PyFWI

Release 0.0.3

Amir Mardan

Jan 02, 2022

CONTENTS:

PyFWI is a Python package for full-waveform inversion (FWI) in elastic media.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

PyFWI can be installed using `pip` as

` pip install PyFWI `

CHAPTER

TWO

PYFWI PACKAGE

2.1 Submodules

2.2 PyFWI.acquisition module

`PyFWI.acquisition.AcqParameters`(*ns*, *rec_dis*, *offsetx*, *depth*, *dh*, *sdo*, *acq_type*)

A function to define the acquisition based on user's demand

Parameters **INPA** (*dictionary*) –

A dictionnary containing required parameters for iversion, at least:

- ns: Number of sources
- rec_dis: Distance between receivers
- offsetx: Length of acquisition in x-direction
- depth: Depth of acquisition
- dh: spatial sampling rate
- sdo: Spatial differentiation order
- acq_type: Type of acquisition (0: crosswell, 1: surface, 2: both)

Returns

- **src_loc** (*float32*) – Location of sources
- **rec-loc** (*float32*) – Location of receivers

class `PyFWI.acquisition.Source`(*src_loc*, *dh*, *dt*)

Bases: `object`

A class for defining different types of sources.

Parameters

- **src_loc** (*float32*) – location of sources.
- **dh** (*float*) – Spatial sampling rate.
- **dt** (*float*) – Temporal sampling rate

`Ricker`(*fdom*)

Amethod to generate Ricker wavelet.

Parameters **fdom** (*float32*) – Dominant frequency of wavelet

delta()

A method to generate spike.

Parameters **fdom** (*float32*) – Dominant frequency of wavelet

PyFWI.acquisition.**SurfaceSeismic**(*ns*, *rec_dis*, *offsetx*, *offsetz*, *dh*, *sdo*)

A function to design a surface seismic acquisition

Parameters

- **ns** (*int*) – Number of sources
- **rec_dis** (*float32*) – Distance between receivers
- **offsetx** (*flloat32*) – Length of survey in x-direction
- **offsetz** (*float32*) – Depth of survey
- **dh** (*float32*) – Spatial sampling rate
- **sdo** (*{2, 4, 8}*) – Spatial order of finite difference method

Returns

- **src_loc** (*float32*) – Location of sources
- **rec_loc** (*float32*) – Location of receivers

PyFWI.acquisition.**acquisition_plan**(*ns*, *nr*, *src_loc*, *rec_loc*, *acq_type*, *n_well_rec*, *dh*)

acquisition_plan generates the matrix of acquisition plan

[extended_summary]

Parameters

- **ns** (*[type]*) – [description]
- **nr** (*[type]*) – [description]
- **src_loc** (*[type]*) – [description]
- **rec_loc** (*[type]*) – [description]
- **acq_type** (*[type]*) – [description]
- **n_well_rec** (*[type]*) – [description]
- **dh** (*[type]*) – [description]

Returns [description]

Return type [type]

PyFWI.acquisition.**crosswell**(*ns*, *rec_dis*, *offsetx*, *offsetz*, *dh*, *sdo*)

A function to design a crosswell acquisition

Parameters

- **ns** (*int*) – Number of sources
- **rec_dis** (*float32*) – Distance between receivers
- **offsetx** (*flloat32*) – Length of survey in x-direction
- **offsetz** (*float32*) – Depth of survey
- **dh** (*float32*) – Sampling rate
- **sdo** (*{2, 4, 8}*) – Spatial order of finite difference method

Returns

- **src_loc** (*float32*) – Location of sources
- **rec_loc** (*float32*) – Location of receivers

`PyFWI.acquisition.`**`discretized_acquisition_plan`**(*data_guide*, *dh*, *npml=0*)

discretized_acquisition_plan discretizes the matrix of acquisition plan

[extended_summary]

Parameters

- **data_guide** (*[type]*) – [description]
- **dh** (*[type]*) – [description]
- **npml** (*int, optional*) – [description]. Defaults to 0.

Returns [description]

Return type [type]

2.3 PyFWI.fwi module

2.4 PyFWI.fwi_tools module

2.5 PyFWI.model_dataset module

`class PyFWI.model_dataset.`**`Circular`**(*name*)

Bases: `object`

`Hu_circles`(*vintage*, *smoothing*)

Hu_circles a model including porosity, clay content, and saturation.

This method creates a model including porosity, clay content, and saturation. It is used in a paper published in 2021 in Geophysics by Qi Qu and his collegues. If you use non-default values, new model with the same structure and new values will be generated.

Parameters

- **rho** (*dictionary, optional*) – A dictionary containing the density of quartz, clay, water, and hydrocarbon. Defaults to None.
- **prop_back** (*dictionary, optional*) – A dictionary containing background properties (porosity, clay content, and saturation). Defaults to None.
- **prop_circle** (*dictionary, optional*) – A dictionary containing properties in the circles (porosity, clay content, and saturation). Defaults to None.
- **nz** (*int, optional*) – Number of saples in z-direction (rows). Defaults to 100.
- **nx** (*int, optional*) – Number of saples in x-direction (column). Defaults to 100.
- **r** (*int, optional*) – Radius of the circles. Defaults to 8.
- **monitor** (*bool, optional*) – Specify if you are looking for monitor model. Defaults to False.

Returns A dictionary containing the created model.

Return type model (dictionary)

Reference: Hu, Q., S. Keating, K. A. Innanen, and H. Chen, 2021, Direct updating of rock-physics properties using elastic full-waveform inversion: Geophysics, 86, 3, MR117-MR132, doi: 10.1190/GEO2020-0199.1.

louboutin(*vintage*, *smoothing*)
louboutin Generate perturbation model based on only vp.

[extended_summary]

Returns [description]

Return type [type]

perturbation_dv(*vintage*, *smoothing*)
perturbation_dv creates perturbation model in different locations

perturbation_dv creates perturbation model in different locations based on vp, vs, density

Returns [description]

Return type [type]

yang(*vintage*, *smoothing*)
Yang et al., 2018 for Truncated Newton method.

[extended_summary]

Returns [description]

Return type [type]

class PyFWI.model_dataset.**Laminar**(*name*)
Bases: `object`

Hu_laminar(*vintage*, *smoothing*)

dupuy(*vintage*, *smoothing*)

class PyFWI.model_dataset.**ModelGenerator**(*name*)
Bases: *PyFWI.model_dataset.Circular*, *PyFWI.model_dataset.Laminar*

marmousi(*vintage*, *smoothing*)

show(*property=['vp']*)

PyFWI.model_dataset.**add_anomaly**(*model*, *anomaly*, *x*, *z*, *dx*, *dz*, *height*, *type='circle'*)
add_anomaly adds anomaly to the previously created model.

This mathod add an anomally to the Earth mode that is already createad.

Parameters

- **model** (*float*) – The previously created model.
- **anomaly** (*float*) – The properties of the anomaly
- **x** (*[type]*) – x-location of the anomaly
- **z** (*[type]*) – z-location of the anomaly
- **width** (*[type]*) – Width of the anomaly
- **height** (*[type]*) – Height of the anomaly
- **type** (*str, optional*) – The shape of the anomaly. Defaults to "circle".

Returns The new model.

Return type model (dict)

PyFWI.model_dataset.**add_circle**(*model*, *circle_prop*, *r*, *cx*, *cz*)

add_circle adds a circle to the model

This function generates a circle in the model.

Parameters

- **model** (*float*) – Already created model.
- **circle_prop** (*float*) – Property of the circle.
- **r** (*int*) – Radius of the circle
- **cx** (*int*) – x_location of the center
- **cz** (*int*) – z-location of the center

Returns Return the model.

Return type model(dict)

PyFWI.model_dataset.**add_layer**(*model*, *property*, *lt*, *lb*, *rt=None*, *rb=None*)

add_layer add alyer to the model

This function add a layer to the mdoel

Parameters

- **model** (*dict*) – Already created model.
- **property** (*dict*) – Property of the new layer
- **lt** (*array, int*) – Sample number ([x ,z]) of the top of the layer in the most left part
- **lb** (*array, int*) – Sample number ([x ,z]) of the bottom of the layer in the most left part
- **rt** (*array, int*) – Sample number ([x ,z]) of the top of the layer in the most right part
- **rb** (*array, int*) – Sample number ([x ,z]) of the bottom of the layer in the most right part
- **#TODO** – to develop for dipping layers

Returns Return the model.

Return type model(dict)

PyFWI.model_dataset.**background**(*size*, *params*)

add_layer genearte a layer of property.

This method generates one layer with property "bp"

Parameters bp (*dict*) – Background property

PyFWI.model_dataset.**model_resizing**(*model0*, *bx=None*, *ex=None*, *bz=None*, *ez=None*, *ssr=(1, 1)*)

PyFWI.model_dataset.**model_smoother**(*model*, *smooting_value*)

PyFWI.model_dataset.**pcs_perturbation**(*rho=None*, *prop_back=None*, *prop_circle=None*, *nz=100*, *nx=100*, *r=8*, *monitor=False*)

pcs_perturbation a model including porosity, clay content, and saturation.

This function creates a model including porosity, clay content, and saturation. It is used in a paper published in 2021 in Geophysics by Qi Qu and his collegues. If you use non-default values, new model with the same structure and new values will be generated.

Parameters

- **rho** (*dictionary, optional*) – A dictionary containing the density of quartz, clay, water, and hydrocarbon. Defaults to None.
- **prop_back** (*dictionary, optional*) – A dictionary containing background properties (porosity, clay content, and saturation). Defaults to None.
- **prop_circle** (*dictionary, optional*) – A dictionary containing properties in the circles (porosity, clay content, and saturation). Defaults to None.
- **nz** (*int, optional*) – Number of saples in z-direction (rows). Defaults to 100.
- **nx** (*int, optional*) – Number of saples in x-direction (column). Defaults to 100.
- **r** (*int, optional*) – Radius of the circles. Defaults to 8.
- **monitor** (*bool, optional*) – Specify if you are looking for monitor model. Defaults to False.

Returns A dictionary containing the created model.

Return type model (dictionary)

Reference: Hu, Q., S. Keating, K. A. Innanen, and H. Chen, 2021, Direct updating of rock-physics properties using elastic full-waveform inversion: Geophysics, 86, 3, MR117-MR132, doi: 10.1190/GEO2020-0199.1.

2.6 PyFWI.optimization module

2.7 PyFWI.processing module

2.8 PyFWI.rock_physics module

class PyFWI.rock_physics.**Density**

Bases: object

static **effective_density**(*phi*, *rho_f*, *rho_s*)

static **fluid**(*r_hydro*, *rho_w*, *sw*)

fluid [summary]

[extended_summary]

Parameters

- **r_hydro** (*[type]*) – [description]
- **rho_w** (*[type]*) – [description]
- **sw** (*[type]*) – [description]

Returns Density of fluid

Return type rho_f [type]

gardner(*vp*, *units='metric'*)

gardner method to estimate the density

This mdethod estimate density of a model based on P-wave velocity. It uses the Gardner's eqution.

Parameters

- **vp** (*float*) – P-wave velocity
- **units** (*str, optional*) – Specify the system of the units fo measurements (Metric or Imperial) . Defaults to "metric".

Returns density

Return type rho

static matrix(*rho_clay*, *cc*, *rho_q*, ***kwargs*)
matrix [summary]

[extended_summary]

Parameters

- **rho_clay** (*[type]*) – [description]
- **cc** (*[type]*) – [description]
- **rho_q** (*[type]*) – [description]

Returns [description]

Return type [type]

rho_from_pcs(*rho_c*, *rho_q*, *rho_w*, *rho_g*, *cc*, *sw*, *phi*)
This function calculate density from Porosity, clay content, and water Saturation

rho_c: Density of clay

rho_q: Density of quartz

rho_w: Density of water

rho_g: Density of gas

cc: clay content

sw: water saturation

phi: Porosity

rho: float Effective density

PyFWI.rock_physics.**Han**(*phi*, *cc*, *a1=5.5*, *a2=6.9*, *a3=2.2*, *b1=3.4*, *b2=4.7*, *b3=1.8*)
Han estimates velocity based on porosity and clasy content

Han found empirical regressions relating ultrasonic (laboratory) velocities to porosity and clay content

Parameters

- **phi** (*[type]*) – [porosity
- **cc** (*[type]*) – clay content
- **a1** (*float, optional*) – Constant value for Vp. Defaults to 5.77.
- **a2** (*float, optional*) – Constant value for Vp. Defaults to 6.94.
- **a3** (*float, optional*) – Constant value for Vp. Defaults to 1.728.
- **b1** (*float, optional*) – Constant value for Vs. Defaults to 5.77.
- **b2** (*float, optional*) – Constant value for Vs. Defaults to 6.94.
- **b3** (*float, optional*) – Constant value for Vs. Defaults to 1.728.

Returns P-wave velocity (km/s) vs = S-wave velocity (km/s)

Return type vp

References

1. Hu et al, 2021, Direct updating of rock-physics properties using elastice full-waveform inversion
2. Mavko, G., Mukerji, T., & Dvorkin, J., 2020, The rock physics handbook. Cambridge university press.

class PyFWI.rock_physics.**Lamb**
Bases: object

vp_rho_mu(*rho*, *vp=None*, *mu=None*)

class PyFWI.rock_physics.**Mu**
Bases: object

vs_rho(*vs*, *rho=None*)
vs_rho generate mu

This function add mu to to the imported model based on S-wave velocity and density.

Parameters

- **vs** (*float or dict*) – S-wave velocity. if dict, it has to contain value for density.
- **rho** (*float, option*) – Density

Returns Shear modulus

Return type mu

class PyFWI.rock_physics.**ShearVelocity**
Bases: object

Han(*phi*, *cc*, ***kwargs*)
Han calulates vs based on Han empirical model.

Han calulates vs based on Han empirical model.

Parameters

- **phi** (*[type]*) – Porosity
- **cc** (*[type]*) – Clay content

Returns S-wave velocity

Return type vp

poisson_ratio_vs(*vp*, *sigma=0.25*)
poisson_ratio_vs calculates the shear velocity.

Calculates the shear velocity based on Poisson's ration.

Parameters

- **vp** (*float*) – P-wave velocity.
- **sigma** (*float, optional*) – Poisson's ration. It could be None if parameter "model" has this property. Defaults to None.

Returns The input model and shear velocity is added.

Return type vs

PyFWI.rock_physics.**biot_gassmann**(*phi*, *k_f*, *k_s*, *k_d*)

`PyFWI.rock_physics.`**`delta_biot_gassmann`**(*phi*, *k_f*, *k_s*, *k_d*)

`PyFWI.rock_physics.`**`drained_moduli`**(*phi*, *k_s*, *g_s*, *cs*)

`PyFWI.rock_physics.`**`error_lack_of_data`**()

`class` `PyFWI.rock_physics.`**`p_velocity`**

Bases: `object`

`Han`(*phi*, *cc*, ***kwargs*)

Han calulates vp based on Han empirical model.

Han calulates vp based on Han empirical model.

Parameters

- **phi** (*[type]*) – Porosity
- **cc** (*[type]*) – Clay content

Returns P-wave velocity

Return type vp

`gardner`(*units='metric'*)

`lam_mu_rho`(*lam*, *mu*, *rho*)

`PyFWI.rock_physics.`**`reverse_Han`**(*vp*, *vs*, *a1=5.5*, *a2=6.9*, *a3=2.2*, *b1=3.4*, *b2=4.7*, *b3=1.8*)

`PyFWI.rock_physics.`**`voigt_berie`**(*k_l*, *rho_l*, *k_g*, *rho_g*, *s_g*)

2.9 PyFWI.seiplot module

`PyFWI.seiplot.`**`earth_model`**(*model*, *keys=[]*, *offset=None*, *depth=None*, ***kwargs*)

earth_model show the earth model.

This function is provided to show the earth models.

Parameters

- **model** (*Dictionary*) – A dictionary containing the earth model.
- **keys** (*list, optional*) – List of parameters you want to show. Defaults to [].

Returns The figure class to which the images are added for furthur settings like im.set-clim().

Return type fig (class)

`PyFWI.seiplot.`**`gn_plot`**(*p*, *grad*, *nz*, *nx*)

`PyFWI.seiplot.`**`seismic_section`**(*ax*, *data*, *x_axis=None*, *t_axis=None*, *aspect_preserving=False*, ***kargs*)

2.10 PyFWI.seismic_io module

PyFWI.seismic_io.**load_mat**(*path*)

This function load python dictionary as a .mat file.

Parameters **path** (*String*) – The path to save the data.

PyFWI.seismic_io.**load_pkl**(*file_path*)

load_pkl loads pkl file.

load_pkl loads pkl file.

Parameters **file_path** (*string*) – Path of file to be loaded.

Returns Loaded file.

Return type output

PyFWI.seismic_io.**read_segy**(*path*)

A function to load segy file.

Parameters **path** – The path of segy file.

Returns The data stored in segy.

Return type data

PyFWI.seismic_io.**save_mat**(*path*, ***kwargs*)

This function save python dictionary as a .mat file.

Parameters

- **path** (*String*) – The path to save the data.
- **unique** (*Boolean*) – If true, it will add current date and time to the name of folder
- ****kwargs** (*type*) – Dictionaries containing the data.

PyFWI.seismic_io.**save_pkl**(*path*, ***kwargs*)

save_pkl saves pkl file.

save_pkl saves file with pkl format. That is better than .mat file for preserving the structure of dictionaries.

Parameters

- **path** (*string*) – path to save the file(s).
- ****kwargs** (*data*) – Variable(s) to be saved.
- **the** (*A boolean argument with name of "unique" can be given to make*) –
- **data.** (*path based on the*) –

2.11 PyFWI.wave_propagation module

2.12 Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

R

S

V

Y