GeoNDT Python library

Release "0.1.0"

Hongwei Liu, Pooneh Maghoul, Guillaume Mantelet, Ahmed Shala

Jun 11, 2021
# CONTENTS

1 Overview ............................................. 1
   1.1 Quick start .................................... 1
   1.2 Troubleshooting .................................. 2
   1.3 References ...................................... 2

2 Installation ......................................... 3
   2.1 PyPI .............................................. 3
   2.2 GitHub .......................................... 3

3 Usage examples ..................................... 5
   3.1 Bender Element Test ............................. 5
   3.2 Ultrasonic pulse velocity ....................... 5
   3.3 Falling Weight Deflectometer ................... 6
   3.4 Multichannel Analysis of Surface Waves ....... 6

4 References .......................................... 7
   4.1 one_phase_dynamic ............................... 7
   4.2 one_phase_dispersion ............................ 8
   4.3 two_phase_dynamic ............................... 8
   4.4 two_phase_dispersion ............................ 9
   4.5 three_phase_dynamic ............................. 10
   4.6 three_phase_dispersion .......................... 11
   4.7 functions inside dynamic modules .............. 12

5 Authors ............................................. 15

6 Indices and tables ................................. 17
GeoNDT is a fast general-purpose computational tool for geotechnical non-destructive testing applications. GeoNDT is flexible, general-purpose, and can be used seamlessly for advanced signal interpretation in geophysical laboratory testing including the bender element (BE) and ultrasonic pulse velocity (UPV) tests, characterization of complex multiphase geomaterials, in-situ shallow seismic geophysics including the falling weight deflectometer (FWD) and multi-channel analysis of surface waves (MASW) tests. The advanced physics-based signal interpretation feature of GeoNDT allows the quantitative characterization of geophysical and geomechanical properties of geomaterials and multilayered geosystems independently without making any simplified assumptions as common in the current practice.

1.1 Quick start

Install (only for Linux):

```
pip install geondt
```

To install development version, clone this repo and install in Linux:

```
git clone https://github.com/siglab/geondt
cd geondt
pip install -e
```

To install development version, clone this repo and install in Windows:

```
git clone https://github.com/siglab/geondt
cd geondt
python setup.py build --compiler=mingw32
python setup.py install
```
1.1.1 Usage

The GeoNDT can efficiently study the three-dimensional wave propagation within soil specimens in the BE test. Sample code is given as follows:

```python
>>> import numpy as np
>>> from geondt import one_phase_dynamic
>>> import json
>>> with open('BE_dry.json', "r") as f:
    data = json.load(f)
>>> BE = one_phase_dynamic(**data["input"])
>>> signal = BE.run_f()
```

1.2 Troubleshooting

The installation procedure assumes that the Fortran compiler such as Gfortran and Lapack library are installed on your system. To install Gfortran and Lapack in Linux:

```
sudo apt install gfortran
sudo apt-get install liblapacke-dev checkinstall
export gfortran="/home/kay/gcc-4.8.5/bin/gfortran"
```

To install Gfortran and Lapack in Windows:

* Use MinGW <https://sourceforge.net/projects/mingw-w64/> to get Gfortran. Make sure the MinGW is added to the system path.
* Then add the liblapack.a file (can be found under lib folder in this respiratory) in the MinGW folder (C:\mingw64\x86_64-w64-mingw32\lib).

For more information, please refer to the documentation: https://geondt.readthedocs.io/en/latest/.

1.3 References
This package can be installed using either PyPI or GitHub:

### 2.1 PyPI

This is the most straightforward option! From the command line in Linux system use `pip`:

```
pip install geondt
```

The PyPI method currently supports for the installation in Linux system. For Windows users, GeoNDT can be installed from the source code. The package has been tested with the following setups (others might work, too):

- Windows (64 bit Python) and Linux (64 bit)
- Python 3.7, 3.8, 3.9

### 2.2 GitHub

If you are intending on modifying geondt modules, it’s easier to install geondt by forking the repository and installing it locally or within a virtual environment. After cloning the repo, you may install in Linux by:

```
git clone https://github.com/siglab/geondt

cd geondt

pip install -e
```

To install development version, clone this repo and install in Windows:

```
git clone https://github.com/siglab/geondt

cd geondt

python setup.py build --compiler=mingw32

python setup.py install
```

The install procedure assumes that the Fortran compiler such as Gfortran and Lapack library are installed on your system. To install Gfortran and Lapack in Linux:
sudo apt install gfortran
sudo apt-get install liblapacke-dev checkinstall
export gfortran="/home/kay/gcc-4.8.5/bin/gfortran"

To install Gfortran and Lapack in Windows:

* Use MinGW <https://sourceforge.net/projects/mingw-w64/> to get Gfortran.
* Then add the liblapack.a file (can be found under lib folder in this respiratory) in the MinGW folder (C:\mingw64\x86_64-w64-mingw32\lib).
3.1 Bender Element Test

The GeoNDT can efficiently study the three-dimensional wave propagation within soil specimens in the Bender element (BE) test. In this example, the soil sample is assumed to be 7.0 cm in diameter and 14 cm in height. The density of the dry sand is 1,800 kg/m$^3$. The P-wave and S-wave velocity of the soil sample is assumed as 380 m/s and 240 m/s, respectively. The code example for BE modeling is shown as follows:

```python
>>> import numpy as np
>>> from geondt import one_phase_dynamic
>>> tlin = 200; tmin = 3*10**(-4); tmax = 150*10**(-5)
>>> vp = np.array([380.0])
>>> vs = np.array([240.0])
>>> rho = np.array([1800])
>>> E = rho1*vs**2*(3*vp**2-4*vs**2)/(vp**2-vs**2)
>>> mu = (vp**2-2*vs**2)/(2*(vp**2-vs**2))
>>> H = [140/1000]
>>> rmax = 70/2/1000
>>> BE = one_phase_dynamic(tmin, tmax, tlin, E, mu, rho, H, rmax/100, rmax, 1, 3, 0, 50)
>>> yt = BE.run_f()
```

3.2 Ultrasonic pulse velocity

Ultrasonic pulse velocity (UPV) test is commonly used for anomaly detection and strength evaluation of construction materials (e.g., concrete and steel). For the demonstration purpose, the example code for the pile integrity test is given as follows:

```python
>>> import numpy as np
>>> from geondt import one_phase_dynamic
>>> def load(s):
...     '''Define external load with 50 kHz in the Laplace domain (for pile-soil interaction example)'''
...     fn1 = -100*10**3*(np.exp(-np.complex(s)/(10*10**3)))*np.pi/(10*10.0**9 *np.pi**2 + 1)
...     fn2 = 100*10**3*(np.exp(-np.complex(s)/(12.5*10**3)))*np.pi/(10*10.0**9 *np.pi**2 + 1)
...     fn = fn1 + fn2
...     return fn
>>> tlin = 500; tmin = 2*10**(-5); tmax = 100*10**(-5)
```

(continues on next page)
>>> vp = np.array([3500.0**2, 3500.0**2, 100.0**2])
>>> vs = np.array([2000.0**2, 2000.0**2, (100/1.5)**2])
>>> rho = np.array([2400, 2400, 1800])
>>> E = rho1*vs*(3*vp-4*vs)/(vp-vs)
>>> mu = (vp**2-2*vs**2)/(2*(vp**2-vs**2))
>>> H = [0.4,0.1,10]
>>> rmax = 0.5
>>> Pile = one_phase_dynamic(tmin, tmax, tlin, E, mu, rho, H, rmax/100, rmax, 4, 2, 0, 50)
>>> Pile.load_i = load
>>> yt = Pile.run_f()

3.3 Falling Weight Deflectometer

Falling weight deflectometer (FWD) is another-in-situ testing method used to evaluate the mechanical properties of pavement structures. The FWD test for a three-layer pavement system can be studied by the GeoNDT:

```python
>>> import numpy as np
>>> from geondt import one_phase_dynamic
>>> tmin = 0.001; tmax=0.05; tlin=200; rmax = 20
>>> E = [1000.0*(10.0**6), 200.0*(10.0**6), 100.0*(10.0**6)] # Young 's modulus in each layer
>>> mu = [0.35, 0.35, 0.35] # Poisson 's ratio in each layer
>>> rho = [2300.0, 2000.0,1500.0] # density in each layer
>>> H1 = [0.15,0.25,5.0] # thickness in each layer
>>> FWD= one_phase_dynamic(tmin, tmax, tlin, E, mu, rho, H1, 0, rmax, 2, 2, 0)
>>> Displacement1 = FWD.run_i()
```

3.4 Multichannel Analysis of Surface Waves

Multichannel Analysis of Surface Waves (MASW) is one of the most popular techniquesfor the in-situ evaluation of the shear wave velocity in different soil layers. The code example for MASW modeling for a three-layer system is shown as follows:

```python
>>> import numpy as np
>>> from geondt import one_phase_dispersion
>>> fmin = 5; fmax = 50; flin = 50
>>> omega = np.linspace(fmin,fmax,flin)*np.pi*2
>>> E = [100.0*(10.0**6), 200.0*(10.0**6), 500.0*(10.0**6)]
>>> mu = [0.2, 0.3, 0.35]
>>> rho = [2300.0, 2000.0,2500.0]
>>> H = [2,3,5.0]
>>> MASW = one_phase_dispersion(fmin,fmax,flin, E, mu, rho, H)
>>> yt = MASW.run()
```
Considering the computational efficiency and user-friendly environment, the GeoNDT software is developed using a dual layer/hybrid Python and Fortran environment to benefit from the strengths of the two languages, as follows: a) Fortran is a compiled language; it is closer to the material architecture it is executed on; it benefits from established mathematical libraries and thus compensates for the lower computational performance of interpreted languages such as Python under CPU intensive tasks; b) Python is a user-friendly language; it has a wide support online; it has a rich set of high-quality scientific computational libraries and frameworks; and it offers improved code reusability, faster and cost-effective development. The detailed discription of the functions in GeoNDT can be found in following reference:

### 4.1 one_phase_dynamic

one_phase_dynamic function is used to study the wave propagation and dynamic response in one-phase material. The inputs of this function is listed as follows:

- **parameter tmin** A mandatory (floating) parameter that defines the minimum time (s)
- **parameter tmax** A mandatory (floating) parameter that defines the maximum time (s)
- **parameter tlin** A mandatory (integer) parameter that defines the number of points within the maximum and minimum time
- **parameter E** A mandatory (floating) list that defines the Young’s modulus (Pa) for all layers
- **parameter mu** A mandatory (floating) list that defines the Poisson’s ratio for all layers
- **parameter rho** A mandatory (floating) list that defines the density (kg/m^3) for all layers
- **parameter H** A mandatory (floating) list that defines the thickness (m) for all layers
- **parameter r** A mandatory (floating) parameter that defines the radius (m) of the location of interest
- **parameter rmax** A mandatory (floating) parameter that defines the maximum radius (m) of the domain
- **parameter node** A mandatory (integer) parameter that specifies the ID of the output node (for instance, the ‘node’ for surface vertical displacement is 2)
- **parameter loc** An optional (integer) parameter that defines the ID of the input load (for instance, the ‘loc’ for surface vertical load is 2)
- **parameter st** An optional (integer) parameter (either 0 or 1) that defined the output type, where 0 represents displacement and 1 represents stress output
- **parameter m_num** An optional (integer) parameter that defines the number of modes in the Fourier-Bessel series
parameter **lap_num** An optional (integer) parameter that defines the number of iteration in the inverse Laplace transform

**parameter a1** An optional (integer) parameter that defines (m) the radius on the contact area between external load and domain

**parameter ncore** An optional (integer) parameter that defines the number of cores (-1 is used in default to use all available cores)

### 4.2 one_phase_dispersion

One-phase dynamic function is used to perform dispersion analysis of one-phase material with the last layer as infinite half space. The inputs of this function is listed as follows:

**parameter f1** A mandatory (floating) parameter that defines the lower bound of frequency (Hz)

**parameter f2** A mandatory (floating) parameter that defines the upper bound frequency (Hz)

**parameter flin** A mandatory (integer) parameter that defines the number of points within the lower and upper bound of frequency

**parameter E** A mandatory (floating) list that defines the Young’s modulus (Pa) for all layers

**parameter mu** A mandatory (floating) list that defines the Poisson’s ratio for all layers

**parameter rho** A mandatory (floating) list that defines the density (kg/m^3) for all layers

**parameter H** A mandatory (floating) list that defines the thickness (m) for all layers

**parameter ncore** An optional (integer) parameter that defines the number of cores (-1 is used in default to use all available cores)

### 4.3 two_phase_dynamic

Object for wave propagation modeling of two-phase material in finite and infinite space. The inputs of this function is listed as follows:

**parameter tmin** A mandatory (floating) parameter that defines the minimum time (s)

**parameter tmax** A mandatory (floating) parameter that defines the maximum time (s)

**parameter tlin** A mandatory (integer) parameter that defines the number of points within the maximum and minimum time

**parameter E** A mandatory (floating) list that defines the Young’s modulus (Pa) of bulk soil for all layers

**parameter mu** A mandatory (floating) list that defines the Poisson’s ratio of bulk soil for all layers

**parameter rho** A mandatory (floating) list that defines the density (kg/m^3) of bulk soil for all layers

**parameter H** A mandatory (floating) list that defines the thickness (m) for all layers

**parameter kh** A mandatory (floating) list that defines the permeability coefficient (m^2) of bulk soil for all layers
parameter **porosity**  A mandatory (floating) list that defines the porosity of bulk soil for all layers

parameter **r**  A mandatory (floating) parameter that defines the radius (m) of the location of interest

parameter **rmax**  A mandatory (floating) parameter that defines the maximum radius (m) of the domain

parameter **node**  A mandatory (integer) parameter that specifies the ID of the output node (for instance, the ‘node’ for surface vertical displacement is 2)

parameter **loc**  An optional (integer) parameter that defines the ID of the input load (for instance, the ‘loc’ for surface vertical load is 2)

parameter **st**  An optional (integer) parameter (either 0 or 1) that defined the output type, where 0 represents displacement and 1 represents stress output

parameter **vis**  An optional (floating) parameter that defines the viscosity (Pa-s) of water

parameter **kf**  An optional (floating) parameter that defines the bulk modulus (Pa) of the fluid

parameter **m_num**  An optional (integer) parameter that defines the number of modes in the Fourier-Bessel series

parameter **lap_num**  An optional (integer) parameter that defines the number of iteration in the inverse Laplace transform

parameter **a1**  An optional (integer) parameter that defines the radius (m) of the contact area between external load and domain

### 4.4 **two_phase_dispersion**

Object for one-phase material in infinite space dispersion modeling. The inputs of this function is listed as follows:

parameter **f1** A mandatory (floating) parameter that defines the lower bound of frequency (Hz)

parameter **f2** A mandatory (floating) parameter that defines the upper bound frequency (Hz)

parameter **flin** A mandatory (integer) parameter that defines the number of points within the lower and upper bound of frequency

parameter **E** A mandatory (floating) list that defines the Young’s modulus (Pa) of bulk soil for all layers

parameter **mu** A mandatory (floating) list that defines the Poisson’s ratio of bulk soil for all layers

parameter **rho** A mandatory (floating) list that defines the density (kg/m^3) of bulk soil for all layers

parameter **H** A mandatory (floating) list that defines the thickness (m) for all layers

parameter **kh** A mandatory (floating) list that defines the permeability coefficient (m^2) of bulk soil for all layers

parameter **porosity** A mandatory (floating) list that defines the porosity of bulk soil for all layers

parameter **vis** An optional (floating) parameter that defines the viscosity (Pa-s) of water

parameter **kf** An optional (floating) parameter that defines the bulk modulus (Pa) of the fluid
4.5 three_phase_dynamic

Object for wave propagation modeling of three-phase material in finite and infinite space. The inputs of this function is listed as follows:

- **parameter ncore**  An optional (integer) parameter that defines the number of cores (-1 is used in default to use all available cores)

4.5 three_phase_dynamic

Object for wave propagation modeling of three-phase material in finite and infinite space. The inputs of this function is listed as follows:

- **parameter tmin** A mandatory (floating) parameter that defines the minimum time (s)
- **parameter tmax** A mandatory (floating) parameter that defines the maximum time (s)
- **parameter tlin** A mandatory (integer) parameter that defines the number of points within the maximum and minimum time
- **parameter kks** A mandatory (floating) list that defines the bulk modulus of soil skeleton (Pa) for all layers
- **parameter muus** A mandatory (floating) list that defines the shear modulus of soil skeleton (Pa) for all layers
- **parameter rho** A mandatory (floating) list that defines the density of soil skeleton (kg/m^3) for all layers
- **parameter phiw** A mandatory (floating) list that defines the volumetric water content for all layers
- **parameter phiwi** A mandatory (integer) parameter that defines the volumetric ice content for all layers
- **parameter H** A mandatory (floating) list that defines the thickness (m) for all layers
- **parameter r** A mandatory (floating) parameter that defines the radius of the location of interest
- **parameter rmax** A mandatory (floating) parameter that defines the maximum radius of the domain
- **parameter node** A mandatory (integer) parameter that specifies the ID of the output node (for surface vertical displacement, it is 2)
- **parameter loc** A mandatory (integer) parameter that defines the ID of the input load
- **parameter st** A mandatory (integer) parameter (either 0 or 1) that defined the output type where 0 represents displacement and 1 represents stress
- **parameter KK_i** An optional (floating) parameter that defines the bulk modulus of ice (Pa)
- **parameter muui** An optional (floating) parameter that defines the shear modulus of ice (Pa)
- **parameter Kww** An optional (floating) parameter that defines the bulk modulus of water (Pa)
- **parameter kappas** An optional (floating) parameter related to the permeability of solid skeleton
- **parameter kappai** An optional (floating) parameter to the permeability of ice
- **parameter b013** An optional (floating) parameter that defines friction coefficient between the solid skeletal frame and ice matrix
- **parameter m_num** A optional (integer) parameter that defines the number of modes
- **parameter lap_num** A optional (integer) parameter that defines the number of iteration in the inverse Laplace transform
parameter a1  A optional (integer) parameter that defines the radius (m) of the contact area between external load and domain

parameter ncore  A optional (integer) parameter that defines the number of cores (-1 is used in default to use all available cores)

4.6 three_phase_dispersion

Object for one-phase material in infinite space dispersion modeling. The inputs of this function is listed as follows:

parameter f1  A mandatory (floating) parameter that defines the lower bound of frequency (Hz)

parameter f2  A mandatory (floating) parameter that defines the upper bound frequency (Hz)

parameter flin  A mandatory (integer) parameter that defines the number of points within the lower and upper bound of frequency

parameter kks  A mandatory (floating) list that defines the bulk modulus of solid skeleton (Pa) for all layers

parameter muus  A mandatory (floating) list that defines the shear modulus of solid skeleton (Pa) for all layers

parameter rho  A mandatory (floating) list that defines the density of solid skeleton (kg/m^3) for all layers

parameter phiw  A mandatory (floating) list that defines the volumetric water content for all layers

parameter phi  A mandatory (floating) list that defines the volumetric ice content for all layers

parameter H  A mandatory (floating) list that defines the thickness (m) for all layers

parameter KKi  An optional (floating) parameter that defines the bulk modulus of ice (Pa)

parameter muui  An optional (floating) parameter that defines the shear modulus of ice (Pa)

parameter Kww  An optional (floating) parameter that defines the bulk modulus of water (Pa)

parameter kappas  An optional (floating) parameter related to the peameability of solid skeleton

parameter kappai  An optional (floating) parameter to the peameability of ice

parameter b013  An optional (floating) parameter that defines friction coefficient between the solid skeletal frame and ice matrix

parameter ncore  An optional (integer) parameter that defines the number of cores (-1 is used in default to use all available cores)
4.7 functions inside dynamic modules

model_f define the dynamic model for finite domain. For instance:

```python
def model_f(self, s):
    ''' Define one-phase model for finite domain'''
    km = jn_zeros(0, self.m_num)/self.rmax
    fm = np.ones(self.m_num, dtype = float)  # point load
    fn = self.external_load(s)
    yt = PoroSEM.one_phase_finite(np.complex(s), self.E, self.mu, self.rho, self.H, km, fm, self.r, np.complex(fn), self.node, self.loc, self.st, self.m_num, len(self.E))
    return yt
```

model_i define the dynamic model for infinite domain. For instance:

```python
def model_i(self, s):
    ''' Define one-phase model for infinite domain'''
    km = jn_zeros(0, self.m_num)/self.rmax
    fm = np.ones(self.m_num, dtype = float)  # point load
    for i in range(self.m_num):
        fm[i] = 2*5**2*pi*km[i]*jv(1,km[i]*self.a1)/(self.rmax**2*km[i]*jv(1,self.rmax)**2) #*(m_num + 1-i)/(m_num+1)*1.05*1.02
    fn = self.external_load2(s)
    yt = PoroSEM.one_phase_infinite(np.complex(s), self.E, self.mu, self.rho, self.H, km, fm, self.r, np.complex(fn), self.node, self.loc, self.st, self.m_num, len(self.E))
    return yt
```

In above function, the external load is required to be defined in the Lapalce domain. For example, the 10 kHz load in the BE test can be defined as:

```python
def external_load(self, s):
    ''' Define external load in the Lapalce domain (for BE example)'''
    fn1 = -20*10**3*(np.exp(-np.complex(s)/(2000)))*np.pi/(400*10.0**6*np.pi**2 + np.complex(s)**2)
    fn2 = 20*10.0**3*(np.exp(-np.complex(s)/(2500)))*np.pi/(400*10.0**6*np.pi**2 + np.complex(s)**2)
    fn = fn1 + fn2
    return fn
```

To run the dynamic response in the finite domain, users should use ‘run_f’ function, defined as:

```python
def run_f(self):
    ''' Run the model using the Parallel computing for finite domain'''
    t = np.linspace(self.tmin, self.tmax, self.tlin)
    yt = Parallel(n_jobs=self.ncore)(delayed(self.inv_f)(i) for i in range(len(t)))
    return np.real(yt)
```

To run the dynamic response in the infinite domain, users should use ‘run_i’ function, defined as:

```python
def run_i(self):
    ''' Run the model using the Parallel computing for infinite domain'''
    t = np.linspace(self.tmin, self.tmax, self.tlin)
    yt = Parallel(n_jobs=self.ncore)(delayed(self.inv_i)(i) for i in range(len(t)))
    return np.real(yt)
```
For dispersion analysis, users can call ‘run’ function directly, defined as follows:

```python
def run(self):
    ''' Run the model using the Parallel computing for infinite domain'''
    omega = np.linspace(self.f1, self.f2, self.flin)*np.pi*2
    yt = Parallel(n_jobs=self.ncore)(delayed(self.final)(i) for i in range(len(omega)))
    return np.real(yt)
```

This python wrapper is made by:

- Hongwei Liu
- Pooneh Maghoul
- Guillaume Mantelet
- Ahmed Shalaby


INDICES AND TABLES

- genindex
- modindex
- search