```
from sympy import Matrix, Symbol, derive_by_array, Lambda, Function, MatrixSymbol, Derivative, diff,
from sympy import var
from sympy.abc import x, i, j, a, b


from sympy.interactive import init_printing

init_printing(pretty_print=True, wrap_line=True, num_columns=60)
```

```
def myvar(letter: str, i: int, j: int) -> Symbol:
    letter_ij = Symbol('{}_{}{}'.format(letter, i+1, j+1), is_commutative=True)
    return letter_ij


n,m,p = 3,3,2

X = Matrix(n, m, lambda i,j : myvar('x', i, j)); X
```

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

```
W = Matrix(m, p, lambda i,j : myvar('w', i, j)); W
```

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

```
A = MatrixSymbol('X',3,3); Matrix(A)
B = MatrixSymbol('W',3,2)
```

```
# Defining $N = \nu(X, W) = X \times W$
#
# * $\nu : \mathbb{R}^{(n \times m) \times (m \times p)} \rightarrow \mathbb{R}^{n \times p}$
# * $N \in \mathbb{R}^{n \times p}$
```

```
v = lambda a,b: a*b

vL = Lambda((a,b), a*b)

n = Function('v') #, Lambda((a,b), a*b))

vN = lambda mat1, mat2: Matrix(mat1.shape[0], mat2.shape[1], lambda i, j: Symbol("n_{}{}".format(i+1

Nelem = vN(X, W); Nelem
```

$$\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \\ n_{31} & n_{32} \end{bmatrix}$$

```
n(X,W)
```

$$v\left(\begin{bmatrix}x_{11} & x_{12} & x_{13}\\x_{21} & x_{22} & x_{23}\\x_{31} & x_{32} & x_{33}\end{bmatrix}, \begin{bmatrix}w_{11} & w_{12}\\w_{21} & w_{22}\\w_{31} & w_{32}\end{bmatrix}\right)$$

```
n(A,B)
```

$$v(X, W)$$

```
n(X,W).replace(n, v) # replace works when v = python lambda
```

$$\begin{bmatrix}w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\\w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\\w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\end{bmatrix}$$

```
n(X,W).subs({n: vL}) # subs works when v = sympy lambda
```

$$\begin{bmatrix}w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\\w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\\w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\end{bmatrix}$$

```
n(X,W).replace(n, vL)
```

$$\begin{bmatrix}w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\\w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\\w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\end{bmatrix}$$

```
n(X,W).subs({n: v})# subs() doesn't work when v is python lambda
```

$$v\left(\begin{bmatrix}x_{11} & x_{12} & x_{13}\\x_{21} & x_{22} & x_{23}\\x_{31} & x_{32} & x_{33}\end{bmatrix}, \begin{bmatrix}w_{11} & w_{12}\\w_{21} & w_{22}\\w_{31} & w_{32}\end{bmatrix}\right)$$

```
Matrix(n(A,B).subs({n: vL}))
```

$$\begin{bmatrix}W_{0,0}X_{0,0} + W_{1,0}X_{0,1} + W_{2,0}X_{0,2} & W_{0,1}X_{0,0} + W_{1,1}X_{0,1} + W_{2,1}X_{0,2}\\W_{0,0}X_{1,0} + W_{1,0}X_{1,1} + W_{2,0}X_{1,2} & W_{0,1}X_{1,0} + W_{1,1}X_{1,1} + W_{2,1}X_{1,2}\\W_{0,0}X_{2,0} + W_{1,0}X_{2,1} + W_{2,0}X_{2,2} & W_{0,1}X_{2,0} + W_{1,1}X_{2,1} + W_{2,1}X_{2,2}\end{bmatrix}$$

```
#N = v(X, W); N
N = n(A,B); N
```

$$v(X, W)$$

```
N.replace(n, v)
```

XW

```
N.replace(n, v).subs({A: X, B:W}) # replacing ariable values after doing function doesn't make the f
```

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

```
N.subs({n: vL, A:X, B:W})
```

$$\begin{bmatrix} w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13} \\ w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23} \\ w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33} \end{bmatrix}$$

```
Nspec = N.subs({A:X, B:W}).replace(n, v); Nspec
```

$$\begin{bmatrix} w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13} \\ w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23} \\ w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33} \end{bmatrix}$$

```
N.diff(N)
```

1

```
N.diff(X)
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
# way 2 of declaring S (better way)
sigma = Function('sigma')

sigmaApply = Function("sigma_apply") #lambda matrix:  matrix.applyfunc(sigma)

sigmaApply_ = lambda matrix: matrix.applyfunc(sigma)

sigmaApply(A)
```

$\sigma_{apply}(X)$

```
sigmaApply(A).subs({A: X})
```

$$\sigma_{apply}\left(\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}\right)$$

```
sigmaApply_(A)
```

$$\left(d \mapsto \sigma(d)\right)_{\circ}\left(X\right)$$

```
sigmaApply(A).subs({A: X}).replace(sigmaApply, sigmaApply_) # NOTE: subs of functions doesn't work,
```

$$\begin{bmatrix}\sigma(x_{11}) & \sigma(x_{12}) & \sigma(x_{13})\\ \sigma(x_{21}) & \sigma(x_{22}) & \sigma(x_{23})\\ \sigma(x_{31}) & \sigma(x_{32}) & \sigma(x_{33})\end{bmatrix}$$

```
S = sigmaApply(N); S
```

$$\sigma_{apply}(v(X, W))$$

```
Derivative(S, S)
```

$$\frac{\partial}{\partial \sigma_{apply}(v(X, W))} \sigma_{apply}(v(X, W))$$

```
Derivative(S, S).doit()
```

$$1$$

```
Derivative(S, n(A,B)).doit()
```

$$\frac{\partial}{\partial v(X, W)} \sigma_{apply}(v(X, W))$$

```
#lambd = Function("lambda")
#Lagain = lambd(sigmaApply(n(A))); Lagain



# diff(Lagain, A) # never execute
#
```

```
S.replace(A,X).replace(B,W)
```

$$\sigma_{apply}\left(v\left(\begin{bmatrix}x_{11} & x_{12} & x_{13}\\ x_{21} & x_{22} & x_{23}\\ x_{31} & x_{32} & x_{33}\end{bmatrix}, \begin{bmatrix}w_{11} & w_{12}\\ w_{21} & w_{22}\\ w_{31} & w_{32}\end{bmatrix}\right)\right)$$

```
S.replace(n, v)
```

$$\sigma_{apply}(XW)$$