Podstawy j. Python/Numpy (1)

- Język Python jest językiem interpretowanym, podobnie jak np. Matlab/Octave.
- W j. Python spacje mają duże znaczenie.
- Instrukcje pętli `for`, `while`, warunków `if` muszą zaczynać się po wcięciu (np. cztery spacje)
- Znak rozpoczynający komentarz to `#`. Wówczas reszta lini nie jest traktowana jako kod j. Python.
- Matlab używa indeksowania zaczynając od jedynki (1), natomiast Python/Numpy zaczyna indeksowanie od zera (0)
- Dla spójności Python/Numpy kończy indeksowanie na wartości `nk - 1`.
- Na przykład w j. Python `range(5)` daje liczby 0, 1, 2, 3, 4. Czyli pięć liczb.
- Dla spójności tablice, wektory, listy mają swój pierwszy element, wiersz o indeksie 0 (w Matlabie to indeks 1).
- Ostatni wiersz, element tablicy lub listy ma indeks -1 (w Matlabie odpowiada to indeksowi `end`)
- W Python/Numpy zasadniczym typem jest tablica wielowymiarowa `ndarray`
- Do elementów tablic odwołujemy się przez nawias kwadratowy, np. `A[0, 0]` (w Matlabie używamy nawiasu okrągłego `A(1,1)`).
- Mnożenie tablic w sensie macierzowym wymaga operatora `@` (w Matlabe `*`). Mnożenie element przez element odbywa się przez operator `*` (w Matlabie `.*`)

Podstawy j. Python/Numpy (2)

- Więcej informacji pod adresem: Numpy for Matlab Users
- MATLAB pozwala spersonalizować środowisko poprzez modyfikację ścieżek przeszukiwania (`Set path`)
- NumPy, a raczej Python, posiada podobne możliwości. Można dodawać lub modyfikować katalogi ścieżki dostępu do modułów poprzez definicję zmiennej środowiskowej `PYTHONPATH`
- Często jednak jest to niepotrzebne jeśli instalujemy moduły przez system anaconda: komenda `conda` lub za pomocą komendy `pip` , wówczas ścieżka dostępu do modułów jest automatycznie uaktualniona
- W przeciwieństwie do Matlaba, gdzie cokolwiek dodana do ścieżki może być od razu używane, w Pythonie trzeba najpierw użyć stwierdzenia `import` , aby określone funkcje były dostępne w bieżącym pliku `skrypt.py` .
- Przykład użycia najczęstszych instrukcji `import` do obliczeń inżynierskich (Numpy, Scipy), wykresów (Matplotlib) i programu OpenSeesPy znajduje się dwa slajdy dalej.

Matlab vs Python/Numpy

Matlab	Python/Numpy
`size(a)`	`a.shape`
`a(end)`	`a[-1]`
`a(1:5,:)`	`a[0:5]` , `a[:5]`
`flipupd(a)`	`a[::-1,:]`
`a*b`	`a@b`
`a.*b`	`a*b`
`a./b`	`a/b`
`a.^3`	`a**3`
`a.'`	`np.transpose(a)`
`b=a`	`b=np.copy(a)`
`y=x(:)`	`y=x.flatten()`
`1:10`	`arange(1.,11.)`
`0:10`	`arange(10.)`
`[1:10]'`	`arange(1.,11.[:,newaxis])`
`zeros(3,4)`	`np.zeros((3,4))`

Matlab	Python/Numpy
`rand(3,4)`	`random.rand(3,4)`
`linspace(1,3,4)`	`np.linspace(1,3,4)`
`repmat(a,m,n)`	`tile(a, (m,n))`
`[a, b]`	`concat((a,b),1), hstack((a,b))`
`[a; b]`	`concat((a,b)), vstack((a,b))`
`max(max(a)), max(a)`	`a.max(), a.max(0)`
`norm(v)`	`sqrt(v@v), np.linalg.norm(v)`
`inv(a)`	`linalg.inv(a)`
`a \ b`	`linalg.solve(a)`
`u,s,v = svd(a)`	`u,s,v = linalg.svd(a)`
`v,d = eig(a,b)`	`sp.linalg.eig(a,b)`
`numel(a)`	`ndim(a), a.ndim`
	`size(a), a.size`
`unique(a)`	`unique(a)`
`squeeze(a)`	`a.squeeze`

Matlab vs Python/Numpy

Matlab	Python/Numpy
`fft(a)`	`fft(a)`
`regress(y,X)`	`linalg.lstsq(X,y)`

Matlab	Python/Numpy
`decimate(x,q)`	`sp.signal.resample(x,len(x))`

Do komend Python/Numpy dodaj przedrostek `np.` , np. `np.arange` i `linalg.xxx` lub `np.linalg.xxx`

Początek każdego `pliku.py` Python/Numpy/Scipy powinien zaczynać się następująco:

```
import numpy as np
from numpy import sin, cos, exp, sqrt, atan
import scipy as sp
import matplotlib.pyplot as plt
import openseespy.opensees as ops
...
```

i wówczas dalej możemy używać komend poszczególnych bibliotek jak na przykład

```
ops.model('basic', '-ndm', 3, '-ndf', 6)
ops.node(1, 0., 0., 0.)
...
x = np.zeros((3,4))  # macierz 3x4 wypelniona zerami
v, d = sp.linalg.eig(M, K)  # rozwiazanie zag. wlasnego

plt.plot(x, y, 'b--')
```