### Table of Rough MATLAB-Numpy Equivalents

前言：

本文是参考scipy官网一篇原文原文网址：<https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html>翻译整理而来，其中也添加了自己的新东西，也难免会有错误，大家获取本文后可以添加、修改内容，共同促进知识的传播和发展！

Numpy是Python的一个科学计算包，具有和matlab相似的函数和语法，但是也有不同，下表根据Scipy官网的一篇文章整理，列出了常用的matlab函数在numpy的实现方法，这些**并不是完全对等相同的**，但可以提供读者正确的参考方向！

在下面的表格中，假设python已经执行了以下命令：

**from** **numpy** **import** \*

**import** **scipy.linalg**

在和matlab比较之前，先讨论一下numpy模块下的array和matrix这两个类（class）。总的来说，array class更加具有通用性，而matrix class主要用于处理线性代数计算，和MATLAB语法跟相近，下面从几点方面来比较：

（1）、操作符 \* ，dot()，和 multiply()

* array：” \* “是按元素相乘的，即相当于matlab中的数乘，” dot() ”是用作矩阵乘法
* matrix：” \* “是矩阵乘法，” multiply() ”是数乘，这点和matlab更相似

（2）、向量（一维数组）处理

* array：向量形状(shape)1×N，N×1和N都是不同的。像A[:,1]这样的运算返回的是shape为N的一维数组（one-dimensional array），不是N×1类型的。一维数组（例如：A=array([1,2,3,4,5])是一维数组，而B=array([[1,2,3,4,5]])则是二维数组）的转置还是其本身，没有发生改变。
* matrix：一维数组都会转化为1×N或N×1类型的二维数组（矩阵），A[:,1]也返回shape为N×1的矩阵。

（3）高维数组处理

* array：维数（ndim）可以大于2
* matrix：维数只有2维

（4）便利的属性（attributes）

* array：有 .T 属性，可以返回转置
* matrix：有 .H, .I, .A，分别返回共轭转置，逆矩阵和asarray()

（5）数组（矩阵）构造

* array：嵌套了Python的array-like对象作为初始化，常用的就是列表(list[])和元组(tuple()),比如array([[1,2,3],[4,5,6]])和array(((1,2,3),(4,5,6)))是一样的
* matrix：采用了方便的字符串（string）作为初始化对象，例如：mat("1 2 3;5 6 8;3 5 1")，从这点也能看出matrix类和matlab更相似

一般来说更推荐使用array类。

**1、General Purpose Equivalents**

|  |  |  |
| --- | --- | --- |
| MATLAB | Numpy | Notes |
| **help** func / **doc** func | **help**(func) / func**?**(IPython) | 函数帮助信息 |
| **type** func | func**??**(IPython) | 函数类似 |
| a **&&** b | a **and** b | 短路逻辑“与”操作符,a,b都要是数字 |
| a **||** b | a **or** b | 短路逻辑“或”操作符 |
| A **&** B | A **and** B / **logical\_and**(a,b) | MATLAB的“&“是按元素操作的与运算，A和B是相同大小的数组；Python的”and”也可以实现元素间的“与”运算，A和B是list |
| A **|** B | A **or** B / **logical\_or**(a,b) | 按元素操作的“或”运算，同上 |
| **bitand**(a,b) | a **&** b | 位与，2 & 3 = 2 |
| **bitor**(a,b) | a **|** b | 位或，2 | 3 = 3 |
| **bitxor**(a,b) | a **^** b | 位异或，60 ^ 13 = 49 |
| **~~bitnot~~**~~(a)~~ | **~** a | 位取反，~60 = -61 |
| **1\*j,1\*i,1j,1i** | **1j** | 复数 |
| **eps** | **spacing(1)** | 离1最近的浮点数 |
| **ode45** | **scipy.integrate.ode(**f**)**  **.set\_integrator(‘dopri5’)** | integrate an ODE with Runge-Kutta 4,5 |
| **ode15s** | **scipy.integrate.ode(**f**)**  **.set\_integrator(‘vode’,methord=**  **‘bdf’,order=5)** | integrate an ODE with BDF method |

**2、Linear Algebra Equivalents**

|  |  |  |
| --- | --- | --- |
| MATLAB | Numpy | Notes |
| **ndims**(a) | **ndim**(a) / a**.ndim** |  |
| **numel**(a) | **size**(a) / a**.size** |  |
| **size**(a) | **shape**(a) / s**.shape** |  |
| **size**(a**, n**) | a**.shape[n-1]** |  |
| **[**1 2 3 **;** 4 5 6**]** | **array(** [ [1, 2, 3] , [4, 5, 6] ] **)** |  |
| **[** a, b **;** c, d **]** | **vstack**([**hstack**([a,b]),**hstack**([c,d])/  **bmat**(‘a b ; c d’)**.A** |  |
| a(**end**) | a[**-1**] |  |
| **a( 2, : )** | **a[ 1 ] / a[ 1, : ]** |  |
| **a(2, 5)** | **a[1, 4]** |  |
| **a(1:5, : )** | **a[0 : 5, : ] / a[ : 5] / a[0 : 5, : ]** |  |
| a(end-4:end, : ) | a[-5 : ] |  |
| **a(1 : 3, 5 : 9)** | **a[0 : 3][ : , 4 : 9]** | 注意这种情况下numpy的切片方法 |
| a([2, 4, 5], [1, 3]) | a[**ix\_**([1, 3, 4],[0, 2])] |  |
| a(3 **:** 2 **:**21, **:** ) | a[2 **:** 21 **:** 2, **:** ] |  |
| a(1 : 2 : end, : ) | a[ : : 2, :] |  |
| a**.’** | a**.transpose()** / a**.T** | 转置,注意python中a的构建要ndim>1,  a=array([1,2,3,4])和  a=array([[1,2,3,4]])是不同的 |
| a**’** | a**.conj().transpose()** / a**.conj.T** | 共轭转置 |
| a **\*** b | a**.dot**(b) / **dot**(a, b) | 同样注意numpy得到数组维数ndim>1,如果a,b的ndim都为1，返回的结果都为数量积 |
| a **.\*** b | a **\*** b | 按数组(矩阵)元素相乘 |
| a **./** b | a **/** b | 按数组(矩阵)元素相除 |
| a**.^3** | a**\*\***3 | 按元素的乘方 |
| （a>0,5） | (a>0.5) | MATLAB返回0，1构成的数组，numpy返回True,False构成的数组 |
| **find**(a>0.5) | **nonzero**(a>0.5) | 返回索引值 |
| a(:,**find**(v>0.5)) | a[ : , **nonzero**(v>0.5)[0] ] | extract the columms of a where vector v > 0.5 |
| a(a<0.5)=0 | a[a<0.5]=0 |  |
| a .\* (a>0.5) | a \* (a>0.5) |  |
| a(:) = 3 | a[:] = 3 | 将数组(矩阵)元素设置为同一值 |
| y=x | y = x**.copy()** |  |
| y=x(2,:) | y = x[1,:]**.copy()** |  |
| y=x(:) | y = x**.flatten()** |  |
| 1:10 | **arange**(1.,11.) / **r\_**[1.:11.] / r\_[1:10:10j] / **range**(1,11) | range(1,11)返回的是python的list类型 |
| [1:10]**'** | **arange**(1.,11.)**[:, newaxis]/**  **c\_[1:11:1]** | c\_可以记忆为column  r\_可以记忆为row |
| **zeros**(3,4,5) | **zeros**(**(**3,4,5**)**) / **zeros**(**[**3,4,5**]**) |  |
| **ones**(3,4) | **ones**((3,4)) |  |
| **eye**(3) | **eye**(3) |  |
| **diag**(a) | **diag**(a) |  |
| **diag**(a,0) | **diag**(a,0) |  |
| **rand**(3,4) | **rand**(3,4) | uniform distribution |
| **randn**(3,4)/  **randn**([3,4]) | **randn**(3,4) | normal distribution |
| **linspace**(-3,-3,n) | **linspace**(-3,3,n) | n equally spaced numbers |
| [x,y]=**meshgrid**(-3:0.1:3,-4:0.2:4) | x,y=**mgrid[**-3:3:0.1,-4:4:0.2**]**/  x,y=**meshgrid(**r\_[-3:3:0.1,r\_[-4:4:0.2]**)**/  x,y=**meshgrid(**linspace(-3,3,100),linspace(-4,4,120)**)** |  |
|  | **ogrid**[0:9.,0:6.] /  **ix\_**(r\_[0:9.],r\_[0:6.] | 官网说“the best way to eval functions on a grid”，自己没怎么用过 |
| [x,y]=meshgrid([1,2,4],[2,4,5]) | x,y=**meshgrid**([1,2,4],[2,4,5])/  **ix\_**([1,2,4],[2,4,5]) |  |
| **repmat**(a,m,n) | **tile**(a,(m,n)) | 把数组a看成整体以其为元素构建m\*n矩阵 |
| [a b] | **concatenate**((a,b),1) /  **hstack**((a,b)) /  **column\_stack**((a,b)) /  **c\_**[a,b] /  **append**(a,b,**axis=1**) | 使用append的时候a,b要是多维数组(ndim>1)，否则会报错，例如：  ~~a=array([1,2,3,4])~~  b=array([[1,2,3,4]])  数组b的情况才是符合的 |
| [a**;** b] | **concatenate**((a,b)) /  **vstack**((a,b)) /  **r\_**[a,b] /  **append**(a,b,**axis=0**) |  |
| **max**(a) | a**.max(0)** | 返回每列最大数值，最终的结果仍是数组 |
| **max**(a,**[],2**) | a**.max(1)** | 返回每行最大数值，最终的结果仍是数组 |
| **max**(**max**(x)) | a**.max()** |  |
| **max**(a,b) | **maximum**(a,b) |  |
| **norm**(v) | sqrt(dot(v,v)) /  **norm**(v) | L2 norm of vector v |
| **det**(a) | **det**(a) | 矩阵行列式 |
| **inv**(a) | **inv**(a) | 矩阵的逆 |
| **pinv**(a) | **pinv**(a) | 伪逆pseudo-inverse of matrix a |
| **rank**(a) | **matrix\_rank**(a) |  |
| a**\**b /  inv(a)\*b | solve(a,b)/**dot(inv(a),b)**  if a is square;  **lstsq(a,b)** otherwise  **dot(pinv(a),b)** | 求解aX=b  dot(pinv(a),b)适用满秩、欠秩和超定方程的情况 |
| b**/**a  b\*inv(a) | **dot(b,pinv(a))** | 求解Xa=b |
| [U,S,V]=**svd**(a) | U, S, Vh =  **svd**(a), V =Vh.T | 奇异值分解 |
| **chol**(a) | **cholesky**(a).T |  |
| [V,D]=**eig**(a) | D,V =  **eig**(a) | 特征值和特征向量 |
| [V,D]=**eig**(a,b) | V,D =  **eig**(a,b) |  |
| [Q,R,P]=**qr**(a,0) | Q,R =  **qr**(a) | QR分解 |
| [L,U,P]=**lu**(a) | L,U = **lu**(a) or LU,  P = **lu\_factor**(a) | LU分解 |
| **fft**(a) | **fft**(a) | 傅里叶变换 |
| **ifft**(a) | **ifft**(a) | 傅里叶反变换 |
| **sort**(a) | **sort**(a) or a**.sort()** | 排序 |
| [b,I] = **sortrows**(a,i) | I = **argsort**(a[:,i]), b=a[I,:] |  |
| **regress**(y,X) | **lstsq**(X,y) | multilinear regression |
| **decimate**(x, q) | **scipy.signal.resample**(x,**len(x)/q**) |  |
| **unique**(a) | **unique**(a) |  |
| **squeeze**(a) | a**.squeeze()** |  |
| **dec2hex**(12) | **hex**(12) | 十进制转十六进制 |
| **dec2bin**(12) | **bin**(12) | 十进制转二进制 |
| **dec2base**(12**,8**) | **oct**(12) | 十进制转八进制 |