MIDAS SQUARE

MIDAS SQUARE 공학 기술강연

공학분야 MATLAB 사용자를 위한 Python 기초와 활용

김두기 | 공주대학교

CONTENTS

1. MATLAB

1.1 개요　　1.2 What & Why?

1.1 개요

유래

- 1984년 MathWorks사(미국 보스턴)에서 개발한 공학용 S/W
- 스크립트 프로그래밍 언어 겸 공학용 S/W(애플리케이션)
- MAT는 수학(MAThematics)이 아니라 행렬(MATrix)에서 유래
- MATLAB 로고는 Membrane의 Eigenmode(모드형상) 중 하나

정체성

- 프로그래밍에 입문하는 공대생들이 제일 먼저 배우는 프로그래밍 언어
- 빛(편리)과 어둠(불편)이 공존하는 프로그래밍 언어 겸 애플리케이션

특징

- 매우 간단하고 편리한 문법
  → 프로그래밍 입문자가 (비체계적으로) 쉽고 빠르게 습득
- 대학 졸업 후 MATLAB 라이선스 비용 발생으로 오픈소스 대체품 필요
  → GNU Octave, Python 등
- 수준 높고 다양한 MATLAB Toolbox

1.2 What & Why?

Matrix Laboratory

- Dynamically typed language
  - ✓ Variables require no declaration
  - ✓ Creation by initialization (x=10;)
- All variables are treated as matrices
  - ✓ Scalar: 1×1 matrix

    Vector: N×1 or 1×N matrix
  - ✓ Calculations are much faster

Advantages

- Fast implementation and debugging
- Natural matrix operation
- Powerful toolbox

2. Python

2.1 개요　　2.2 MATLAB vs Python

2.1 개요

유래

1991년 발표된 스크립트 프로그래밍 언어
(개발자: 네덜란드 Guido van Rossum)

특징

- 오픈 소스
- 강력한 라이브러리와 풍부한 생태계 보유
- 문법이 쉽고 활용성이 높음

공학분야에서 MATLAB과 전쟁 중인 Python

- 64비트를 넘는 매우 큰 정수 지원
- 허수를 기본적으로 지원
- 표준 라이브러리의 decimal, fractions 모듈로 소수점과 유리수의 정밀 연산 가능
- 강력하고 무료인 Package와 도구
  → NumPy, SciPy, Matplotlib, Jupyter Notebook 등
- 진화 중인 Python
  → Ellipsis (...) 상수, 행렬곱 연산자 @ 등

2.2 MATLAB vs Python

MATLAB이 Python보다 좋은 이유

구분	MATLAB	Python
1. 행렬과 배열	함수 호출이 불필요	함수 호출이 필요

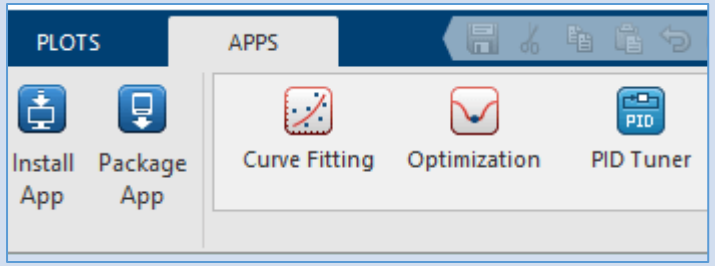
```
>> row = [1, 2, 3]
row =

   1   2   3

>> col = row'
col =

   1
   2
   3

>> inner = row*col
inner = 14
>> outer = col*row
outer =

   1   2   3
   2   4   6
   3   6   9
```

```
In [1]: import numpy as np
   ...: row = np.array([1, 2, 3]).reshape(1,-1)

In [2]: row
Out[2]: array([[1, 2, 3]])

In [3]: col = row.T

In [4]: col
Out[4]:
array([[1],
       [2],
       [3]])

In [5]: inner = np.dot(row, col)

In [6]: inner
Out[6]: array([[14]])

In [7]: outer = np.dot(col, row)

In [8]: outer
Out[8]:
array([[1, 2, 3],
       [2, 4, 6],
       [3, 6, 9]])
```

(참조: MATLAB&Simulink 공식 네이버 블러그, 2017.9.25)

2.2 MATLAB vs Python

MATLAB이 Python보다 좋은 이유 (계속)

구분	MATLAB	Python
2. 개발 환경	통합개발환경(IDE) 제공	신뢰성이 결여된 개발 환경
3. 추가 기능	전문가들이 개발하고 검증한 Toolbox 제공	커뮤니티 등에서 개발한 Package 제공
4. 매크로 기능	독립 애플리케이션인 앱(App) 기능 제공 	앱 기능 없음
5. 호환성	H/W, 코드 변환, 시스템 통합 등 호환성 용이	호환성/일관성 부족
6. 실행 속도	빠름(JIT 컴파일(1), 병렬 프로그래밍 등)	느림
7. 결과 신뢰성	Closed S/W로 높음	Open S/W로 낮음

(1) JIT 컴파일(just-in-time compilation)

(참조: MATLAB&Simulink 공식 네이버 블러그, 2017.9.25)

2.2 MATLAB vs Python

제목 설명글을 작성하세요.

구분	MATLAB	Python
1. 태생	Mathworks 소유 closed 소스	오픈 소스
2. 언어	기본 데이터 구조로써 행렬과 배열을 제공	list, tuple, set 등 다양한 데이터 구조 제공
3. 도구	Simulink, Toolbox 등 전용 IDE	Numpy, Scipy, Matplotlib 등 Visual Studio, PyCharm, Jupyter, Spyder 등
4. 사용자	주로 공대생들/연구원들	다양한 분야 많은 사용자
5. 비용	유료	무료

- Python은 무료이므로, 더 나은 선택
- Python이 Data Science, Machine Learning, S/W 개발 및 프로그래밍에 더 적합
- MATLAB을 배우고, 처음부터 Python을 배워야 한다고 해도
  MATLAB을 고수하는 것보다는 Python을 추가로 배우는 것이 더 나음

(참조: Natalie Schooner 블로그 https://blog.boot.dev/, 2023.4.16)

2.2 MATLAB vs Python

소고(小考)

공학분야 파이썬(Python)은

계산할 때 계산기(Calculator)보다 불편하지만,

그래프 그릴 때 엑셀(EXCEL)보다 불편하지만,

행렬을 연산할 때 매트랩(MATLAB)보다 불편하지만,

프로그래밍 언어로서 확장가능성이 훨씬 큰

파이썬(Python)은 사용할수록 참 좋은

프로그래밍 언어입니다.

3. 설치와 실행

3.1 설치　　3.2 실행

3.1 설치

Python 코딩을 위해 (1) Python 또는 (2) Anaconda 중에 하나를 설치한다.

(1) Python 표준 배포본 (설치 비추천)

- Python은 www.python.org에서 관리하며 "표준 배포본"을 제공하고 있다.
- 표준 배포본에는 실제 Python 사용 시 필요한 다양한 "패키지(package; 라이브러리)"가 설치되어 있지 않으므로 사용 시 필요한 패키지를 일일이 설치하여야 한다.

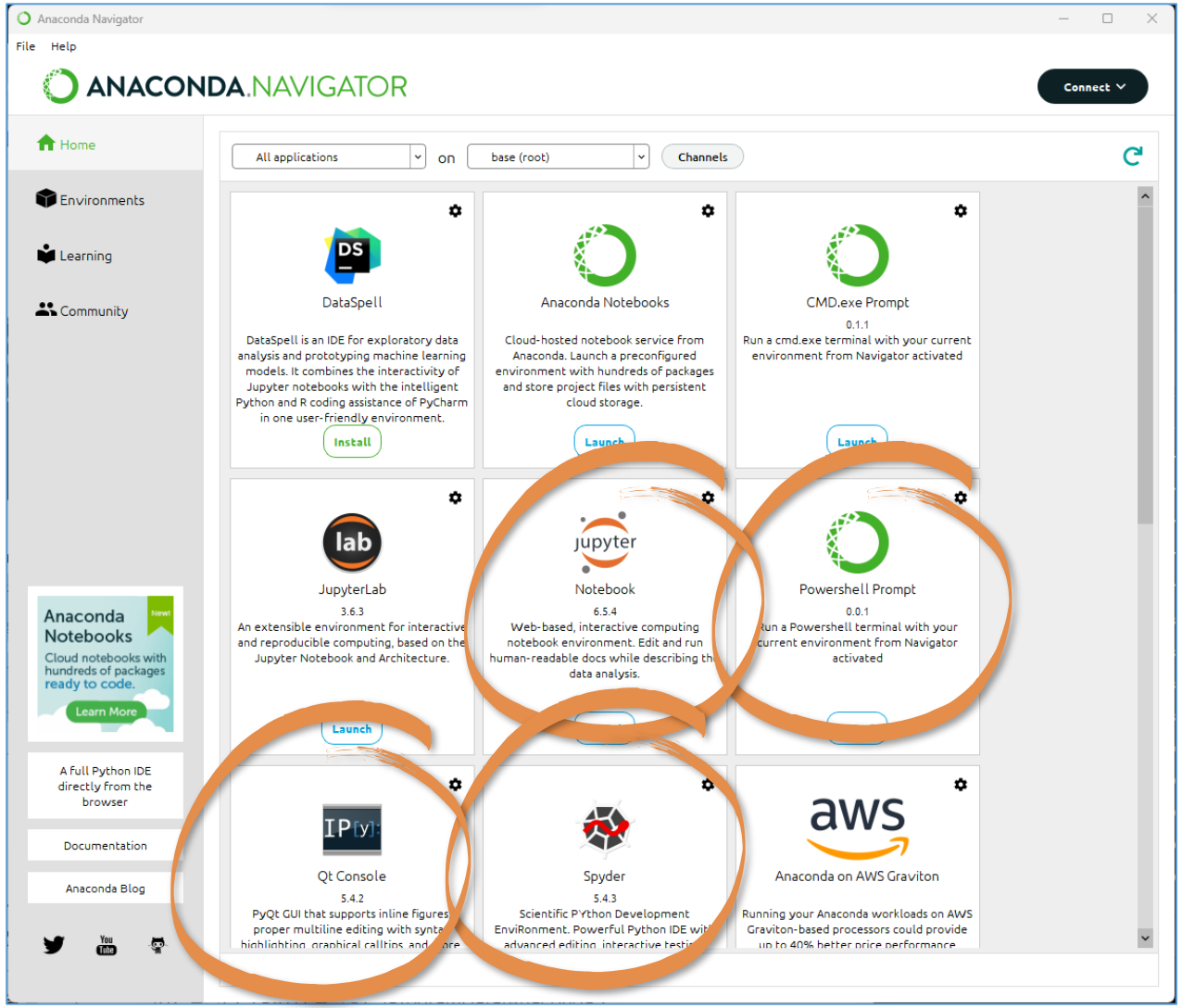
(2) Anaconda 배포본 (설치 추천)

- 대부분 사용자는 Anaconda, Canopi 등과 같이 필수 패키지가 포함된 "배포본(distribution)"를 많이 사용한다. 여기서 Python 표준 배포본 대신에 Anaconda(www.anaconda.com) 배포본을 설치한다.
- Anaconda에는 NumPy, matplotlib 등 필수 패키지를 포함하고 있으며, 통합개발환경(IDE)인 Spyder, Jupyter Notebook, Powershell Prompt 등을 함께 제공하기 때문에 사용하기 편리하다.

3.2 실행

Anaconda

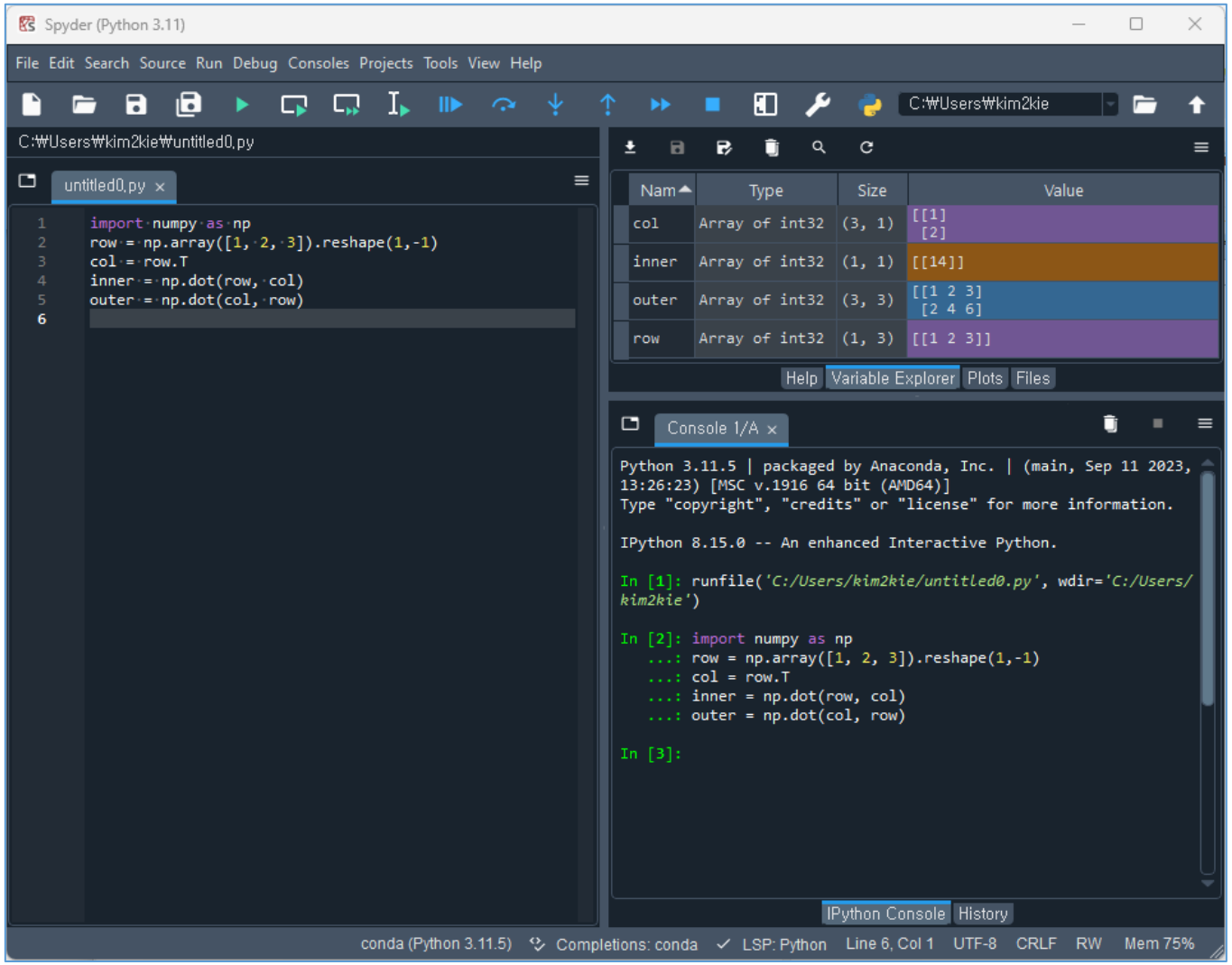
Spyder, Jupyter Notebook, Powershell Prompt, Qt Console



3.2 실행

Spyder

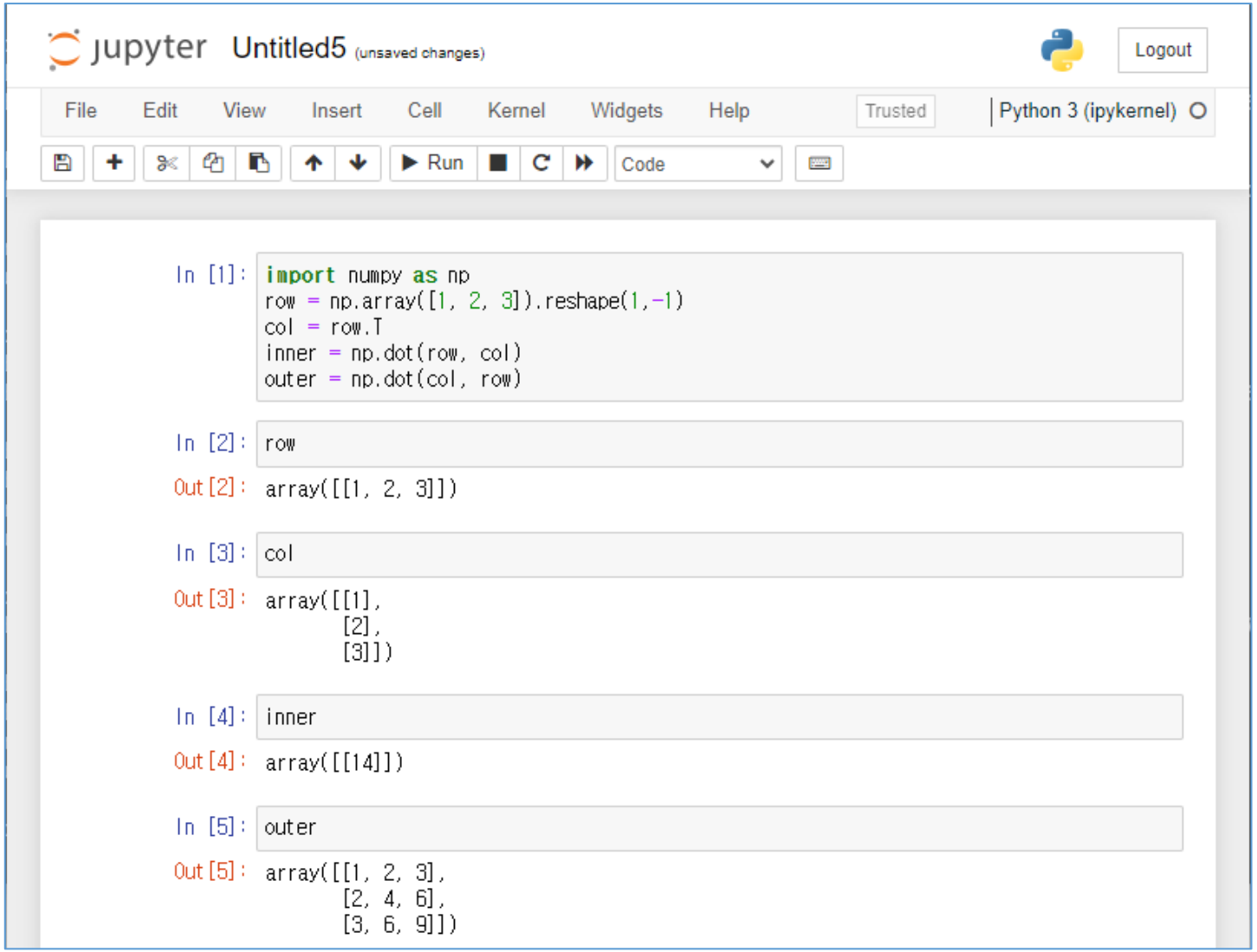
과학기술계산용 Crossplaform IDE로, IPython(Interactive Python, 대화형 파이썬) 포함



3.2 실행

Jupyter Notebook

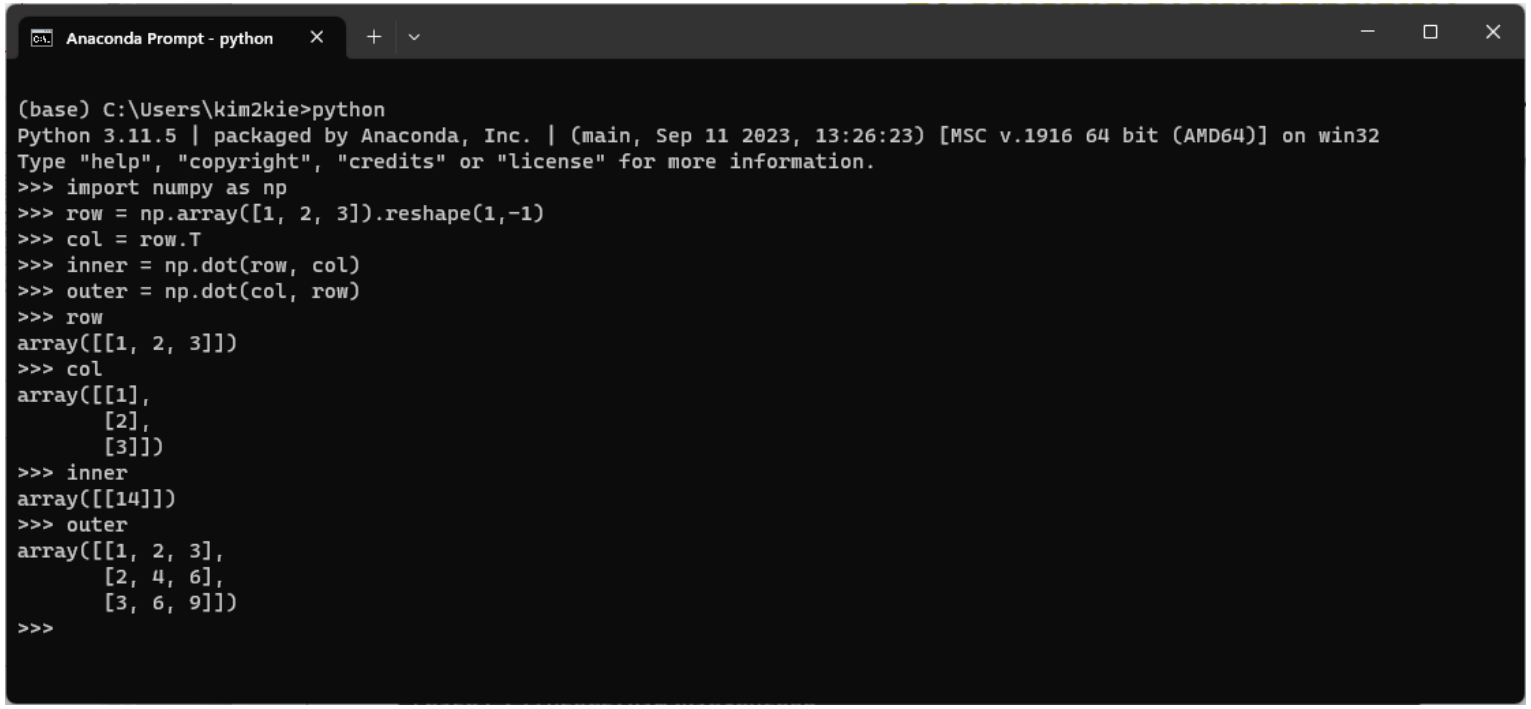
웹상에서 Python 코드를 작성하고 그 결과를 (문자 수식으로도) 바로 확인가능



3.2 실행

Powershell Prompt

간단한 계산을 위해 사용



Online (임시 사용 추천)

1) Replit(전체 실행): https://replit.com/languages/python3

2) PythonAnywhere(line by line 실행): https://www.pythonanywhere.com/try-ipython/

4. 기초 문법

4.1 기본　　4.2 비교

4.1 기본

패키지

약어	패키지	설명
linalg	import scipy.linalg as linalg import numpy.linalg as linalg	linalg는 선형대수(linear algebra)를 위한 라이브러리이다.
m	import math as m	C 표준에서 정의된 수학(mathematics) 함수에 대한 액세스를 제공한다. 단순 python 연산을 넘어 조금 더 복잡한 산술 연산이 필요할 때 사용한다.
mp	import mpmath as mp	
np	import numpy as np	NumPy는 과학 컴퓨팅을 위한 기본 패키지이다. 다차원 배열 외에도 이산 푸리에 변환, 선형대수, 통계, 난수(random number) 등 다양한 기능을 가지고 있다.
opt	import scipy.optimize as opt	SciPy는 과학기술계산을 위한 라이브러리이다.
pd	import pandas as pd	Pandas는 데이터 분석 라이브러리 중 하나로, 데이터 처리, 분석, 시각화 등을 위한 다양한 기능을 제공한다.
PyQt		(C++용으로 개발된) Qt의 레이아웃으로 파이썬 GUI 프로그램을 만드는 framework이다.
plt	import matplotlib.pyplot as plt	Matplotlib은 그래프를 그리는 패키지이다.
random	import random	
signal	import scipy.signal as signal	SciPy는 과학기술계산을 위한 라이브러리이다.
sp	import scipy as sp	SciPy는 과학기술계산을 위한 라이브러리이다.
sy	import sympy as sy	SymPy는 변수를 심볼로 지정한 문자 연산이 가능하다.
tf	import tensorflow as tf	

4.1 기본

파일

MATLAB 파일		PYTHON 파일	
*.m *.mat *.mdl *.fig	: 매트랩 구문을 저장한 파일 : workspace를 저장한 파일 : simulink를 저장한 파일 : GUI를 저장한 파일	*.py	: 파이썬 구문을 저장한 파일

기본 명령어

MATLAB 기본 명령어		PYTHON 기본 명령어
help, doc	: 도움말	help()
clear, clc, clf	: 변수와 화면 삭제	del(), os.system('cls'), plt.clf()
lookfor	: 키워드 검색	help()
dir, cd, delete, is, pwd	: 디렉터리, 파일 관련 명령어	os.listdir, os.chdir, os.remove, is, os.getcwd()
who, whos	: 변수 출력	dir(), type()
!	: 언어 외 shell 명령어(예: dir) 실행	! 예) !dir
;	: 명령 결과를 출력하지 않는 끝맺음	;
…	: 줄 바꾸기 연속 문자	\
simulink	: simulink 실행	패키지
Guide → App Designer	: GUI 실행	PyQt, TkInter 패키지
deploytool	: Exe 파일 작성을 위한 compiler 실행	Pyinstaller 패키지

4.1 기본

숫자

숫자	MATLAB	PYTHON

x = 3.14
z = 2 + 3j
	x = real(z); % 실수부 반환
	y = imag(z); % 허수부 반환
	abs_z = abs(z); % 절댓값 계산
	conj_z = conj(z); % 켤레복소수 계산
		x = z.real # 실수부 반환
		y = z.imag # 허수부 반환
		abs_z = abs(z) # 절댓값 계산
		conj_z = z.conjugate() # 켤레복소수 계산

행렬

MATLAB 행렬		PYTHON 행렬
>> A=[1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 >> r = [11 12 13] r = 11 12 13	>> A1=[A; r] A1 = 1 2 3 4 5 6 7 8 9 11 12 13 >> A2 = [A1(1:2,:)] A2 = 1 2 3 4 5 6 >> size(A1) ans = 4 3 >> size(A2) ans = 2 3	import numpy as np A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) r = np.array([11, 12, 13]) A1 = np.vstack([A, r]) A2 = A1[:2, :] np.shape(A1) np.shape(A2)

4.1 기본

행렬

행렬	MATLAB	PYTHON
전치행렬(transpose)	`A_transpose = A';`	`A_transpose = A.T`
역행렬(inverse)	`A_inv = inv(A);`	`A_inv = np.linalg.inv(A)`
덧셈(addition)	`C = A + B;`	`C = A + B`
스칼라곱(dot product)	`C = dot(A, B);`	`C = np.dot(A, B)`
벡터곱(cross product)	`C = cross(A, B);`	`C = np.cross(A, B)`
대각성분(diagonal)	`D = diag(v);`	`D = np.diag(v)`
원소연산(element operation)	`D = A .* B;` `E = A .^ 2;`	`D = A * B` `E = A ** 2`

증분

증분	MATLAB	PYTHON
1씩 증가(예: x = 1 2 3)	`x = 1:3`	`x = np.arange(1, 4)`
0.5씩 증가(예: x = 1.0 1.5 2.0 2.5 3.0)	`x = 1:0.5:3`	`x = np.arange(1, 3.5, 0.5)`

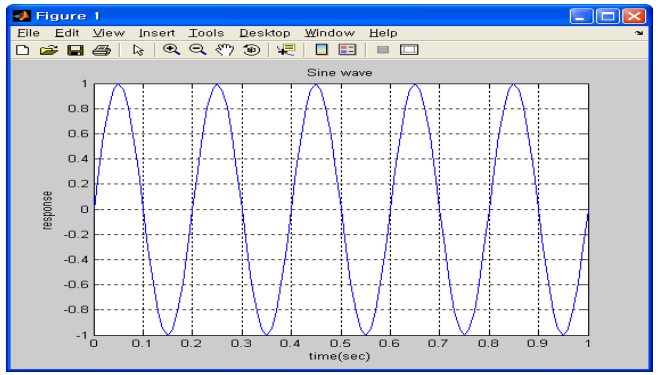
파이썬에서 콜론(:)은 슬라이싱(slicing)을 위해 사용되지만,
다른 언어처럼 직접 범위를 만들 때는 사용되지 않는다.

파이썬에서 arange은 NumPy 패키지에서 있는 범위(array)를 생성하는 함수

4.1 기본

그래픽

그래픽	MATLAB	PYTHON

	t=0:0.01:1;
	y=sin(2*pi*5*t);
	plot(t,y)
	grid
	xlabel('time(sec)')
	ylabel('response')
	title('Sine wave')
		t = np.arange(0, 1.01, 0.01)
		y = np.sin(2 * np.pi * 5 * t)
		plt.plot(t, y)
		plt.grid()
		plt.xlabel('time(sec)')
		plt.ylabel('response')
		plt.title('Sine wave')

Symbolic 연산

Symbolic 연산	MATLAB	PYTHON
$au^2 + bu + c = 0$ $\rightarrow u = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	syms a b c u [u] = solve(a*u^2 + b*u + c == 0,u)	import sympy as sp a, b, c, x = sp.symbols('a b c x') eqn = a*x**2 + b*x + c sol = sp.solve(eqn, x) print("The solutions are:", sol)
$u = \int_{\sin t}^{1} 2x dx$ $\rightarrow u = \cos^2 t$	syms x t u u = int(2*x, [sin(t), 1])	import sympy as sp x, t, u = sp.symbols('x t u') u = sp.integrate(2*x, (x, sp.sin(t), 1))

4.1 기본

함수 정의

MATLAB 함수 정의

```
function [add, minus]
= addminus(a, b)
add = a + b;
minus = a - b;
```

function의 이름과 M-파일명을 동일하게 한다. 다음 예제의 경우, 파일명은 addminus.m이다.

```
>> a=3;
>> b=2;
>> [r_add,r_minus] =
 addminus(a,b)
r_add =   5
r_minus =   1
```

PYTHON 함수(매서드) 정의

```
def addminus(a, b):
    add = a + b
    minus = a - b
    return add, minus
```

```
>>> a = 3
>>> b = 2
>>> r_add, r_minus =
addminus(a, b)
```

4.2 비교

설명문

설명문(주석)	MATLAB	PYTHON
	% comment (행의 끝까지)	# comment (행의 끝까지)

연산자

산술연산자	MATLAB	PYTHON
덧셈	+	+
뺄셈	-	-
곱셈	* 및 .*	*
나눗셈	/ 및 ./	/
지수	^ 및 .^	**
나머지		%
증가분		i += 1
감소분		i −= 1
그룹 구분	( )	( )

콜론연산자	MATLAB/PYTHON
기본 형식	B:I:E(1)
≥B	B:
≤E	:E
전체 영역	: 또는 ...(생략 기호)(2)
마지막 인덱스	-1(3)

관계 및 논리 연산자	MATLAB	PYTHON
Equal to	==	==
Not equal to	~=	!=
Less than	<	<
Less than or equal to	<=	<=
Greater than	>	>
Greater than or equal to	>=	>=
Object identity		is
Negated object identity		is not
Logical NOT	~	not
Logical AND	&&	and
Logical inclusive OR	||	or
Logical exclusive OR	xor	^
Logical equivalent	==	==
Logical not equivalent	~=	!=

(1) B = 시작값(beginning), E = 끝값(ending), I = 증분값(increment)
(2) 생략 기호(ELLIPSIS): 배열의 모두를 나타낼 때 Python에서는 '...'를 사용할 수 있다.
(3) 배열의 마지막 인덱스: Python에서는 '-1'를 사용할 수 있다.

4.2 비교

변수

변수	MATLAB	PYTHON
	실수/복소수 문자	(1) 정수/실수/복소수 (2) 문자열:“”, 리스트:[], 튜플:() (3) dictionary: {:, …} (4) set: {} (5) Boolean: True, False (6) Binary: bytes, bytearray, memoryview

반복문

반복문(1)	MATLAB	PYTHON
유한 반복문	for k=1:n end	for i in range(n):
무한 반복문	while end	while True:
반복문 종료 및 탈출	break	break
반복문 1회 건너뜀	-	continue
메시지 출력 후 멈춤	error	except
호출함수로 환원	return	return

조건부 반복문	initialize test
	while l_express
	  true group
	  change test
	end
		initialize test
		while True:
		  key = input()
		  print(key)

반복문(2)	MATLAB	PYTHON

색인 반복문	for index=matrix
	  statements
	end
		for i in range(10):
		
		xdata = [0.1,4,3]
		for x in xdata:

Pretest 반복문	while (test) statements end	items = [9,5,4,10] for idx, val in enumerate(items):
Posttest 반복문		

4.2 비교

조건문

조건문	MATLAB	PYTHON

조건부 실행
 	
조건부 교체
 	
조건부 선택
	if
	end
	else
	elseif
	switch/case 문이 없으므로, if 문 사용
		if a > 0:
		
		else:
		elif:
		switch/case 문이 없으므로, if 문 사용

논리형 IF문	if l_express true group end	if (l_express):

논리형
내부 IF문
(nested)
	if l_express1
	  true group A
	  if l_express2
	  true group B
	  end
	  true group C
	end
	statement group D
		if l_express1:
		  true group A
		  if l_express2:
		  true group B
		  true group C
		statement group D

논리형
IF-ELSE문
	if l_express
	  true group A
	else
	  true group B
	end
		if l_express:
		  true group A
		else:
		  true group B

논리형
IF-ELSE-IF문
	if l_express1
	  true group A
	elseif l_express2
	  true group B
	else
	  default group C
	end
		if l_express1:
		  true group A
		elif l_express2:
		  true group B
		else:
		  default group C

4.2 비교

배열

배열	MATLAB	PYTHON
1차원 배열 초기화	A(100)=0 for j=1:100 A(j)=12 end 또는 A=12*ones(1,100)	np.ones([3,3]) np.ones([3,3],dtype='int32') np.zeros([3,3]) np.empty([3,3]) mx1 = np.array([[5, 10], [15, 20]])
2차원 배열 초기화	A=ones(10,10)	
배열 초기화 및 할당	A=zeros(2,3) A=[1,7,2; 3,4,6];	

배열	MATLAB	PYTHON

스칼라곱 C=aB	C=a*B	Z = X.copy() # deep copy
		Y = X # shallow copy

역행렬 $B=A^{-1}$	B=inv(A)	import scipy.linalg as linalg Ainv = linalg.inv(A)

배열	MATLAB	PYTHON
덧셈 C=A+B	C=A+B	요소별 계산 C=A+B C=A*B C=A**2 C=A+2 C= 10*np.sin(A) A<3 A *= B A[0:2,]
곱셈 C=AB	C=A*B	행렬 연산 C1=np.dot(A,B) C2=np.matmul(A,B) C3 = A@B A.T np.transpose(A) [D,V] = np.linalg.eig(A)

5. 구조동역학 활용

5.1 운동방정식 풀기　　5.2 고유치 해석

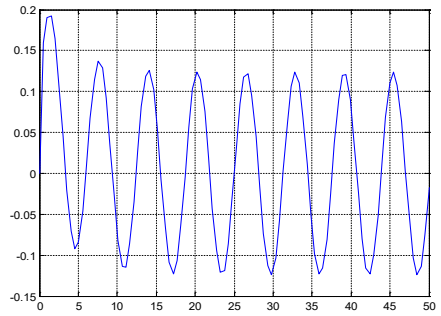
5.1 운동방정식 풀기

2계 미분방정식

Symbolic 연산

Symbolic 연산

$$\ddot{u}(t) + 8\dot{u}(t) + 2u(t) = \cos t,$$
$$u(0) = 0, \dot{u}(0) = 1$$
$$\rightarrow u(t) = \frac{1}{65}\cos t + \frac{8}{65}\sin t +$$
$$\frac{1}{1820}\left(-14 + 53\sqrt{14}\right)e^{\left(-4+\sqrt{14}\right)t} +$$
$$\frac{1}{1820}\left(-14 - 53\sqrt{14}\right)e^{\left(-4-\sqrt{14}\right)t}$$



MATLAB

```
syms u(t) t;
eqn2 = diff(u,t,2) +
8*diff(u,t) + 2*u == cos(t);
inits2 = [u(0) == 0, diff(u)(0) == 1];
uSol(t) = simplify(dsolve(eqn2, inits2));

uSol_fn =
matlabFunction(uSol);  %
Convert symbolic function to
function handle
fplot(uSol_fn, [0, 100]);
xlabel('t');
ylabel('u(t)');
title('Solution of ODE');
grid on;
```

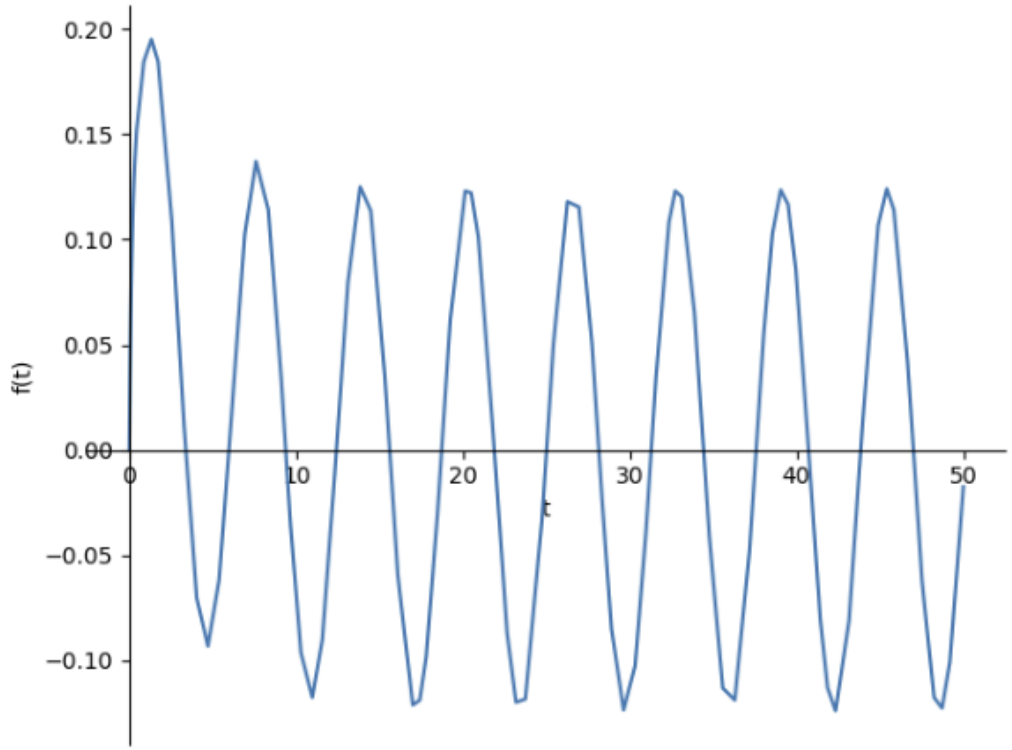
PYTHON

```
from sympy import *

t = symbols('t')
u = symbols('u', cls=Function)
deq = Eq(u(t).diff(t,2) +
8*u(t).diff(t) + 2*u(t),cos(t))
# 방정식 세우기
usol = dsolve(deq,ics={u(0):0,
u(t).diff(t).subs(t,0):1}) #
초기 조건(2개)

plot(usol.rhs,(t,0,50))
```

5.1
운동방정식 풀기

2계 미분방정식 (계속)

In [1]:
```
from sympy import *

t = symbols('t')
u = symbols('u',cls=Function)
deq = Eq(u(t).diff(t,2) + 8*u(t).diff(t) + 2*u(t),cos(t)) # 방정식 세우기
usol = dsolve(deq,ics={u(0):0, u(t).diff(t).subs(t,0):1}) # 초기 조건(2개)

plot(usol.rhs,(t,0,50))
```



Out[1]: <sympy.plotting.plot.Plot at 0x242327a4910>

5.2 고유치 해석

함수: eig() vs eigs()

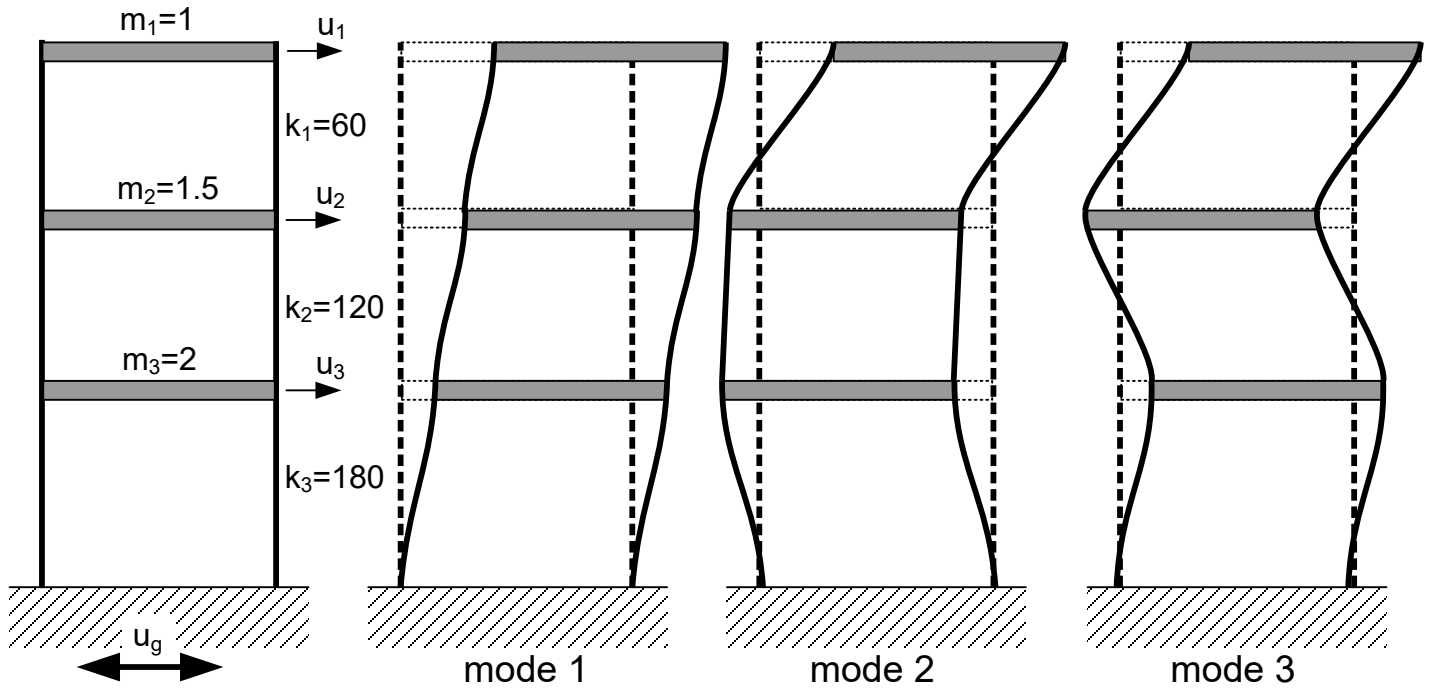
<table>
<tr><th>MATLAB</th><th>PYTHON</th></tr>
<tr><td rowspan="3">[V, D] = eigs(K, M, m, ‘SA’)</td><td>import numpy as np
[D, V] = np.linalg.eig(np.linalg.inv(M)@K)</td></tr>
<tr><td>from scipy.sparse.linalg import eigs
[D, V] = eigs(K, m, M, which='SM')</td></tr>
<tr><td>from scipy.linalg import eig
[D, V] = eig(np.linalg.inv(M)@K)</td></tr>
<tr><td>여기서
V = [Φ̄]: 일반질량이 1인 정규모드
D = [\Ω²\]
K = [K]
M = [M]
m = 구할 고유치 개수(< n)
‘SA’ = 크기가 작은 고유치부터 출력</td><td>여기서
V = [Φ̄]: SRSS가 1인 정규모드
D = {Ω²}
K = [K]
M = [M]
m = 구할 고유치 개수(< n)
‘SM’ = 크기가 작은 고유치부터 출력
[Note]
1. eig 함수는 작은 행렬의 고유치 전부를 구할 때,
eigs 함수는 큰 행렬의 고유치 일부를 구할 때 사용
2. MATLAB에서 D = [\Ω²\]이고, PYTHON에서 D = {Ω²}</td></tr>
</table>

5.2
고유치 해석

예제: 3.1.1 고유치해석 (ch03_eigen)

$$[M] = \begin{bmatrix} 1.0 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 2.0 \end{bmatrix}, \quad [K] = 60 \begin{bmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 5 \end{bmatrix}$$

$$\rightarrow [A] = [M]^{-1}[K] = 60 \begin{bmatrix} 1 & -1 & 0 \\ -2/3 & 2 & -4/3 \\ 0 & -1 & 5/2 \end{bmatrix}$$



MATLAB: 6.4.3 모드중첩법을 사용한 시간이력해석
(ch06_modesuper.m, newmark.m)

5.2 고유치 해석

예제: 3.1.1 고유치해석 (ch03_eigen) (계속)

$$[M]=\begin{bmatrix}1.0 & 0 & 0\\ 0 & 1.5 & 0\\ 0 & 0 & 2.0\end{bmatrix},\quad [K]=60\begin{bmatrix}1 & -1 & 0\\ -1 & 3 & -2\\ 0 & -2 & 5\end{bmatrix}$$

$$[A]=[M]^{-1}[K]=60\begin{bmatrix}1 & -1 & 0\\ -2/3 & 2 & -4/3\\ 0 & -1 & 5/2\end{bmatrix}$$

[ch03_eigen.m]

$$[K][\Phi]=[M][\Phi][\backslash\Omega^2\backslash]$$

```
M=[1.0  0.0  0.0;    0.0  1.5  0.0;    0.0  0.0  2.0];
K=60*  [1   -1   0;    -1    3  -2;    0   -2   5];
[V, D] = eigs(K, M, 3, 'SA');
```

```
D =
21.0879         0         0
      0   96.3959         0
      0         0  212.5162
```

```
V =
 -0.7427   -0.6358    0.2104
 -0.4816    0.3857   -0.5348
 -0.2242    0.4317    0.5132
```

```
V'*M*V =
1.0   0.0   0.0
0.0   1.0   0.0
0.0   0.0   1.0
```

[ch03_eigen.py]

$$[M]^{-1}[K][\Phi]=[\Phi][\backslash\Omega^2\backslash]$$

```
import numpy as np
M = np.array( [[1.0,  0.0,  0.0],[ 0.0,  1.5,  0.0],[0.0,  0.0,  2.0]] )
K = np.array( [[1.0, -1.0,  0.0],[-1.0,  3.0, -2.0],[0.0, -2.0,  5.0]] )*60.0
[D,V] = np.linalg.eig(np.linalg.inv(M)@K)   # D = eigenvalues, V = eigen vectors
```

```
D =
21.0879
96.3959
212.516
```

```
V =
0.813328  -0.739429   0.273045
0.527472   0.448537  -0.694062
0.245503   0.502056   0.666127
```

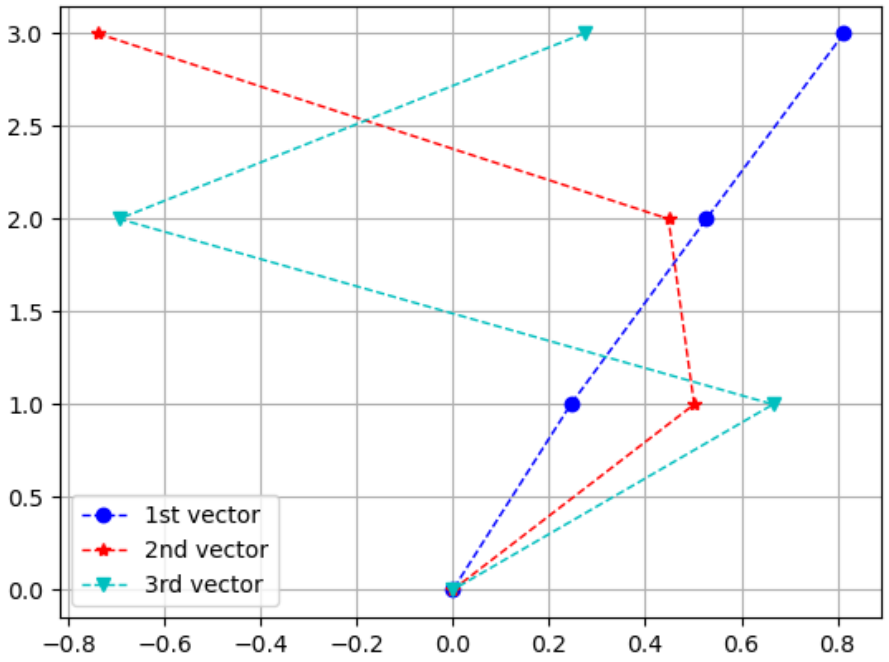
```
NV = np.array([]) # check SRSS of V
for idx in range(len(V)):
    NV1 = np.sqrt(np.sum(V[:,idx].T*V[:,idx]))
    NV = np.append(NV,NV1)
```

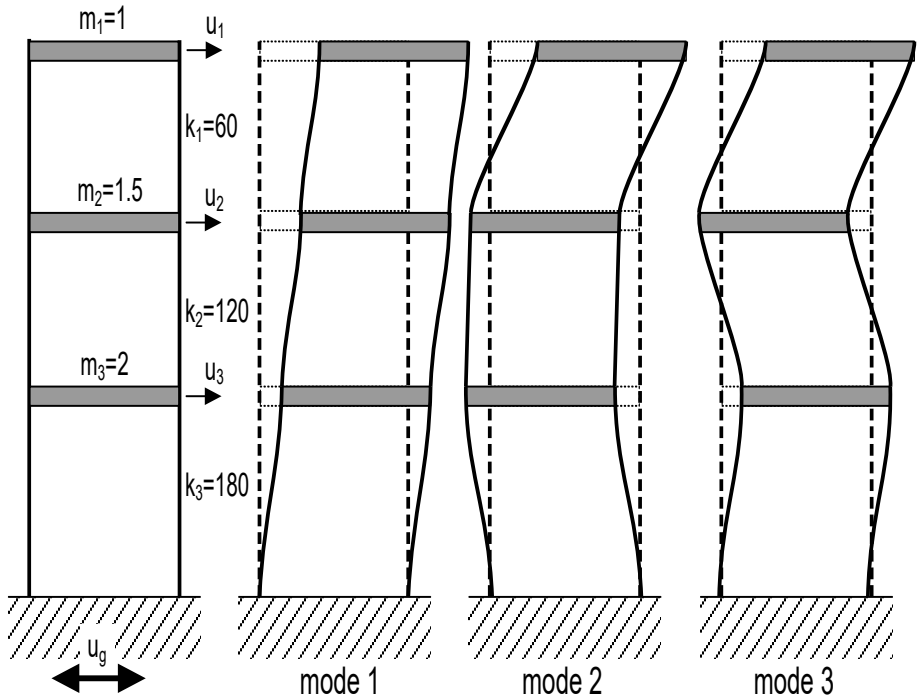
```
NV = [1., 1., 1.]
```

5.2 고유치 해석

예제: 3.1.1 고유치해석 (ch03_eigen) (계속)

```
import matplotlib.pyplot as plt
V0 = np.zeros([1,3]) # (1x3) 크기의 zero array 생성
V1 = np.append(V,V0,axis=0) # 생성된 array를 새로운 행으로 추가
plt.plot(V1[:,0],[3,2,1,0],'b--o',label=r'1st vector',lw=1) # 2D 선그래프
plt.plot(V1[:,1],[3,2,1,0],'r--*',label=r'2nd vector',lw=1) # 2D 선그래프
plt.plot(V1[:,2],[3,2,1,0],'c--v',label=r'3rd vector',lw=1) # 2D 선그래프
plt.grid(True); plt.legend(loc='lower left'); plt.show() # 화면 표시
```

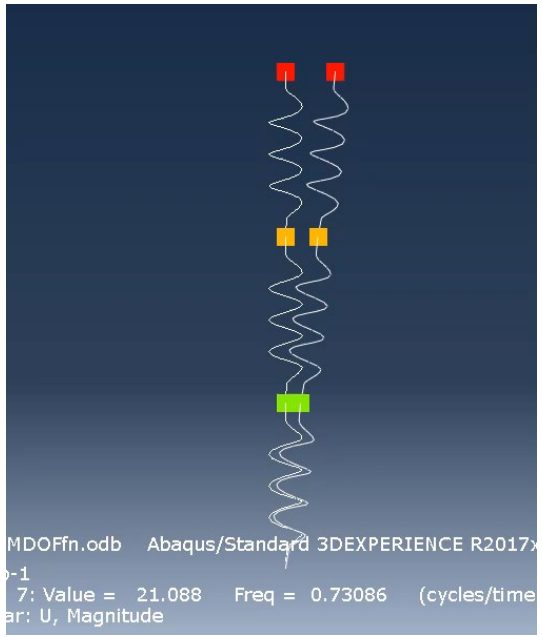


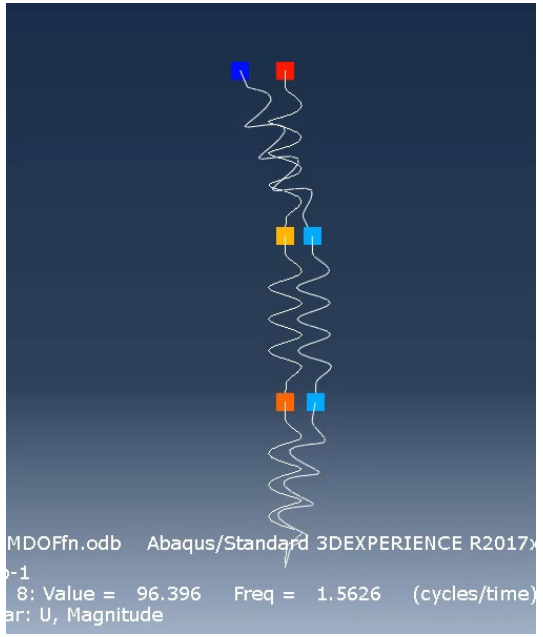


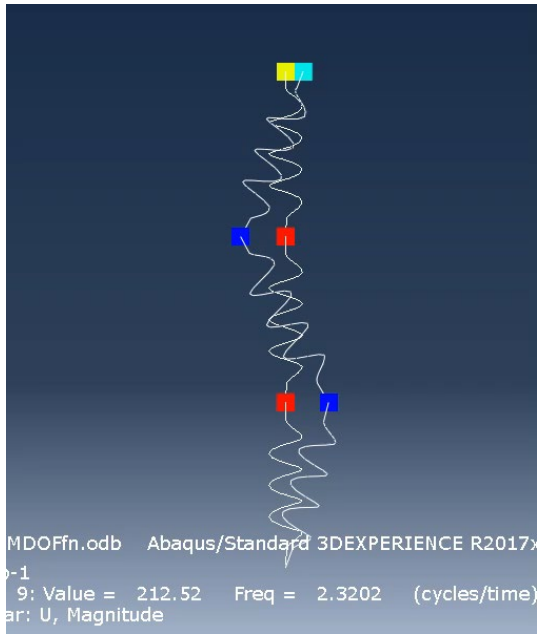
참조) MATLAB: 6.4.3 모드중첩법을 사용한 시간이력해석

5.2
고유치 해석

예제: 3.1.1 고유치해석 (ch03_eigen) (계속)







ABAQUS를 사용한 검증

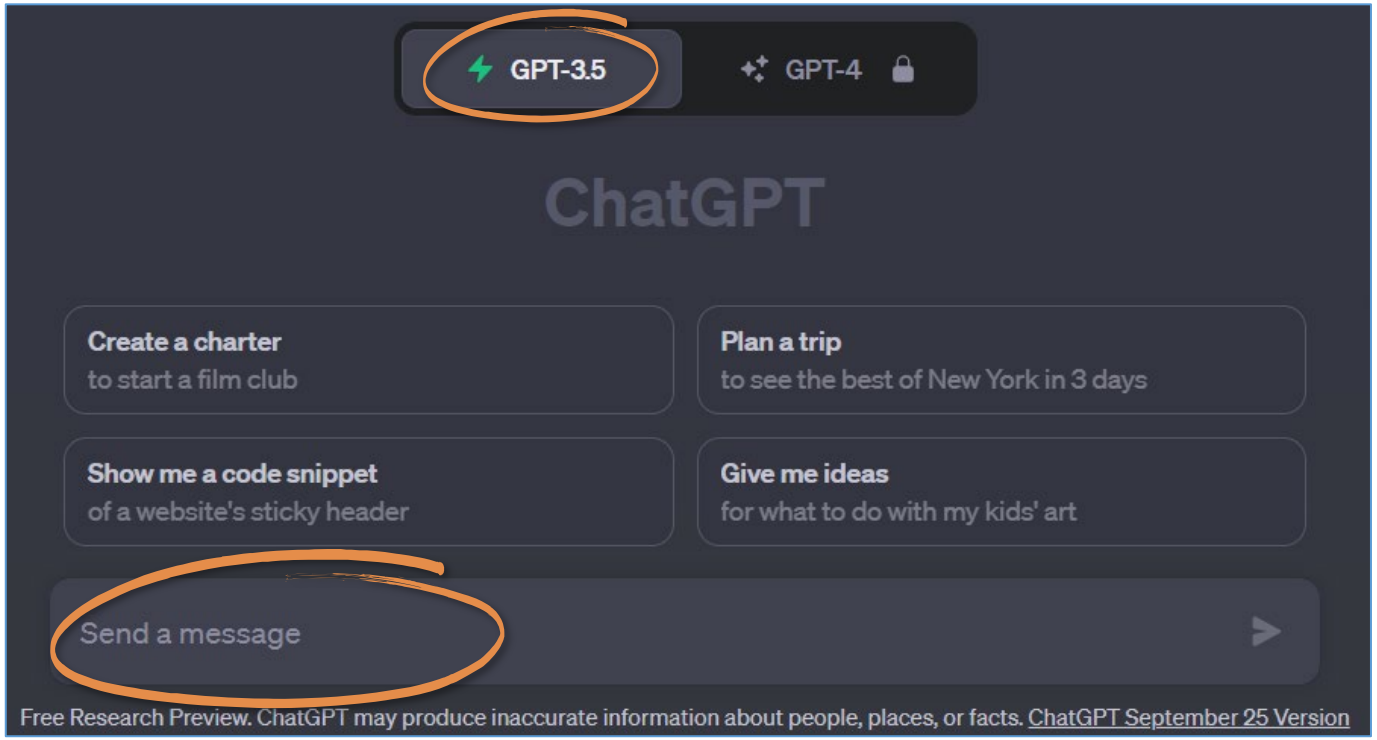
6. ChatGPT

6.1 개요　　6.2 활용

6.1 개요

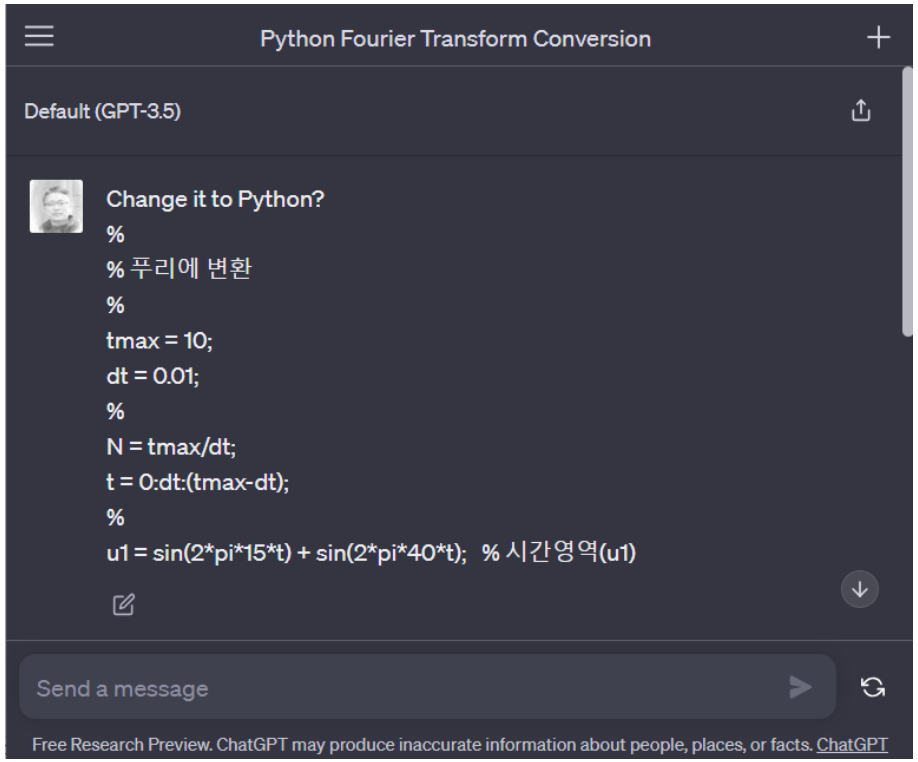
유래

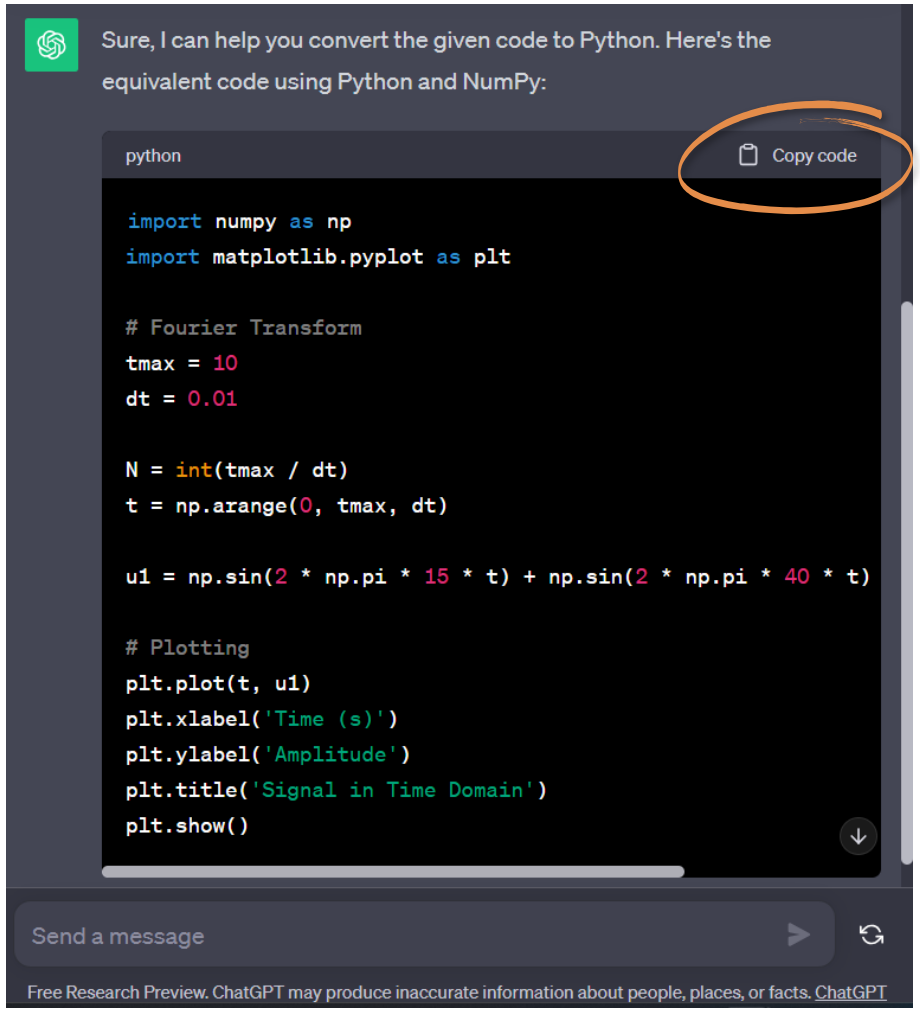
- 미국 OpenAI가 2022년 11월 공개한 대화형 인공지능
- https://chat.openai.com/chat 회원 가입 후 이용
  (초기 버전 GPT-3.5 무료, GPT-4.0 이후 유료)
- GPT는 'Generative Pre-trained Transformer'의 약자
- 사용자가 채팅으로 질문하면, 학습된 ChatGPT는 '사람처럼' 대답



6.2 활용

ch01_fft.m → ch01_fft.py





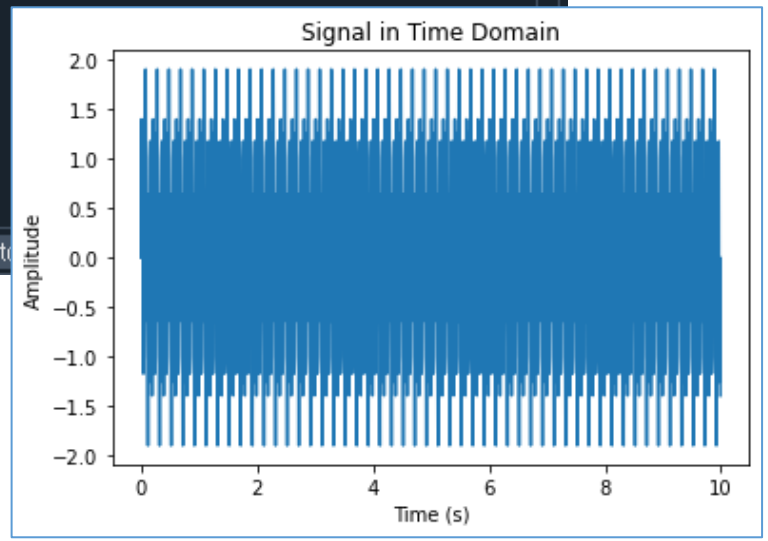
6.2 활용

ch01_fft.m → ch01_fft.py (계속)

Console 1/A

```
Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1916 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: # Fourier Transform
   ...: tmax = 10
   ...: dt = 0.01
   ...:
   ...: N = int(tmax / dt)
   ...: t = np.arange(0, tmax, dt)
   ...:
   ...: u1 = np.sin(2 * np.pi * 15 * t) + np.sin(2 * np.pi * 40 * t)  # Time domain (u1)
   ...:
   ...: # Plotting
   ...: plt.plot(t, u1)
   ...: plt.xlabel('Time (s)')
   ...: plt.ylabel('Amplitude')
   ...: plt.title('Signal in Time Domain')
   ...: plt.show()
```

IPython Console Hist



참조

1. 공주대학교 SSL(www.kim2kie.com)
   → Lecture 파이썬
2. 김두기, 구조동역학, 구미서관,
   2021(5판, MATLAB), 2025(6판, MATLAB/Python 예정)

감사합니다.