StrainTool - Improving the Mapping of Tectonic Strain in Eurasia

D.Anastasiou[1,2], X. Papanikolaou[1,2], V. Kapetanidis[1,3], V. Tsironi[1], A. Ganas[1], D. Paradissis[2]

[1]Institute of Geodynamics - National Observatory of Athens
[2]Dionysos Satellite Observatory - Higher Geodesy Laboratory, National Technical Univarsity of Athens
[3]Department of Geology, National and Kapodistrian University of Athens

https://dsolab.github.io/StrainTool dganastasiou@gmail.com

12HSTAM2019
INTERNATIONAL CONGRESS ON MECHANICS
22-25.09.2019 | THESSALONIKI, GREECE
in honor of Professor Ioannis Vardoulakis on the occasion of 10 years from his death

Presentation Structure

Introduction

Open Source Software **StrainTool v1.0**

Data analysis and Validation

Strain Analysis and Discussion

Conclusions

Introduction

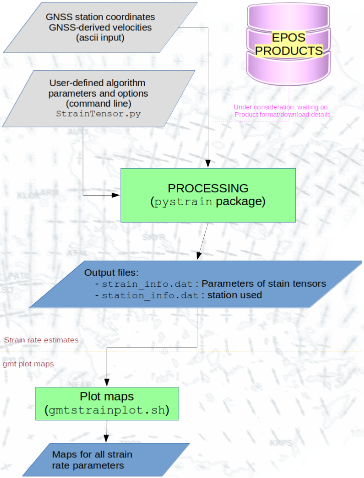
- StrainTool is a free and open-source software.
- Cooperation between the National Technical University of Athens (NTUA) and National Observatory of Athens (NOA) under EPOS-IP project.
- User-friendly software can be used directly by the scientific community.
- Pyhton programming language: free, flexible and cross-platform-compatible nature.
- Software's development was performed using Github.
- Input a list of data points along with their tectonic velovcities.
- Estimate Strain Tensor parameters.

Open Source Software **StrainTool v1.0**

StrainTool has three basic components:

- **pystrain:** A python package.
- **StrainTensor.py:** the main executable.
- A list of shell scripts to plot results from StrainTensor.py

TODO: structure design



Python Package `pystrain`

`pystrain` the core part of the project.
Python functions and classes, enable computation of strain tensor.
The package includes:

- `iotools`: input/output classes to parse ASCII files.
- `geodesy`: functions for basic geodetic calculations.
- `grid.py`: a simple grid generator
- `strain.py`: main class and necessary functions for estimation of strain tensor parameters

Estimate strain tensor parameters

Strain tensor parameters aestimated (or calculated) by solving for the system:

$$\begin{bmatrix} V_{x,S_1} \\ V_{y,S_1} \\ \cdots \\ V_{x,S_n} \\ V_{y,S_n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_{y_1} & \Delta_{x_1} & \Delta_{y_1} & 0 \\ 0 & 1 & -\Delta_{x_1} & 0 & \Delta_{x_1} & \Delta_{y_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & \Delta_{y_n} & \Delta_{x_n} & \Delta_{y_n} & 0 \\ 0 & 1 & -\Delta_{x_n} & 0 & \Delta_{x_n} & \Delta_{y_n} \end{bmatrix} \begin{bmatrix} U_x \\ U_y \\ \omega \\ \tau_x \\ \tau_{xy} \\ \tau_y \end{bmatrix}$$

$\Delta_{x_i}, \Delta_{y_i}$ are the displacement components between station i and the point.
A minimum of three stations is required to compute the parameters.

Estimate strain tensor parameters

Assuming that there is a variance information for the station velocities (and a Gaussian distribution), we can add the covariance matrix C of the velocity data in the system. In the simplest case, C is a diagonal matrix, with the velocity component standard deviations as its elements.

$$C = \sigma_0^2 \begin{bmatrix} (\frac{1}{\sigma_{V_{x_1 S_1}}})^2 & 0 & 0 & \dots & 0 \\ 0 & (\frac{1}{\sigma_{V_{y_1 S_1}}})^2 & 0 & \dots & 0 \\ 0 & 0 & (\frac{1}{\sigma_{V_{x_2 S_2}}})^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & (\frac{1}{\sigma_{V_{y_i S_i}}})^2 \end{bmatrix}$$

Shen Algorithm

Shen et al. 2015, propose a more elaborate approach.

The weighing function

$$G_i = L_i Z_i$$

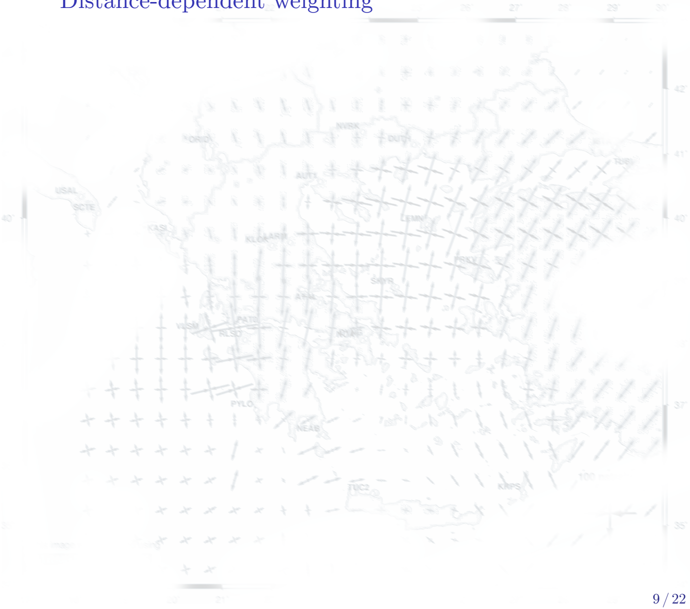
L_i : distance-dependent weighting

Z_i : spatial weighting

The final covariance matrix $C_i = C_i G_i^{-1}$

Shen Algorithm

Distance-dependent weighting



Shen Algorithm

Optimal smoothin parameter D

Smoothing coefficient D neede to actually compute the distance-dependent weights L_i.

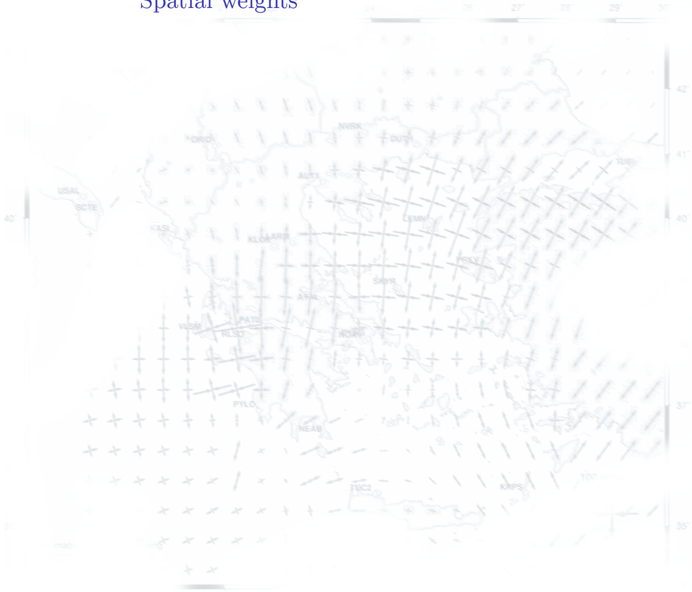
- Either pass in the parameter value as a command line option, or
- search for an optimal D value, given the range Dmin, Dmax, Dstep, and the limit value W

Searching for an optimal D value

1. If a Gaussian approach is selected, then Lmax = 2.15D (in km), while for the quadratic approach Lmax = 10D (in km).
2. Compute distance-dependent and spatial weights L_i and Z_i respectively, for every station
3. compute the summary $W = 2\sum Z_i L_i$.
4. repeat the process for the next D value if absolute value W is smaller than Wt, else current D value is optimal.

Shen Algorithm

Spatial weights



Veis Algorithm

The region is split into delaunay triangles at the barycenter of which a strain tensor is computed.
This approach uses only three points to calculate tensor parameters
Assumptions:

- 2-dimensional deformation of earth's crust in time
- Crust is considered a thin deformable shell on a spherical earth
- Mapping distortions are ignored for regions with radius of less than 5^o
- Time (earthquakes) or space (faults) discontinuities are not incuded in the calculation

Strain Tensor parameters

$$\tau_{max} = \sqrt{{\tau_{xy}}^2 + {e_{diff}}^2}$$

$$e_{max} = e_{mean} + \tau_{max} \qquad Az_{e_{max}} = 90 + \frac{-atan2(\tau_{xy}, e_{diff})}{2}$$

$$e_{min} = e_{mean} - \tau_{max}$$

$$dil = \tau_x + \tau_y \qquad 2nd\_inv = \sqrt{{\tau_x}^2 + {\tau_y}^2 + 2{\tau_{xy}}^2}$$

where,

$$e_{mean} = \frac{\tau_x + \tau_y}{2} \qquad e_{diff} = \frac{\tau_x - \tau_y}{2}$$

Input Datasets

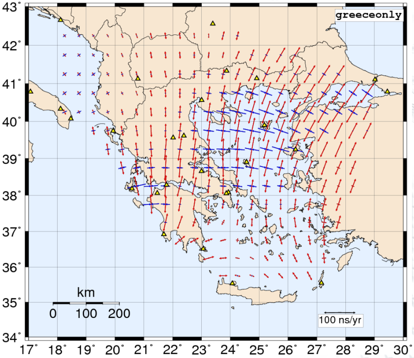
For validation:

1. EPN network, solution C2010 - ETRF 2014 (299 stations)
2. EPOS network, INGV solution (571 stations)
3. EPOS network, CNRS solution (MIDAS) (452 stations)
4. network GREECE, NTUA reprocessing 2017 (153 stations)

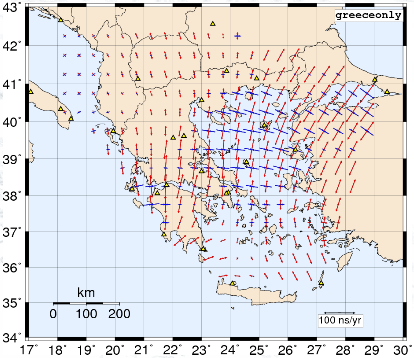
Validation

emax - emin maps comparison

VISR



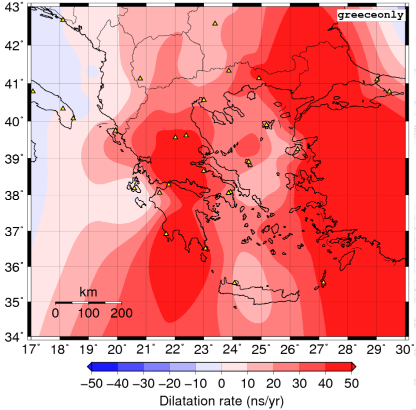
PyStrain



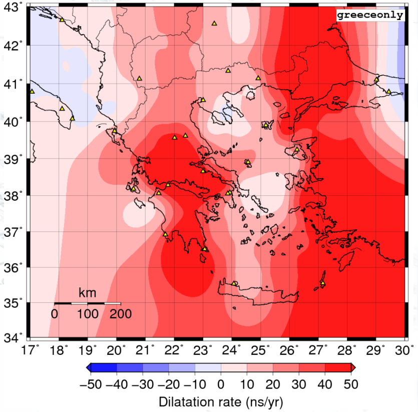
Validation

dilatation maps comparison

VISR

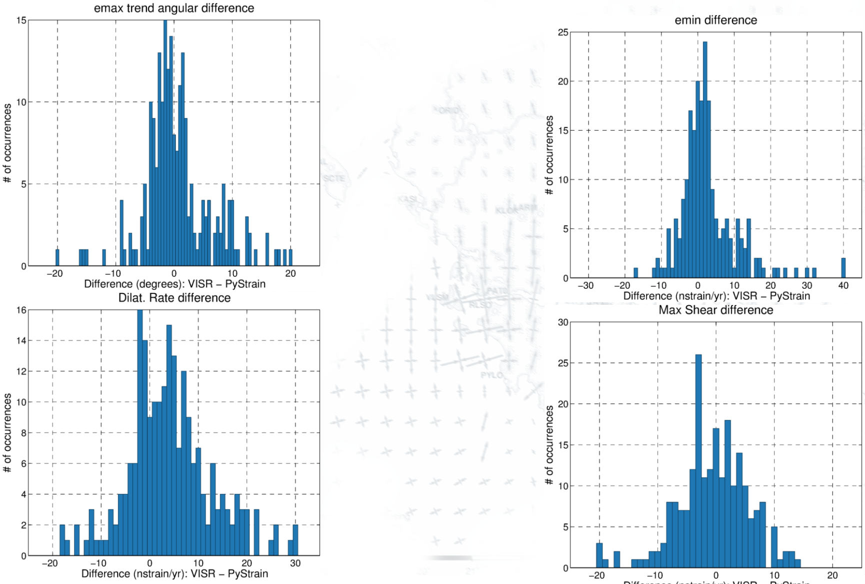


PyStrain



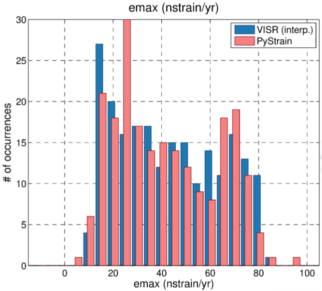
Validation

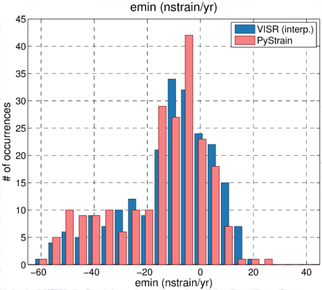
differences

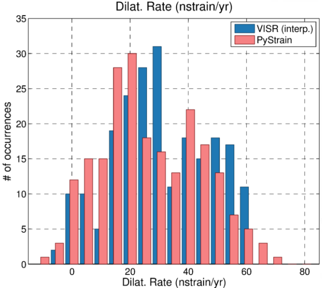


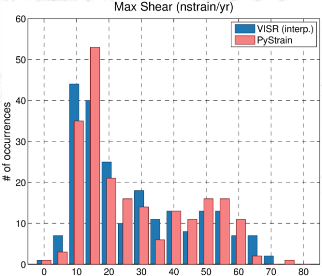
Validation

histograms









Strain Analysis

Model parameters fr Shen algorithm
Data Set: MIDAS

- Wt=6
- dmin = 1km
- dmax = 500km
- dstep = 1km
- ltype = gaussian
- x step = y step = 0.5 ° ‘
- region = 18/30/34/43

Strain Analysis

Conclusions

Thank you for your attention!