SympyMatrixTryouts

October 2, 2020

[1]:
```python
from sympy import sin, cos, Matrix
from sympy.abc import rho, phi
X = Matrix([rho*cos(phi), rho*sin(phi), rho**2])
Y = Matrix([rho, phi])

X
```

[1]:
$$\begin{bmatrix}\rho\cos(\phi)\\ \rho\sin(\phi)\\ \rho^2\end{bmatrix}$$

[2]:
```python
Y
```

[2]:
$$\begin{bmatrix}\rho\\ \phi\end{bmatrix}$$

[3]:
```python
X.jacobian(Y)
```

[3]:
$$\begin{bmatrix}\cos(\phi) & -\rho\sin(\phi)\\ \sin(\phi) & \rho\cos(\phi)\\ 2\rho & 0\end{bmatrix}$$

[4]:
```python
from sympy import MatrixSymbol, Matrix
from sympy.core.function import Function
from sympy import FunctionMatrix, Lambda

n, m, p = 3, 3, 2

X = MatrixSymbol('X', n, m)
W = MatrixSymbol('W', m, p)

(X.T*X).I*W
```

[4]: $X^{-1}\left(X^{T}\right)^{-1}W$

[5]:
```
Matrix(X)
```

[5]:
$$\begin{bmatrix} X_{0,0} & X_{0,1} & X_{0,2} \\ X_{1,0} & X_{1,1} & X_{1,2} \\ X_{2,0} & X_{2,1} & X_{2,2} \end{bmatrix}$$

[6]:
```
Matrix(X * W)
```

[6]:
$$\begin{bmatrix} W_{0,0}X_{0,0} + W_{1,0}X_{0,1} + W_{2,0}X_{0,2} & W_{0,1}X_{0,0} + W_{1,1}X_{0,1} + W_{2,1}X_{0,2} \\ W_{0,0}X_{1,0} + W_{1,0}X_{1,1} + W_{2,0}X_{1,2} & W_{0,1}X_{1,0} + W_{1,1}X_{1,1} + W_{2,1}X_{1,2} \\ W_{0,0}X_{2,0} + W_{1,0}X_{2,1} + W_{2,0}X_{2,2} & W_{0,1}X_{2,0} + W_{1,1}X_{2,1} + W_{2,1}X_{2,2} \end{bmatrix}$$

[7]:
```
from sympy import I, Matrix, symbols
from sympy.physics.quantum import TensorProduct

m1 = Matrix([[1,2],[3,4]])
m2 = Matrix([[1,0],[0,1]])

m1
```

[7]:
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

[8]:
```
TensorProduct(m1, m2)
```

[8]:
$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}$$

[9]:
```
f = Function('f')
F = FunctionMatrix(3,4, f)
Matrix(F)
```

[9]:
$$\begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & f(0,3) \\ f(1,0) & f(1,1) & f(1,2) & f(1,3) \\ f(2,0) & f(2,1) & f(2,2) & f(2,3) \end{bmatrix}$$

[10]:
```
y = Function('y')(X)
y
```

[10]:

$y(X)$

[11]:

```
# y.diff(X)
from sympy.abc import x,y,z,t,e,r,a,b,c
from sympy import derive_by_array
from sympy import sin, exp, cos

#basis = Matrix([x, y, z])
Matrix(X)

m = Matrix([[exp(x), sin(y*z), t*cos(x*y)], [x, x*y, t], [x,x,z] ])
basis = Matrix([[x,y,z], [x,y,z], [x,y,z]])
type(basis)
X.as_explicit()
M = Matrix([[x,y,z], [t,e,r], [a,b,c]])
ax = derive_by_array(m, M)
ax
```

[11]:

$$\left[\begin{matrix}\left[\begin{matrix}e^{x} & 0 & - t y \sin{\left(x y \right)}\\1 & y & 0\\1 & 1 & 0\end{matrix}\right] & \left[\begin{matrix}0 & z \cos{\left(y z \right)} & - t x \sin{\left(x y \right)}\\0 & x & 0\\0 & 0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & y \cos{\left(y z \right)} & 0\\0 & 0 & 0\\0 & 0 & 1\end{matrix}\right]\\\left[\begin{matrix}0 & 0 & \cos{\left(x y \right)}\\0 & 0 & 1\\0 & 0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}\right]\\\left[\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}\right]\end{matrix}\right]$$

[12]:

```
Matrix(X)[1,2]
x_12 = Matrix(X)[1,2]
type(Matrix(X)[1,2])
f = Function('f')
g = Function('g')
h = f(g(x_12))
h
h.diff(x_12)
```

[12]:

$$\frac{d}{dg(X_{1,2})} f(g(X_{1,2})) \frac{d}{dX_{1,2}} g(X_{1,2})$$

[13]:

```
```

[13]:
```python
from sympy import Symbol
from sympy.abc import i, j

f = Function('f')

#def makeF(i, j):
#     x_ij = Symbol('x_{}{}'.format(i,j), is_commutative=True)
#     return f(x_ij)


def makeX(i, j):
     x_ij = Symbol('x_{}{}'.format(i, j), is_commutative=True)
     return x_ij #Lambda((i,j), x_ij)


# NOTE: even if i, j are sympy Symbols, passing them here with python's␣
↪lambda instead of sympy's Lambda lets the indices actually be seen!!!
X = Matrix(2, 3, lambda i,j: makeX(i+1, j+1)); X
# BAD
#X = FunctionMatrix(2,3, Lambda((a,b), makeX(int(a), int(b))))
```

[13]:
$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix}$$

[14]:
```python
X[0,0]
```

[14]:
$$x_{11}$$

[15]:
```python
from sympy import derive_by_array

derive_by_array(Matrix(X), Matrix(X))
```

[15]:
$$\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}$$

[16]:
```python
from sympy import Symbol
x_11 = Symbol('x_11', is_commutative=True)
x_12 = Symbol('x_12', is_commutative=True)
x_13 = Symbol('x_13', is_commutative=True)
```

```
x_21 = Symbol('x_21', is_commutative=True)
x_22 = Symbol('x_22', is_commutative=True)
x_23 = Symbol('x_23', is_commutative=True)
x_31 = Symbol('x_31', is_commutative=True)
x_32 = Symbol('x_32', is_commutative=True)
x_33 = Symbol('x_33', is_commutative=True)
X = Matrix([[x_11,x_12, x_13], [x_21,x_22,x_23], [x_31,x_32,x_33]]); X

w_11 = Symbol('w_11', is_commutative=True)
w_12 = Symbol('w_12', is_commutative=True)
w_21 = Symbol('w_21', is_commutative=True)
w_22 = Symbol('w_22', is_commutative=True)
w_31 = Symbol('w_31', is_commutative=True)
w_32 = Symbol('w_32', is_commutative=True)
W = Matrix([[w_11, w_12], [w_21, w_22], [w_31,w_32]]); W


y_11 = Function('y_11')
y_12 = Function('y_12')
y_21 = Function('y_21')
y_22 = Function('y_22')
Y = Matrix([[y_11(X),y_12(X)],[y_21(X),y_22(X)]])
Y

#Y.diff(X)

#y = Function('y')
```

[16]:

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

[17]:

```
#derive_by_array(Y, X)
```

[18]:

```
A = Matrix(MatrixSymbol('x', 3,3)); A
B = Matrix(MatrixSymbol('w', 3,2)); B
```

[18]:

$$\begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \\ w_{2,0} & w_{2,1} \end{bmatrix}$$

[19]:
```
A*B

derive_by_array(A*B, A)
```

[19]:

$$\begin{bmatrix} \begin{bmatrix} w_{0,0} & w_{0,1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} w_{1,0} & w_{1,1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} w_{2,0} & w_{2,1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ w_{0,0} & w_{0,1} \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ w_{1,0} & w_{1,1} \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ w_{2,0} & w_{2,1} \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{0,0} & w_{0,1} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{1,0} & w_{1,1} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{2,0} & w_{2,1} \end{bmatrix} \end{bmatrix}$$

[20]:
```
(A*B).diff(A)

assert (A*B).diff(A) == derive_by_array(A*B, A)
```

[21]:
```
X*W
```

[21]:

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

[22]:
```
derive_by_array(X*W, X)
```

[22]:

$$\begin{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} w_{21} & w_{22} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} w_{31} & w_{32} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ w_{11} & w_{12} \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ w_{21} & w_{22} \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ w_{31} & w_{32} \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{11} & w_{12} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{21} & w_{22} \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{31} & w_{32} \end{bmatrix} \end{bmatrix}$$

[23]:
```
(X*W).diff(X)
```

[23]:

$$\left[\begin{matrix}\left[\begin{matrix}w_{11} & w_{12}\\0 & 0\\0 & 0\end{matrix}\right] & \left[\begin{matrix}w_{21} & w_{22}\\0 & 0\\0 & 0\end{matrix}\right] & \left[\begin{matrix}w_{31} & w_{32}\\0 & 0\\0 & 0\end{matrix}\right]\\\left[\begin{matrix}0 & 0\\w_{11} & w_{12}\\0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & 0\\w_{21} & w_{22}\\0 & 0\end{matrix}\right] & \left[\begin{matrix}0 & 0\\w_{31} & w_{32}\\0 & 0\end{matrix}\right]\\\left[\begin{matrix}0 & 0\\0 & 0\\w_{11} & w_{12}\end{matrix}\right] & \left[\begin{matrix}0 & 0\\0 & 0\\w_{21} & w_{22}\end{matrix}\right] & \left[\begin{matrix}0 & 0\\0 & 0\\w_{31} & w_{32}\end{matrix}\right]\end{matrix}\right]$$

[24]:

```
### JACOBIAN:
def makeX(i, j):
     x_ij = Symbol('x_{}{}'.format(i,j), is_commutative=True)
     return x_ij #Lambda((i,j), x_ij)

#u = lambda i,j : makeX(i,j)
#FunctionMatrix(9,1, Lambda((i,j), makeX(i,j)))

def makeYofX(i, j):
     yx_ij = Symbol('y_{}{}'.format(i,j), is_commutative=True)
     return yx_ij #Lambda((i,j), x_ij)
```

[25]:

```
def var(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
# NOTE: even if i, j are sympy Symbols, passing them here with python's␣
 ↪lambda instead of sympy's Lambda lets the indices actually be seen!!!
X = Matrix(n,m, lambda i,j: var('x', i,j)); X
```

[25]:

$$\left[\begin{matrix}x_{11} & x_{12} & x_{13}\\x_{21} & x_{22} & x_{23}\\x_{31} & x_{32} & x_{33}\end{matrix}\right]$$

[26]:

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

[26]:

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

[27]:
```
#res = y.subs({y[0]:x[0]**2* x[2] +x[1]}); res
```

[28]:
```
#res.subs({x[0]: 23, x[1]: 14})
```

[29]:
```
#derive_by_array(res[0], x[0])
#derive_by_array(res[1], x[0])
```

[30]:
```
#derive_by_array(y, x)
```

[31]:
```
#y.diff(x)
#y.jacobian(x)
```

[32]:
```
from sympy import Function, hessian, pprint
from sympy.abc import x, y
f = Function('f')(x, y)
g1 = Function('g')(x, y)
g2 = x**2 + 3*y
hessian(f, (x,y),[g1,g2])
```

[32]:

$$\begin{bmatrix} 0 & 0 & \frac{\partial}{\partial x} g(x,y) & \frac{\partial}{\partial y} g(x,y) \\ 0 & 0 & 2x & 3 \\ \frac{\partial}{\partial x} g(x,y) & 2x & \frac{\partial^2}{\partial x^2} f(x,y) & \frac{\partial^2}{\partial y \partial x} f(x,y) \\ \frac{\partial}{\partial y} g(x,y) & 3 & \frac{\partial^2}{\partial y \partial x} f(x,y) & \frac{\partial^2}{\partial y^2} f(x,y) \end{bmatrix}$$

[33]:
```
#Matrix(FunctionMatrix(n, 1, Lambda((i,j), Function("y_{}{}".
 ↪format(i,j)))))
rho = Matrix([[Symbol("r_{}{}".format(i+1,j+1)) for j in range(5)] for i␣
 ↪in range(5)])
rho
```

[33]:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} \\ r_{21} & r_{22} & r_{23} & r_{24} & r_{25} \\ r_{31} & r_{32} & r_{33} & r_{34} & r_{35} \\ r_{41} & r_{42} & r_{43} & r_{44} & r_{45} \\ r_{51} & r_{52} & r_{53} & r_{54} & r_{55} \end{bmatrix}$$

[34]:
```python
derive_by_array(rho, rho[2,1])
```

[34]:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[35]:
```python
derive_by_array(rho[2,1], rho)
```

[35]:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[36]:
```python
rho.diff(rho[2,1])
```

[36]:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[37]:
```python
from sympy.abc import x
from sympy.utilities.lambdify import lambdify, implemented_function
from sympy import Function

f = implemented_function('f', lambda x: x)
lam_f = lambdify(x, f(x))

lam_f(4)
```

[37]: 4