
asva

adc21

Oct 05, 2021

CONTENTS:

- 1 Quick Start** **3**
- 1.1 Requirements 3
- 1.2 Installation 3
- 1.3 Minimum code example 3

- 2 General Indices** **11**

Important: It is **very important** to check accuracy of asva analysis on your own for production use since asva is under development. Please report bugs to [Github Issues](#).

asva is a package to simulate vibration response of multi degree of freedom system subjected to earthquakes. Response time history and amplitude can be calculated.

QUICK START

1.1 Requirements

Python 3.8+

1.2 Installation

```
pip install asva
```

1.3 Minimum code example

```
import asva as ap

config: ap.AnalysisConfigType = {
    # analysis
    'BETA': 1 / 4,

    # case
    'CASES': [
        {
            'NAME': 'Example',
            'WAVE': 'Sample',
            'AMP': 1,
            'DAMPER': 'None',
            'NDIV': 2,
            'START_TIME': 0,
            'END_TIME': None,
        },
    ],

    # damper
    'DAMPERS': {
        'None': [
            [],
        ],
    },
},

# model
```

(continues on next page)

(continued from previous page)

```

'BASE_ISOLATION': False,
'H': 0.02,
'H_TYPE': 0,
'I': [
    [1],
],
'MI': [100],
'KI': [
    [{
        'type': 'elastic',
        'k0': 4000,
    }],
],

# wave
'WAVES': {
    'Sample': {
        'NAME': 'Sample',
        'DT': 0.02,
        'NDATA': 2688,
        'TO_METER': 0.01,
        'INPUT_FILE': 'wave/Sample.csv',
        'DELIMITER': None,
        'SKIPROWS': 3,
        'COL': 0,
        'ENCODING': 'utf',
    },
},
}

def main():
    analysis = ap.Analysis(config, 0) #
    analysis.analysis()
    print(analysis.resp.dis)

if __name__ == '__main__':
    main()

```

1.3.1 Setup Config

Config dict must be provided to Analysis class in asva. asva provides types to validate the dict. Types are defined in `Types`. You can use them as shown below if needed.

```

import asva as ap

analysis_config: ap.AnalysisConfigType = {
    <your config>
}

# optional
amp_config: ap.AmplitudeConfigType = {
    <your config>
}

```

(continues on next page)

(continued from previous page)

```

# optional
export_config: ap.ExportConfigType = {
    <your config>
}

analysis = ap.Analysis(analysis_config, 0, amp_config, export_config)

```

Analysis Config

```

class AnalysisConfigType(TypedDict):
    # analysis
    BETA: float          # Newmark
    BASE_ISOLATION: bool # 1 (C10)

    # wave
    WAVES: Dict[str, WaveType] #

    # case
    CASES: List[CASESType] #

    # model
    H: float            #
    H_TYPE: Literal[0, 1] # 0: 1:
    I: List[List[float]] # NDOF×11
    MI: List[float]      # [ton]
    KI: List[KIType]     # [kN/m]

    # damper
    DAMPERS: Dict[str, List[List[DamperType]]]

```

Amplitude Config

```

class AmplitudeConfigType(TypedDict):
    N_W: int          #
    DF: float         # [Hz]

```

Export Config

```

class ExportConfigType(TypedDict):
    RESULT_DIR: str #
    RESULT_DATA_DIR_NAME: str #
    RESULT_PLOT_DIR_NAME: str #
    DATA_PLOT_STORIES: Optional[List[int]] # ( or None)

```

1.3.2 Hysteretic Models

Hysteretic models can be defined and set to `AnalysisConfig` like below.

```
# Example
import asva as ap

config: ap.AnalysisConfigType = {
    ...,
    'KI': [
        [ # first storey
            { # first hysteresis
                'type': 'elastic',
                'k0': 4000,
            },
            { # second hysteresis
                'type': 'elastic',
                'k0': 4000,
            },
        ],
        [ # second storey
            {
                'type': 'elastic',
                'k0': 4000,
            },
        ],
    ],
    ...,
},
    ...,
}
```

Elastic

```
class ElasticType(TypedDict):
    type: Literal["elastic"]
    k0: float # [kN/m]
```

Bilinear

```
class BilinearType(TypedDict):
    type: Literal["bilinear"]
    k0: float # [kN/m]
    a1: float # [-]
    f1: float # [kN]
```

Trilinear, Gyakko, Takeda

```
class TrilinearType(TypedDict):
    type: Literal["gyakko", "takeda", "trilinear"]
    k0: float          # [kN/m]
    a1: float          # 1[-]
    a2: float          # 2[-]
    f1: float          # 1[kN]
    f2: float          # 2[kN]
```

1.3.3 Dampers

Dampers can be defined and set to AnalysisConfig like below.

You can register several dampers in config and choose it in CASES.

```
# Example
import asva as ap

Oil: ap.VDBType = {
    'c1': 100,
    'c2': 50,
    'vr': 0.75,
    'vel_max': 1.5,
}

config: ap.AnalysisConfigType = {
    ...,
    'CASES': [
        {
            'DAMPER': 'VDB_DAMPERS',
            ...,
        },
    ],
    ...,
    'DAMPERS': {
        'VDB_DAMPERS': [
            [
                {
                    'type': 'VDB',
                    'Nd': 1,
                    'd': Oil,
                },
            ],
        ],
    },
    ...,
}
```

MASS Damper

type MASS

```
class MASSType (TypedDict):  
    m: float
```

Stopper

type Stopper

```
class StopperType (TypedDict):  
    k: float  
    ft: float
```

Viscous Damper (CV[^])

type VDA

```
class VDAType (TypedDict):  
    cd: float  
    alpha: float  
    vy: Optional[float]  
    vel_max: Optional[float]
```

Viscous Damper (Bilinear)

type VDB

```
class VDBType (TypedDict):  
    c1: float  
    c2: float  
    vr: float  
    vel_max: float
```

TMD

type TMD

```
class TMDType (TypedDict):  
    md: float  
    cd: float  
    kd: float
```

iRDT

type iRDT

```
class iRDType(TypedDict):  
    md: float  
    cd: float  
    alpha: float  
    kb: float  
    fr: float  
    cosA: float
```


GENERAL INDICES

- [genindex](#)
- [modindex](#)
- [search](#)