Python and Matlab Reference for Linear Algebra and Image Processing

Nithin C Babu

August 6, 2022

Contents

1 Code organisation

1.1 main()

```
#python
def main():
    ...............
    return


if __name__ == '__main__':
    main()
```

```
%matlab
%------
```

2 Basics

2.1 Take Integer Input

```
#python
var = int(input(’Question:’))
```

```
%matlab
var = input(’Question:’)
```

2.2 Output Text and Variable together

```
#python
print(’text ’ + str(var))
```

```
%matlab
disp([’text ’, num2str(var)])
```

2.3 Adding Values Before and After

```
#python
import numpy as np
out_vec = np.concatenate(([0],in_vec,[0]),axis =
   None)
```

```
%matlab
out = [0 in 0]
```

2.4 Generate Random Integer

```
#python
i = random.randint(start_val, end_val)
#both start_val and end_val included
i = np.random.randint(start_val, end_val, (m, n))
#for m-by-n matrix; end_val not included
```

```
%matlab
i = randi([start_val, end_val])
%both start_val and end_val included
i = randi([start_val, end_val], m, n) %for m-by-n
   matrix
```

2.5 Randomly choose N members from list without repetition

```
#python
randList = random.sample(inputList, N)
```

```
%matlab
randList = randsample(inputList, N)
```

2.6 Take Power of a Number

```
#python
sq = a**2
```

```
%matlab
sq = a^2
```

2.7 Rounding

```
#python
A = np.around(A, decimal_num)
#If decimal_num < 0 : round to integer left to
    decimal point
```

```
%matlab
A = round(A, decimal_num)
```

2.8 Pi (π)

```
#python
pie = math.pi
```

```
%matlab
pie = pi
```

2.9 Array Indexing

```
#python
A[i,j]
# 0 <= i,j <= N-1
```

```
%matlab
A(i,j)
% 1 <= i,j <= N
```

2.10 Dimensions (Length) of Matrix

```
#python
length = len(v) #length of list
dim = A.shape # for numpy matrix
```

```
%matlab
len = length(v) %length of vector
dim = size(A)   %dimensions of array
```

2.11 Check for duplicates in a List

```python
#python
dupTrue = any(v.count(element)>1 for element in v)
```

```matlab
%matlab
if length(unique(v)) < length(v)
    dupTrue = true
else
    dupTrue = false
end
%unique(v) creates list without duplicates
```

2.12 Approximate very small values to Zero

```python
#python
A[abs(A) < 1e-4] = 0
```

```matlab
%matlab
A(abs(A) < 1e-4) = 0
```

2.13 Print Output Precision

```python
#python
var = np.array([5.2366871])
np.set_printoptions(precision = 2)
#2 decimal places only
print(var)      #[5.24]
```

```matlab
%matlab
format short
var = [5.2366871]       %5.2367
```

2.14 Calculate processing time

```python
#python
t = time.time()
.................
.................
print(time.time() - t)
```

```matlab
%matlab
tic
.................
.................
toc
```

2.15 Convert Numpy type to List type

```
#python
listVar = v.tolist()
```

```
%matlab
%------
```

2.16 Convert Matrix type to Numpy type

```
#python
A = np.array(M)
```

```
%matlab
%------
```

3 Program Flow

3.1 If Else

```python
#python
if condition1:
        ..............
        ..............
elif condition2:
        ..............
        ..............
else:
        ..............
        ..............
```

```matlab
%matlab
if condition1
        ..............
        ..............
elseif condition2
        ..............
        ..............
else
        ..............
        ..............
end
```

3.2 For Loop

```python
#python
for i in range (start_val, end_val, increment):
        ..............
        ..............
#end_val NOT included; increment optional (default = 1)
```

```matlab
%matlab
for v = start_val:increment:end_val
        ..............
        ..............
end
%end_val IS included; increment optional (default = 1)
```

3.3 Do While

```
#python
while True:
        ...............
        ...............
        if condition:
                break
```

```
%matlab
while 1
        ...............
        ...............
        if condition
                break;
        end
end
```

3.4 Conditional (ternary) Operator

```
#python
val = on_True if condition else on_False
```

```
%matlab
%NO TERNARY OPERATOR
```

4 Matrix Creation

4.1 Create Matrix with Values

```
#python
A = np.np.array([1,2,3],[4,5,6])
```

```
%matlab
A = [1 2 3; 4 5 6]
```

4.2 Initialize Empty and Append

```
#python
v = []
v.append(N)
```

```
%matlab
v = []
v = [v N]
```

4.3 Zero Matrix

```
#python
Z = np.zeros([m,n], dtype = int)
```

```
%matlab
Z = zeros(m,n)
```

4.4 Ones Matrix

```
#python
Z = np.ones([m,n], dtype = int)
```

```
%matlab
Z = ones(m,n)
```

4.5 Constant Number Matrix

```
#python
K = np.full((m,n), N)
```

```
%matlab
K = repmat(N,[m,n])
```

4.6 [2,3,....,9]

```
#python
v = np.arange(2,10)
```

```
%matlab
v = 2:9
```

4.7 [[0,1,2];[3,4,5];[6,7,8]]

```
#python
A = np.arange(9).reshape((3,3))
```

```
%matlab
A = reshape(0:8,3,3)'
```

4.8 Linearly Spaced Vector

```
#python
A = np.linspace(start_num,end_num,num_of_ele)
#both start and end included
```

```
%matlab
y = linspace(start_num,end_num,num_of_ele)
```

4.9 Vector with Constant Increment

```
#python
v = np.arange(start_num,end_num+increment,
   increment)
```

```
%matlab
v = start_num:increment:end_num
```

4.10 Random Matrix

```
#python
A = np.random.rand(m,n)
#only includes uniformly distributed values
   between 0 and 1
```

```
%matlab
A = rand(m,n)
```

4.11 Identity Matrix

```
#python
I = np.eye(n)
```

```
%matlab
I = eye(n)
```

4.12 Diagonal Matrix with Values at k^{th} diagonal

```
#python
D = np.diag(v, k = d_num)
#v is the vector at the diagonal and d_num is the
    position of the diagonal (+ve -> to the right;
     -ve -> to the bottom)
```

```
%matlab
D = diag(v, d_num)
```

4.13 Hilbert Matrix $\left(\frac{1}{1+i+j}\right)$

```
#python
H = scipy.linalg.hilbert(n)
```

```
%matlab
H = hilb(n)
```

4.14 Toeplitz Matrix

```
#python
T = scipy.linalg.toeplitz([1,2,3])
"""
[1  2  3]
[2  1  2]
[3  2  1]
"""
T = scipy.linalg.toeplitz([1,2,3],[1,4,5,6])
"""
[1  4  5  6]
[2  1  4  5]
[3  2  1  4]
"""
T = scipy.linalg.toeplitz([1,2+3j,4-1j])
"""
[1+0j   2-3j    4+1j]
[2+3j   1+0j    2-3j]
[4-1j   2+3j    1+0j]
"""
```

```
%matlab
T = toeplitz([1,2,3])
%       [1  2  3]
%       [2  1  2]
%       [3  2  1]
T = toeplitz([1,2,3],[1,4,5,6])
%       [1  4  5  6]
%       [2  1  4  5]
%       [3  2  1  4]
T = toeplitz([1,2+3j,4-1j])
%       [1+0j   2+3j    4-1j]
%       [2-3j   1+0j    2+3j]
%       [4+1j   2-3j    1+0j]
```

5 Matrix Operations

5.1 Append Row or Column

```
#python
A_r = np.append(A, row, axis=0)
A_c = np.append(A, col, axis=1)
```

```
%matlab
A_r = [A; row]
A_c = [A col]
```

5.2 Matrix Multiplication

```
#python
m = a @ b
```

```
%matlab
m = a * b
```

5.3 Matrix Conjugate, Transpose and Hermitian

```
#python
A_c = A.conj()
A_t = A.T
A_h = A.T.conj()
```

```
%matlab
A_c = conj(A)
A_t = A.’
A_h = A’
```

5.4 Element-wise Power

```
#python
A_N = np.power(A,N)
```

```
%matlab
A_N = A.^N
```

5.5 Matrix Power

```
#python
A_k = np.linalg.matrix_power(A,k)
```

```
%matlab
A_k = A^k
```

5.6 Extract Diagonal Elements

```
#python
v = np.diag(A,k = d_num)
#v is the vector at the diagonal and d_num is the
    position of the diagonal (+ve -> to the right;
     -ve -> to the bottom)
```

```
%matlab
v = diag(A, d_num)
```

6 Linear Algebra Operations

6.1 Determinant

```
#python
d = np.linalg.det(A)
```

```
%matlab
d = det(A)
```

6.2 Inverse

```
#python
A_i = np.linalg.inv(A)
```

```
%matlab
A_i = inv(A)
```

6.3 Eigen Values and Eigen Vectors

```
#python
e_val, e_vect = np.linalg.eig(A)
#eigh ; hermitial or symmetric
#eigvals ; eigen value only
```

```
%matlab
[e_val e_vect] = eig(A)
%e_val = eig(A) ; eigen value only
```

6.4 Null Space

```
#python
ns = scipy.linalg.null_space(A)
#orthonormal basis for null space
```

```
%matlab
ns = null(A)
```

6.5 Row Reduced Echelon Form (rref)

```
#python
A_rref, pivot_col = sympy.matrices.Matrix(A).rref
    ()
#A_rref is of Matrix type, not Numpy type
```

```
%matlab
A_rref = rref(A)
```

6.6 Solve full rank Linear Systems

```
#python
x = np.linalg.solve(A, b)
```

```
%matlab
x = linsolve(A,b)
%or
x = A\b
```

7 Matrix Decomposition and Factorisation

7.1 LU Decomposition

```python
#python
P, L, U = scipy.linalg.lu(A)
#P is the permutation matrix for exchanging zero pivots
```

```matlab
%matlab
[L,U,P] = lu(A)
```

7.2 QR Decomposition

```python
#python
Q, R = np.linalg.qr(A)
```

```matlab
%matlab
[Q,R] = qr(A)
```

7.3 Cholesky Factorisation ($CC^H = A$)

```python
#python
C = np.linalg.cholesky(A)
#A should be conjugate symmetric
```

```matlab
%matlab
C = chol(A)
```

8 Vectorization

8.1 Time test

```
#python
import time
t = time.time()
.....................
t = time.time() - t

# More accurate timer
import timeit
import statistics
t = statistics.median(timeit.Timer(’function_name(arguments)’, setup=setup).repeat())
```

```
%matlab
timer_a = tic
.....................
t = toc(timer_a)

% More accurate timer
f = @() function_name(arguments);
t = timeit(f)
```

8.2 Add constant to matrix

```
#python
A1 = A + c      # c is a scalar constant
```

```
%matlab
A1 = A + c
```

9 Image Basics

9.1 Image Read

```
#python
im = skimage.io.imread(image_path)
```

```
%matlab
im = imread(image_path)
```

9.2 Image Write

```
#python
skimage.io.imsave(output_path, im)
```

```
%matlab
imwrite(im, output_path)
```

9.3 Show Image

```
#python
plt.imshow(im)  # cmap='gray'
plt.show()
```

```
%matlab
imshow(im)
```

9.4 Normalise to [0, 1]

```
#python
im_norm = skimage.im_as_float(im)
```

```
%matlab
im_norm = im2double(im)
```

9.5 RGB to gray

```
#python
gray_image = skimage.color.rgb2gray(rgb_image)
```

```
%matlab
gray_image = rgb2gray(rgb_image)
```

10 Plots

10.1 Curve Plot

```
#python
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('Title')
plt.show()
```

```
%matlab
plot(x, y)
xlabel('x - axis')
ylabel('y - axis')
title('Title')
```

10.2 Multiple Plots on Same Figure

```
#python
plt.plot(x1, y1, 'b', label = "line 1") #blue
plt.plot(x2, y2, 'r', label = "line 2") #red
plt.legend()
plt.show()
```

```
%matlab
plot(x1, y1, x2, y2)
legend('line 1', 'line 2')
```

10.3 Multiple Figures

```
#python
plt.figure(1)
plt.plot(x1,y1)
plt.figure(2)
plt.plot(x2,y2)
plt.show()
```

```
%matlab
figure(1)
plot(x1,y1)
figure(2)
plot(x2,y2)
```

10.4 Subplots

```
#python
plt.subplot(2,1,1)
plt.plot(x1, y1)
plt.subplot(2,1,2)
plt.plot(x2, y2)
plt.show()
#Arguments inside subplot function:
#(num_row, num_col, index)
#index is read from left to right row by row
```

```
%matlab
subplot(2,1,1)
plot(x1, y1)
subplot(2,1,2)
plot(x2, y2)
```

10.5 Dashed Line and Linewidth

```
#python
plt.plot(x, y, ’k--’, linewidth = 1)
plt.show()
```

```
%matlab
plot(x, y, ’--k’, ’LineWidth’, 1)
```

10.6 Plot only Points

```
#python
plt.scatter(x,y)
plt.show()
```

```
%matlab
scatter(x,y)
```

10.7 Histogram Plot

```
#python
plot.hist(x,bins = nbins)
#nbins : number of bins for grouping
plt.show()
```

```
%matlab
histogram(x,nbins)
%nbins not necessary as there is automatic binning
```

10.8 3D Plot

```python
#python
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection = '3d')
ax.plot3D(x,y,z)
plt.show()
```

```matlab
%matlab
plot3(x,y,z)
```