.

[2] In this case, it is *a* = *eve*, *b*= *evn*, and *theta* = 0, i.e. an error ellipse whose axes are parallel to the East and North direction is defined.

3D version (`grid_strain3`)

This program is like `GridStrain`, with the difference that the strain tensors, now having three components instead of two, are computed on the points of a digital terrain model (DTM). For this reason, only the main differences are reported here. However, the current release of `grid_strain3` is like the older releases of `grid_strain` and, therefore, is not based on the use of a struct variable like `GSS` for all the quantities used and elaborated as the main program runs (for this reason, the old-style name `grid_strain3` is used). The upgrade of `grid_strain3`, leading to `GridStrain3`, is planned and is soon to be implemented.

10. Data importation in the 3D case

The ASCII file containing the data of displacements must have 13 columns (the last three are optional):

1. ns (experimental point identifier);
2. e (x coordinates, expressed in m);
3. n (y coordinates, m);
4. z (z coordinates, m);
5. ve (x displacements, m);
6. vn (y displacements, m);
7. vz (z displacements, m);
8. eve (rms error on ve, m);
9. evn (rms error on vn, m);
10. evz (rms error on vz, m);
11. a (length of major half-axis of xy error ellipse, m);
12. b (lenght of minor half-axis of xy error ellipse, m);
13. theta (azimuth of major axis of error ellipse, degrees).

Note that the displacements are now assumed to be expressed in m, since the displacement data are generally provided by terrestrial laser scanner (TLS), aerial laser scanner, digital photogrammetry or other topography techniques. This is an important difference between `grid_strain3` and `GridStrain`. In the case of `GridStrain`, the displacements are often obtained by GPS measurements and mm is used as default unit. As in the 2D case, a check on the used unit is performed, and a menu-message is presented to the user if the data could expressed in an unit different from the meter.

The optional three columns contain the data about the error ellipses of the *x* and *y* components of the displacements. In this way, if the data are provided by GPS measurements and the error ellipses are defined, the corresponding information is used[3]. The function `pp_to_gs.m` is conceived for the generation of the ASCII file using the displacements obtained by application of `PAMpoly` software to TLS data. The `PAMpoly` software implements the Piecewise Alignment method for the displacement field computation starting from two or more multi-temporal TLS observations (Teza *et al.*, 2007). In this case, the data file for the `grid_strain3` execution has 10 columns.

If the user types

[3] Generally, in a GPS measurement, the error on the third coordinate is higher than the errors on the first two ones. The error ellipse is defined on the horizontal plane only.

```
grid_strain3
```

on the command window, the program `grid_strain3` runs. The general option is shown:

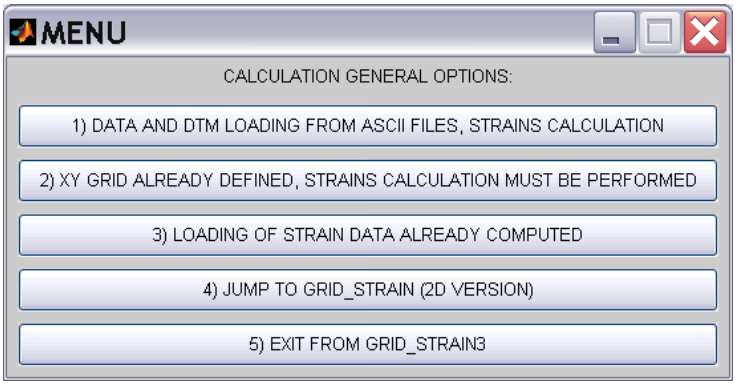

Fig. 10.1. General options menu. Note that a direct passage to `GridStrain` is possible.

The DTM used for the computations must have either the Surfer `.grd` ASCII format or the `.asc` ESRI ASCII Raster format. In all the cases, the data must be space separated (in particular, comma-separated data cannot be managed). In the Surfer-like case, the DTM data are stored in a `.grd` ASCII file whose rows are structured as follows:

`id`	:	Identification string, that identifies the file as an ASCII grid file;
`nc, nr`	:	integer numbers of grid lines along the *x*-axis (columns) and *y*-axis respectively (rows);
`xlo, xhi`	:	minimum and maximum values of *x* along the grid;
`ylo, yhi`	:	minimum and maximum values of *y* along the grid;
`zlo, zhi`	:	minimum and maximum values of *z* along the grid;
`row 6` … `last row`	:	row 6 corresponds to *ylo* and the last grid row corresponds to *yhi*. Within each row, the *z*-values are arranged from *xlo* to *xhi*. In other words, the rows 6, 7, …, last contain the matrix whose elements are the DTM's elevations.

An ESRI ASCII `.asc` grid file, generated e.g. by ArcGIS, contains six header lines that provide information about the size and limits of the grid, followed by the list of Z values. The fields within ASCII grid files must be space % delimited. The header of an ASCII `.asc` grid file is:

```
ncols         xxx
nrows         xxx
xllcorner     xxx
yllcorner     xxx
cellsize      xxx
NODATA_value  xxx
```

or, alternatively,

```
ncols         xxx
nrows         xxx
xllcenter     xxx
yllcenter     xxx
cellsize      xxx
NODATA_value  xxx
```

where

`ncols, nrows`	:	integer numbers of grid lines along the *X*-axis (columns) and *Y*-axis respectively (rows);
`xllcorner`	:	*X*-coordinate of the grid origin by lower-left corner of the cell;
`yllcorner`	:	*Y*-coordinate of the grid origin by lower-left corner of the cell;
`xllcenter`	:	*(alternative with respect to xllcorner) X*-coordinate of the grid origin by center of the cell;
`yllcenter`	:	*(alternative with respect to yllcorner) Y*-coordinate of the grid origin by center of the cell;
`cellsize`	:	Cell size (equal for X- and Y-directions);
`NODATA_value`	:	It is the value related to no data cells (`NaN` in the oputput matrix managed by MATLAB)). Although the row is compulsory (the header of an ESRI raster file must have six rows), the value can be omitted. In this case, the default `NODATA_value` (-9999) is considered, i.e. -9999.

The headers of the `.grd` and `.asc` file related to a same DTM are show below. It should be noted that the nodata value, -32767, is explicitly shown, and therefore automatically managed, only in the case of an `.asc` file.

```
DSAA
5944   5439
1757839.024   1758189.558
5143731.543   5144052.297
527.815   650.878

ncols        5944
nrows        5439
xllcorner    1757839.024005563
yllcorner    5143731.543726818
cellsize     0.05897279999999135
NODATA_value -32767.000
```

Please note that:

- In an ESRI ASCII `.asc` Raster file, the rows from the 7th one are arranged from the maximum to the minimum *y*, i.e. the row arrangement is inverted with respect to the case of a Surfer `.grd` ASCII file (ESRI: 7-th row: highest *y*, end row: lowest *y*; Surfer: 6-th row: lowest *y*, end row: highest *y*). The output MATLAB .mat DTM is arranged as a Surfer matrix, according to conventions used in MATLAB;
- the main script `grid_strain3` (through the function `gs_dtmread`) automatically discriminates between a Surfer or an ESRI file on the basis of the extension, `.grd` in the first case and `.asc` in the second one. Since some users adopt the extension `.grd` for the ESRI ASCII files, if a `grid_strain3` user do not know the origin of a DTM ASCII file, he should check the header (a warning message is provided if the file header does not correspond to the expected one);
- a combined use of the functions `gs_dtmread` and `gs_dtmwrite` allows a conversion between a `.grd` and an `.asc` ASCII DTM file (see the helps of these functions).

Obviously, other software can be used to generate the DTM, but the data must be arranged either in Surfer ASCII or in ESRI ASCII raster format. Since the ASCII file containing the DTM data is read by using the standard MATLAB function `dlmread`, non-zero values are assumed to represent the elevations (zero-values are assumed to be related to empty fields).

Please note that:

1. **if an elevation value is exactly zero, a small, but not-zero, value must be used to obtain correct computations[4];**
2. **the loss elevations (i.e., grid nodes where the interpolation algorithm did not provided results) must be characterized by abnormally high values, with a Surfer-like notation;**
3. **In the elevation matrix, each value outside the *z* range (if a Surfer .grd raster file is used) or equal to NODATA_value (if an ESRI Raster file is used) is assumed to be a not-modeled point, and no computations are performed by `grid_strain3` in this point**.

[4] Each number strictly higher than the spacing of floating point numbers (`eps`) can be used to model a zero-elevation point. To known `eps`, please type `eps` in the MCW. It is `eps` = 2^(-52)= 2.2204e-016.

11. Strain computation in the 3D case and visualizations

The strain field is computed on the DTM nodes. A subsampling can be carried out if one or both the DTM grid side are too large. Clearly, in order to properly use the information contained in a displacement field, the strain field should be computed with a similar step. The subsampled points are always elements of the original DTM, no interpolations are carried out. In particular, the options "x2", "x4" and "x8" are available for the DTM sampling step along *x* and/or *y*.

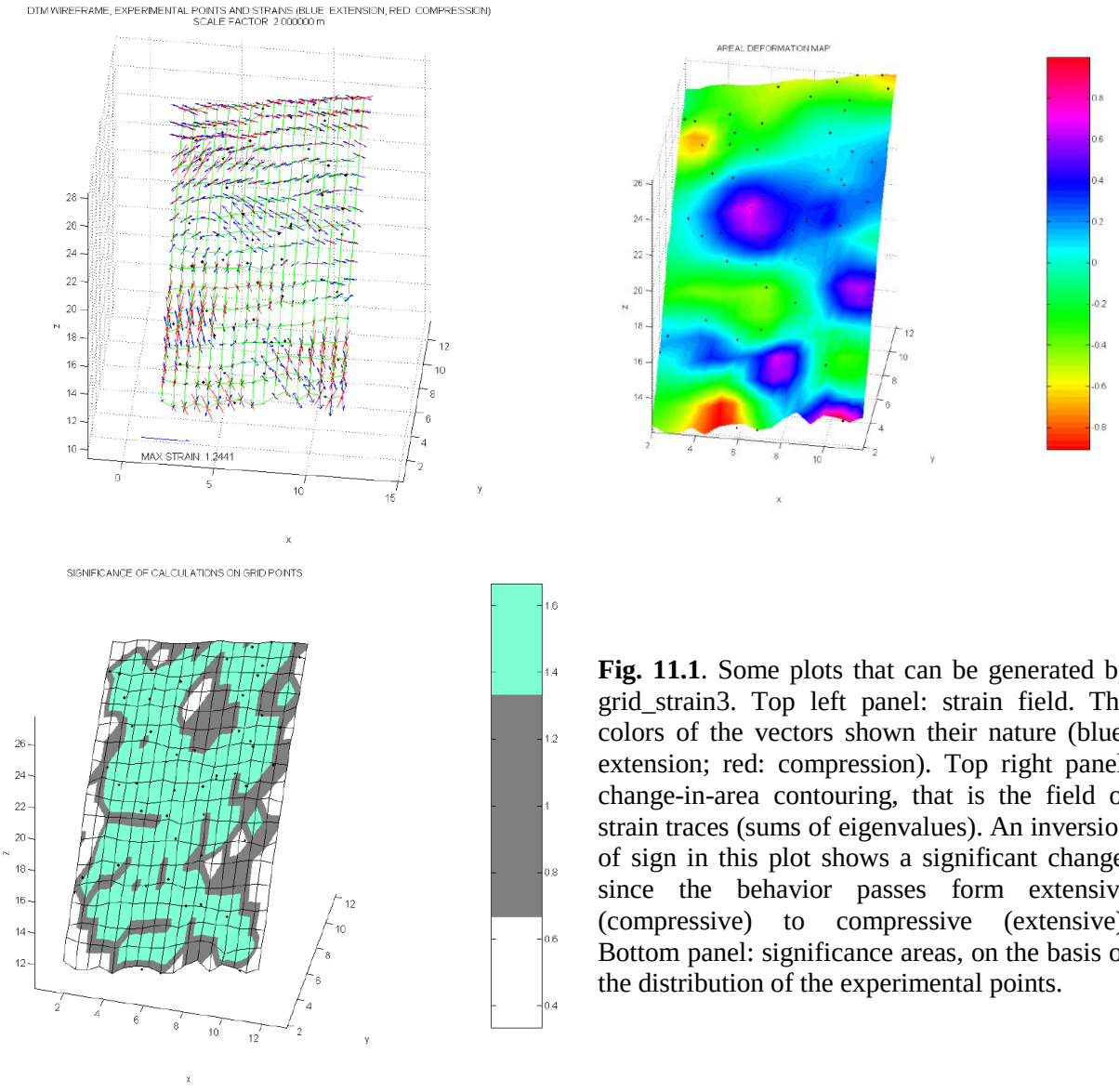

Fig. 11.1. Some plots that can be generated by grid_strain3. Top left panel: strain field. The colors of the vectors shown their nature (blue: extension; red: compression). Top right panel: change-in-area contouring, that is the field of strain traces (sums of eigenvalues). An inversion of sign in this plot shows a significant change, since the behavior passes form extensive (compressive) to compressive (extensive). Bottom panel: significance areas, on the basis of the distribution of the experimental points.

The principal strain directions are now three in each point where the calculations are performed. Flags are as in `GridStrain`, with the difference that a good distribution of the experimental points in checked in 3D.

Regarding to the eventual exclusion of one or more points from the calculations, the following facts should be taken in account:

a) All the experimental points outside the chosen DTM limits are automatically excluded.

b) Since in MATLAB environment no selection can be directly operated from a 3D view, the eventual manual exclusion is actuated on a 2D projection of the scene. The user can choose the projection (*xy*, *xz*, or *yz*).

The final data are shown by means of 3D plots. In particular, the strain field is overlapped to a wireframe representation of the DTM. In this way, if a subsampling has been chosen, only the sampled model points are shown (Fig. 11.1). The change in volume, the maximum shear strain, the errors on calculations, and the significance map are instead shown as contour maps on the DTM surface. Regarding to the shear, the user can chose the representation: natural scale, logarithmic scale (in this case, the logarithms of the absolute values are shown), or both. Also in 3D, the representation of the strain can be limited to the high significance area only, or to the high and mid significance areas. A sample data set and the corresponding DTM are available and attached to grid_strain3 software. Fig. 11.1 shows some plots can are obtained using the sample data and DTM.

In the 3D case, the following output data are provided:

1. The strain field tensors, consisting of the principal strain directions (normalized eigenvectors) and eigenvalues in each DTM point where the computations can effectively be performed. In each computation point the ellipsoid's semi-axes are plotted. Since such a plot is very memory consuming, a menu box allows the choice of the plot of the strain tensors (the memory consuming for all the other plots is low).
2. Change-in-volume, i.e. the trace of the strain tensor, presented as a contour plot rendered on DTM surface.
3. Prevailing eigenvalue, i.e. the eigenvalue having the largest modulus, shown as a contour plot.
4. Flinn's *k*-value, presented as a contour plot. This parameter encodes the shape of the strain ellipsoid. Let be $a = (1 + \varepsilon_{\max})/(1 + \varepsilon_{\text{int}})$, $b = (1 + \varepsilon_{\text{int}})/(1 + \varepsilon_{\min})$, the Flinn's k value is defined by $k = (a - 1)/(b - 1)$. The cases $k < 1$ and $k > 1$ mean respectively flattening and constriction, while $k = 1$ is the case of a planar strain.
5. Shear normalized to the strain trace, shown as a contour plot.
6. Relative errors on strain estimates, presented as a contour plot.
7. Geometrical significance of the results, shown as a contour plot.

The ASCII files `tutorial_data3.txt` (experimental points), `tutorial_dtm_in.grd` (initial DTM, i.e. DTM of a landslide before the displacement), `tutorial_dtm_fin.grd` (final DTM, related to the same landslide after the displacement) are provided within the toolbox for tutorial purposes.

13. References

Livieratos E., 1980. Crustal strains using geodetic methods. *Quaterniones Geodesiae*, **3**, 191-211.

Pesci A., Teza G., 2007. Strain rate analysis over the central Apennines from GPS velocities: the development of a new free software. *Bollettino di Geodesia e Scienze Affini*, **56**, 69-88.

Shen Z.-K., Jackson D.D., Ge B.X., 1996. Crustal deformation across and beyond the Los Angeles basin from geodetic measurements, *Journal of Geophysical Research*, **101**, 27957-27980.

Shen Z.-K., Jackson D.D., 2000. Optimal estimation of geodetic strain rates from GPS data, *EOS Transactions AGU*, **81** (19), p. S406.

Teza G., Galgaro A., Zaltron, N., Genevois R., 2007. Terrestrial laser scanner to detect landslide displacement fields: a new approach. *International Journal of Remote Sensing*, **28** (16), 3425-3446.

Teza G., Pesci A., Galgaro A., 2008a. Grid_strain and grid_strain3: software packages for strain field computation in 2D and 3D environment. *Computers & Geosciences*, **34** (9), 1142-1153.

Teza G., Pesci A., Genevois R., Galgaro A., 2008b. Characterization of landslide ground surface kinematics from terrestrial laser scanning and strain field computation. *Geomorphology*, **97** (3-4), 424-437.

Teza, G., Pesci, A., Meschis, M., 2022. A MATLAB toolbox for computation of velocity and strain rate field from GNSS coordinate time series. *Annals of Geophysics*, submitted.