ch1_phase3

September 24, 2020

[1]:
```
from sympy import Matrix, Symbol, derive_by_array, Lambda, Function,␣
 ↪MatrixSymbol, Derivative
from sympy import var
from sympy.abc import x, i, j, a, b, c, d
```

[2]:
```
def myvar(letter: str, i: int, j: int) -> Symbol:
    letter_ij = Symbol('{}_{}{}'.format(letter, i+1, j+1), is_commutative=True)
    return letter_ij
ns,ms,ps = 3,3,2
X = Matrix(ns, ms, lambda i,j : myvar('x', i, j)); X
```

[2]:
$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

[3]:
```
W = Matrix(ms, ps, lambda i,j : myvar('w', i, j)); W
```

[3]:
$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

[4]:
```
#TODO how to make matrix symbols commutative?
# A = MatrixSymbol('X',ns,ms, is_commutative=True); Matrix(A)
A = MatrixSymbol('X',ns,ms); Matrix(A)
B = MatrixSymbol('W',ms,ps)
```

[5]:
```
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, j+1))); vN
Nelem = vN(X, W)
Nelem
```

[5]:
$$\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \\ n_{31} & n_{32} \end{bmatrix}$$

[6]:
```
Nspec = v(X,W)
Nspec
```

[6]:
$$\begin{bmatrix} w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13} \\ w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23} \\ w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33} \end{bmatrix}$$

[7]:
```
#N = v(X, W); N
N = n(A,B)
N
```

[7]:
$$v(X, W)$$

[8]:
```
def siga(mat: Matrix) -> Matrix:
     #lst = mat.tolist()
     nr, nc = mat.shape
     applied = [[sigma(mat[i,j]) for j in range(0, nc)] for i in range(0, nr)]
     return Matrix(applied)
# 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_2 = lambda matrix: siga(matrix)
S = sigmaApply(N); S
```

[8]:
$$\sigma_{apply}(v(X, W))$$

[9]:
```
sigmaApply_(Nelem)
```

[9]:
$$\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[10]:
```
sigmaApply_2(Nelem)
```

[10]:
$$\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[11]:
```
#sigmaApply_2(A*B).diff(Matrix(A))
```

[12]:
```
Sspec = S.subs({A:X, B:W}).replace(n, v).replace(sigmaApply, sigmaApply_)
Sspec
```

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

[13]:
```
Selem = S.replace(n, vN).replace(sigmaApply, sigmaApply_)
Selem
```

[13]:
$$\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[14]:
```
import itertools
elemToSpecD = dict(itertools.chain(*[[(Nelem[i, j], Nspec[i, j]) for j in␣
 ↪range(2)] for i in range(3)]))
elemToSpec = list(elemToSpecD.items())
Matrix(elemToSpec)
```

[14]:
$$\begin{bmatrix} n_{11} & w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} \\ n_{12} & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13} \\ n_{21} & w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} \\ n_{22} & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23} \\ n_{31} & w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} \\ n_{32} & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33} \end{bmatrix}$$

[15]:
```
elemToSpecFuncD = dict(itertools.chain(*[[(Nelem[i, j], Function("n_{}{}".
 ↪format(i + 1, j + 1))(Nspec[i, j])) for j in range(2)] for i in range(3)]))
elemToSpecFunc = list(elemToSpecFuncD.items())
Matrix(elemToSpecFunc)
```

[15]:
$$\begin{bmatrix} n_{11} & \mathrm{n}_{11}\left(w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13}\right) \\ n_{12} & \mathrm{n}_{12}\left(w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\right) \\ n_{21} & \mathrm{n}_{21}\left(w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23}\right) \\ n_{22} & \mathrm{n}_{22}\left(w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\right) \\ n_{31} & \mathrm{n}_{31}\left(w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33}\right) \\ n_{32} & \mathrm{n}_{32}\left(w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\right) \end{bmatrix}$$

[16]:
```
elemToSpecFuncArgsD = dict(itertools.chain(*[[(Nelem[i, j], Function("n_{}{}".
 ↪format(i + 1, j + 1))(*X,*W)) for j in range(2)] for i in range(3)]))
elemToSpecFuncArgs = list(elemToSpecFuncArgsD.items())
Matrix(elemToSpecFuncArgs)
```

[16]:
$$\begin{bmatrix} n_{11} & \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{12} & \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{21} & \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{22} & \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{31} & \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{32} & \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \end{bmatrix}$$

[17]:
```
elemToMatArgD = dict(itertools.chain(*[[(Nelem[i, j], Function("n_{}{}".
 ↪format(i+1,j+1))(A,B) ) for j in range(2)] for i in range(3)]))
elemToMatArg = list(elemToMatArgD.items())
Matrix(elemToMatArg)
```

[17]:
$$\begin{bmatrix} n_{11} & \mathrm{n}_{11}\left(X,W\right) \\ n_{12} & \mathrm{n}_{12}\left(X,W\right) \\ n_{21} & \mathrm{n}_{21}\left(X,W\right) \\ n_{22} & \mathrm{n}_{22}\left(X,W\right) \\ n_{31} & \mathrm{n}_{31}\left(X,W\right) \\ n_{32} & \mathrm{n}_{32}\left(X,W\right) \end{bmatrix}$$

[18]:
```
matargToSpecD = dict(zip(elemToMatArgD.values(), elemToSpecD.values()))
matargToSpec = list(matargToSpecD.items())
Matrix(matargToSpec)
```

[18]:
$$\begin{bmatrix} \mathrm{n}_{11}\left(X,W\right) & w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13} \\ \mathrm{n}_{12}\left(X,W\right) & w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13} \\ \mathrm{n}_{21}\left(X,W\right) & w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23} \\ \mathrm{n}_{22}\left(X,W\right) & w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23} \\ \mathrm{n}_{31}\left(X,W\right) & w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33} \\ \mathrm{n}_{32}\left(X,W\right) & w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33} \end{bmatrix}$$

[19]:
```
Selem
```

[19]:
$$\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[20]:
```
Selem.subs(elemToSpecD)
```

[20]:
$$\begin{bmatrix} \sigma(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}) & \sigma(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}) \\ \sigma(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}) & \sigma(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}) \\ \sigma(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}) & \sigma(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}) \end{bmatrix}$$

[21]:
```
Selem[0,1].diff(Nelem[0,1])
```

[21]:
$$\frac{d}{dn_{12}}\sigma(n_{12})$$

[22]:
```
Selem[0,1].diff(Nelem[0,1]).subs({Nelem[0,1] : Nspec[0,1]})
#Selem[0,1].diff(Nelem[0,1]).subs(dict([{Nelem[0,1] : Nspec[0,1]}]))
```

[22]:
$$\left.\frac{d}{dn_{12}}\sigma(n_{12})\right|_{n_{12}=w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}}$$

[23]:
```
Selem[0,1].diff(Nelem[0,1]).subs({Nelem[0,1] : Nspec[0,1]}).subs({Nspec[0,1] :␣
↪23})
```

[23]:
$$\left.\frac{d}{dn_{12}}\sigma(n_{12})\right|_{n_{12}=23}$$

[24]:
```
Selem[0,1].diff(Nelem[0,1]).subs({Nelem[0,1] : Nspec[0,1]}).replace(sigma,␣
↪lambda x: 8*x**3)
```

[24]:
$$\frac{d}{dn_{12}} 8n_{12}^3 \Big|_{n_{12}=w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}}$$

[25]:
```
Selem[0,1].diff(Nelem[0,1]).replace(sigma, lambda x: 8*x**3)
```

[25]:
$$\frac{d}{dn_{12}} 8n_{12}^3$$

[26]:
```
Selem[0,1].diff(Nelem[0,1]).replace(sigma, lambda x: 8*x**3).doit()
```

[26]:
$$24n_{12}^2$$

[27]:
```
# ### GOT IT: can replace now with expression and do derivative with respect to␣
 ↪that expression.
Selem[0,1].diff(Nelem[0,1]).subs({Nelem[0,1] : Nspec[0,1]}).replace(sigma,␣
 ↪lambda x: 8*x**3).doit()
```

[27]:
$$24\left(w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\right)^2$$

[28]:
```
Selem[0,1].subs({Nelem[0,1] : Nspec[0,1]}).diff(X[0,1])#.subs({Nelem[0,1] :␣
 ↪Nspec[0,1]})
```

[28]:
$$w_{22} \frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}}$$

[29]:
```
Selem
```

[29]:
$$\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[30]:
```
nt = Nelem.subs(elemToSpecFunc); nt
```

[30]:
$$\begin{bmatrix} \mathrm{n}_{11}\left(w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13}\right) & \mathrm{n}_{12}\left(w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\right) \\ \mathrm{n}_{21}\left(w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23}\right) & \mathrm{n}_{22}\left(w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\right) \\ \mathrm{n}_{31}\left(w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33}\right) & \mathrm{n}_{32}\left(w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\right) \end{bmatrix}$$

[31]:
```
st = Selem.subs(elemToSpecFunc); st
```

[31]:
$$\begin{bmatrix} \sigma(\mathrm{n}_{11}\left(w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13}\right)) & \sigma(\mathrm{n}_{12}\left(w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\right)) \\ \sigma(\mathrm{n}_{21}\left(w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23}\right)) & \sigma(\mathrm{n}_{22}\left(w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\right)) \\ \sigma(\mathrm{n}_{31}\left(w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33}\right)) & \sigma(\mathrm{n}_{32}\left(w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\right)) \end{bmatrix}$$

[32]:
```
st.diff(nt)
```

[32]:

$$\begin{bmatrix}\begin{bmatrix}\frac{\partial}{\partial \mathrm{n}_{11}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right)}\sigma(\mathrm{n}_{11}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right)) & 0\\ 0 & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & \frac{\partial}{\partial \mathrm{n}_{12}\left(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}\right)}\sigma(\mathrm{n}_{12}\left(w_{12}x_{11}+w_{2\\ 0 & 0\\ 0 & 0\end{bmatrix}\\ \begin{bmatrix}0 & 0\\ \frac{\partial}{\partial \mathrm{n}_{21}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right)}\sigma(\mathrm{n}_{21}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right)) & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & 0\\ 0 & \frac{\partial}{\partial \mathrm{n}_{22}\left(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}\right)}\sigma(\mathrm{n}_{22}\left(w_{12}x_{21}+w_{2\\ 0 & 0\end{bmatrix}\\ \begin{bmatrix}0 & 0\\ 0 & 0\\ \frac{\partial}{\partial \mathrm{n}_{31}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right)}\sigma(\mathrm{n}_{31}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right)) & 0\end{bmatrix} & \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & \frac{\partial}{\partial \mathrm{n}_{32}\left(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}\right)}\sigma(\mathrm{n}_{32}\left(w_{12}x_{31}+w_{2\end{bmatrix}\end{bmatrix}$$

[33]:
```
st[0,0].diff(st[0,0].args[0])
```

[33]:
$$\frac{\partial}{\partial \mathrm{n}_{11}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right)}\sigma(\mathrm{n}_{11}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right))$$

[34]:
```
temp = st[0,0].diff(X[0,0]); temp
#nt[0,0]
#temp.replace(Function("n_11")(nt[0,0].args[0]), nt[0,0].args[0])
#temp.subs({nt[0,0] : nt[0,0].args[0]})
```

[34]:
$$w_{11}\frac{\partial}{\partial \mathrm{n}_{11}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right)}\sigma(\mathrm{n}_{11}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right))\ \frac{d}{d\xi_1}\mathrm{n}_{11}\left(\xi_1\right)\Big|_{\xi_1=w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}}$$

[35]:
```
st[0,0].diff(st[1,0].args[0])
```

[35]:
$$0$$

[36]:
```
Selem.diff(Nelem)
```

[36]:
$$\begin{bmatrix}\begin{bmatrix}\frac{d}{dn_{11}}\sigma(n_{11}) & 0\\ 0 & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & \frac{d}{dn_{12}}\sigma(n_{12})\\ 0 & 0\\ 0 & 0\end{bmatrix}\\ \begin{bmatrix}0 & 0\\ \frac{d}{dn_{21}}\sigma(n_{21}) & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & 0\\ 0 & \frac{d}{dn_{22}}\sigma(n_{22})\\ 0 & 0\end{bmatrix}\\ \begin{bmatrix}0 & 0\\ 0 & 0\\ \frac{d}{dn_{31}}\sigma(n_{31}) & 0\end{bmatrix} & \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & \frac{d}{dn_{32}}\sigma(n_{32})\end{bmatrix}\end{bmatrix}$$

[37]:
```
Selem.diff(Nelem).subs(elemToSpecFunc)
```

[37]:

$$\begin{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \mathrm{n}_{11}\,(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13})}\sigma(\mathrm{n}_{11}\,(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13})) & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \frac{\partial}{\partial \mathrm{n}_{12}\,(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13})}\sigma(\mathrm{n}_{12}\,(w_{12}x_{11}+w_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ \frac{\partial}{\partial \mathrm{n}_{21}\,(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23})}\sigma(\mathrm{n}_{21}\,(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23})) & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & \frac{\partial}{\partial \mathrm{n}_{22}\,(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23})}\sigma(\mathrm{n}_{22}\,(w_{12}x_{21}+w_2 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\partial}{\partial \mathrm{n}_{31}\,(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33})}\sigma(\mathrm{n}_{31}\,(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33})) & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{\partial}{\partial \mathrm{n}_{32}\,(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33})}\sigma(\mathrm{n}_{32}\,(w_{12}x_{31}+w_2 \end{bmatrix}$$

[38]:
```
# CAN even replace elements after have done an operation on them!!! replacing␣
 ↪n_21 * 2 with the number 4.
Sspec.subs({Nspec[0, 0]: 3}).replace(sigma, lambda x: 2 * x).replace(Nspec[2, 1]␣
 ↪* 2, 4)
```

[38]:
$$\begin{bmatrix} 6 & 2w_{12}x_{11}+2w_{22}x_{12}+2w_{32}x_{13} \\ 2w_{11}x_{21}+2w_{21}x_{22}+2w_{31}x_{23} & 2w_{12}x_{21}+2w_{22}x_{22}+2w_{32}x_{23} \\ 2w_{11}x_{31}+2w_{21}x_{32}+2w_{31}x_{33} & 4 \end{bmatrix}$$

[39]:
```
lambd = Function("lambda")
lambd_ = lambda matrix : sum(matrix)
L = lambd(S); L
```

[39]:
$$\lambda\left(\sigma_{apply}\left(v(X,W)\right)\right)$$

[40]:
```
L.replace(n, vN).replace(sigmaApply, sigmaApply_)
```

[40]:
$$\lambda\left(\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}\right)$$

[41]:
```
#L.replace(n, vN).replace(sigmaApply, sigmaApply_).diff(Nelem[0,0])
```

[42]:
```
Lsum = L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_)
Lsum
```

[42]:
$$\sigma(n_{11})+\sigma(n_{12})+\sigma(n_{21})+\sigma(n_{22})+\sigma(n_{31})+\sigma(n_{32})$$

[43]:
```
Lsum.diff(Nelem)
```

[43]:
$$\begin{bmatrix} \frac{d}{dn_{11}}\sigma(n_{11}) & \frac{d}{dn_{12}}\sigma(n_{12}) \\ \frac{d}{dn_{21}}\sigma(n_{21}) & \frac{d}{dn_{22}}\sigma(n_{22}) \\ \frac{d}{dn_{31}}\sigma(n_{31}) & \frac{d}{dn_{32}}\sigma(n_{32}) \end{bmatrix}$$

[44]:
```
Lsum.subs(elemToSpecD)#.diff(X[2,1])
```

[44]:
$$\sigma(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}) + \sigma(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}) + \sigma(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}) + \sigma(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13})+\sigma(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23})+\sigma(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33})$$

[45]:
```
Lsum.subs(elemToSpecD).diff(X)
```

[45]:
$$\begin{bmatrix} w_{11}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}} + w_{12}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}} & w_{21}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}} + w_{2} \\ w_{11}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}} + w_{12}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}} & w_{21}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}} + w_{2} \\ w_{11}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}} + w_{12}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}} & w_{21}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}} + w_{2} \end{bmatrix}$$

[46]:
```
# METHOD 1: direct matrix diff
#
### END RESULT ACHIEVED HERE (this is the end result and the most specific form␣
 ↪of the result of the  matrix differentiation, when sigma is unknown)
specToElemD = {v:k for k,v in elemToSpecD.items()}
assert Lsum == L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd,␣
 ↪lambd_)
Lsum.subs(elemToSpecD).diff(X).subs(specToElemD)
```

[46]:
$$\begin{bmatrix} w_{11}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{11}} + w_{12}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{12}} & w_{21}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{11}} + w_{22}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{12}} & w_{31}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{11}} + w_{32}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right| \\ w_{11}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{21}} + w_{12}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{22}} & w_{21}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{21}} + w_{22}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{22}} & w_{31}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{21}} + w_{32}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right| \\ w_{11}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{31}} + w_{12}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{32}} & w_{21}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{31}} + w_{22}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{32}} & w_{31}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right|_{\xi_1=n_{31}} + w_{32}\left.\frac{d}{d\xi_1}\sigma(\xi_1)\right| \end{bmatrix}$$

[47]:
```
# METHOD 2: doing matrix symbol diff
#
#### NOW DOING THE MATRIX SYMBOL DIFF EXPRESSION (trying to achieve a form that␣
 ↪shows the chain rule w.r.t to matrix symbol)
Selem
```

[47]:
$$\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[48]:
```
L
```

[48]:
$$\lambda\left(\sigma_{apply}(v(X,W))\right)$$

[49]:
```
#L.replace(A, A.T).diff(A) #ERROR: fatal python error ... why??
```

[50]:
```
#L.replace(n,v).diff(A).replace(sigmaApply, sigmaApply_) # ERROR
#L.replace(n,vN).subs(elemToSpecFuncD).replace(sigmaApply, sigmaApply_).diff(X)␣
 ↪# why the zero matrix?
```

[51]:
```
L.replace(n,v).diff(A)
```

[51]:
$$\left.\frac{d}{d\xi_1}\sigma_{apply}(\xi_1)\right|_{\xi_1=XW} \frac{\partial}{\partial\sigma_{apply}(XW)}\lambda\left(\sigma_{apply}(XW)\right)\frac{\partial}{\partial X}XW$$

[52]:
```
L.replace(n,vL).diff(A)
```

[52]:
$$\frac{d}{d\xi_1}\sigma_{apply}(\xi_1)\Big|_{\xi_1=XW}\frac{\partial}{\partial\sigma_{apply}(XW)}\lambda\left(\sigma_{apply}(XW)\right)\frac{\partial}{\partial X}XW$$

[53]:

[53]:

[53]:

[53]:

[53]:
```
#L.replace(n,v).diff(A).replace(lambd,lambd_) ### ERROR sigma object is not␣
 ↪iterable
#L.replace(n,vL).diff(A).replace(sigmaApply, sigmaApply_)### ERROR
#L.replace(n,v).diff(A).replace(sigmaApply, sigmaApply_) ### ERROR dummy object␣
 ↪has no attribute applyfunc
```

[54]:
```
#L.replace(sigmaApply, sigmaApply_).diff(A) # ERROR
# L.replace(lambd, lambd_) # ERROR
#L.replace(n, v).replace(sigmaApply, sigmaApply_2)# shows matrix results, too␣
 ↪specific, want the function composition notation as below but just applied to␣
 ↪the function v(X,W) in abstract way
### METHOD 0: (prepare by substituting n --> v, then sigmaApply --> sigma)
L.replace(n, v).replace(sigmaApply, sigmaApply_)#.replace(lambd, lambd_)
```

[54]:
$$\lambda\left(\left(d\mapsto\sigma(d)\right)_{\circ}(XW)\right)$$

[55]:
```
# NOTE: the point here is that even replacing with a sympy Lambda doesn't give␣
 ↪same result as above since above uses the V.applyfunc(sigma) within the Lambda.
L.replace(sigmaApply, Lambda(d, sigma(d)))
```

[55]:
$$\lambda(\sigma(v(X,W)))$$

[56]:
```
vSym = Symbol('v', applyfunc=True)
L.replace(n(A,B), vSym)
```

[56]:
$$\lambda\left(\sigma_{apply}(v)\right)$$

[57]:
```
#L.replace(n(A,B), vSym).replace(sigmaApply, sigmaApply_)# ERROR because Symbol␣
 ↪has no atttribute applyfunc (that is the one we want though so must use matrix␣
 ↪symbol which for some reason works instead of just an ordinary symbol v
#V = MatrixSymbol()
# Takes in the symbols A and B matrices and returns the matrix symbol with the␣
 ↪shape that is supposed to result after A*B
V = lambda matA, matB: MatrixSymbol('V', matA.shape[0], matB.shape[1])
```

```
V
V(A,B)#.shape
```

[57]: V

[58]:
```
from sympy import symbols
#V = MatrixSymbol('V', X.shape[0], W.shape[1])
i, j = symbols('i j')
M = MatrixSymbol('M', i, j)# abstract shape
sigmaApply_L = Lambda(M, M.applyfunc(sigma))
lambda_L = Lambda(M, sum(M))
```

[59]:
```
sigmaApply_L(A)
```

[59]: $(d \mapsto \sigma(d))_{\circ}(X)$

[60]:
```
# TODO: trying to figure out how to write L so that it is in terms of lambdas so␣
 ↪get the form (d ---> sigma(d) COMPOAED ((X,W) -> V)) instead of␣
 ↪(sigmaApply(v(X,W)))
Vs = MatrixSymbol("Vs", A.shape[0], B.shape[1])
VL = Lambda((A,B), MatrixSymbol('V', A.shape[0], B.shape[1]))
VL
```

[60]: $((X,\ W) \mapsto V)$

[61]:
```
L.replace(n, VL)#.replace(sigmaApply, sigmaApply_L).subs({V:VL})
```

[61]: $\lambda\left(\sigma_{apply}(V)\right)$

[62]:
```
L.replace(n, VL).replace(sigmaApply, sigmaApply_)#.subs({VL(A,B) : n(A,B)}) ###␣
 ↪ERROR
# This is v(X,W) in Lambda form:
VL
```

[62]: $((X,\ W) \mapsto V)$

[63]:
```
VL(A,B)
#L.subs({n: V})
```

[63]: V

[64]:
```
L.replace(n(A,B), VL(A,B))#.replace(sigmaApply, sigmaApply_).subs({V(A,B) : n})
```

[64]: $\lambda\left(\sigma_{apply}(V)\right)$

[65]:
```
lambd(sigmaApply(VL))
```

[65]: $\lambda\left(\sigma_{apply}(((X,\ W) \mapsto V))\right)$

[66]:
```
lambd(sigmaApply(VL)).replace(VL, n(A,B))
```

[66]:
$$\lambda\left(\sigma_{apply}(v(X,W))\right)$$

[67]:
```
lambd(sigmaApply(VL)).diff(A)
```

[67]:
$$\frac{d}{d\xi_1}\sigma_{apply}(\xi_1)\Big|_{\xi_1=((X,\ W)\mapsto V)}\ \frac{d}{d\sigma_{apply}(((X,\ W)\mapsto V))}\lambda\left(\sigma_{apply}(((X,\ W)\mapsto V))\right)\frac{d}{dX}\left(((X,\ W)\mapsto V)\right)$$

[68]:
```
lambd(sigmaApply(VL)).diff(A).replace(VL, n(A,B))
```

[68]:
$$\frac{d}{d\xi_1}\sigma_{apply}(\xi_1)\Big|_{\xi_1=v(X,W)}\ \frac{\partial}{\partial\sigma_{apply}(v(X,W))}\lambda\left(\sigma_{apply}(v(X,W))\right)\frac{\partial}{\partial X}v(X,W)$$

[69]:
```
lambd(sigmaApply(VL))#.replace(sigmaApply, sigmaApply_)#replace(V, n(A,B)).
 ↪replace(sigmaApply, sigmaApply_)
```

[69]:
$$\lambda\left(\sigma_{apply}(((X,\ W)\mapsto V))\right)$$

[70]:
```
# GOAL: want both sigma_apply to be in ---> form composed with the above x,w␣
 ↪---> V form
#lambd(sigmaApply(V)).replace(V, Vs).replace(sigmaApply, sigmaApply_).
 ↪replace(Vs, V(A,B))### ERROR
lambd(sigmaApply(n(A,B))).replace(n(A,B), VL)
sigmaApply_(A)
sigmaApply_L(A)
```

[70]:
$$\left(d\mapsto\sigma(d)\right)_\circ(X)$$

[71]:
```
sigmaApply(Vs).replace(sigmaApply, sigmaApply_)
```

[71]:
$$\left(d\mapsto\sigma(d)\right)_\circ(Vs)$$

[72]:
```
sigmaApply(VL(A,B)).replace(sigmaApply, sigmaApply_)#.replace(V(A,B), V)#.
 ↪subs({sigmaApply: sigmaApply_L})
```

[72]:
$$\left(d\mapsto\sigma(d)\right)_\circ(V)$$

[73]:
```
#sigmaApply(Vs).subs({Vs : V, sigmaApply: sigmaApply_L}) ### ERROR must be␣
 ↪matrix instance
#sigmaApply(Vs).replace(sigmaApply , sigmaApply_L).subs({Vs : V})
#sigmaApply(V).replace(sigmaApply, sigmaApply_L)
```

[74]:
```
sa = Lambda((A,B), VL)
sa
```

[74]:
$$((X,\ W)\mapsto((X,\ W)\mapsto V))$$

[75]:
```
### ALTERNATE try of declaring a sigma-apply kind of function
#sas = Lambda((A,B), Vs.applyfunc(sigma))
```

[76]:
```
Lambda((A,B), sigma(VL))
```

[76]:
$$((X,\ W) \mapsto \sigma(((X,\ W) \mapsto V)))$$

[77]:
```
Lambda((A,B), sigma(VL)).diff(A) # nothing useful with this format, and␣
↪weird-wrong since doesn't do chain rule wi.r. to sigma
```

[77]:
$$\frac{d}{dX}\left(((X,\ W) \mapsto \sigma(((X,\ W) \mapsto V)))\right)$$

[78]:
```
Lambda((A,B), sigma(VL(A,B)))
```

[78]:
$$((X,\ W) \mapsto \sigma(V))$$

[79]:
```
sas = Lambda((A,B), VL(A,B).applyfunc(sigma))
sas
```

[79]:
$$((X,\ W) \mapsto (d \mapsto \sigma(d))_{\circ}(V))$$

[80]:
```
# YAY this works now I can replace MATRIX SYMBOLS with ordinary sympy LAMBDAS␣
↪(replace cano only replace same kind of thing / type)
sigma(Vs).subs(Vs, VL)
#
```

[80]:
$$\sigma(((X,\ W) \mapsto V))$$

[81]:
```
sas(A,B)
```

[81]:
$$(d \mapsto \sigma(d))_{\circ}(V)$$

[82]:
```
# A.applyfunc(sigma).subs(A, VL)# subs method doesn't work here with applyfunc
L
```

[82]:
$$\lambda\left(\sigma_{apply}(v(X,W))\right)$$

[83]:
```
#sas(A,B).replace(V, V(A,B))
```

[84]:
```
sigmaApply_L
```

[84]:
$$(M \mapsto (d \mapsto \sigma(d))_{\circ}(M))$$

[85]:
```
sigmaApply_L(M)
```

[85]:
$$(d \mapsto \sigma(d))_{\circ}(M)$$

[86]:
```
#sigmaApply_LFake = Lambda(M, M.applyfunc(sigma))
sigmaApply(M).replace(sigmaApply, sigmaApply_L)
```

[86]:
$$(d \mapsto \sigma(d))_{\circ}(M)$$

[87]:
```
#sigmaApply(M).replace(sigmaApply, sigmaApply_).subs(M, n(A,B))
n = Function("v", applyfunc=True)
#sigmaApply_(Vs.subs(Vs, Lambda((A,B), n(A,B))))
from sympy import lambdify
#sigma(lambdify([A,B], n(A,B)))
#inner = Lambda((A,B), n(A,B)); inner
#sigmaApply_(n(A,B))
#sigmaApply(inner).replace(sigmaApply, Lambda(A, sigma(A)))
```

[88]:
```
#sigmaApply_L(M).subs(M, inner)
Lambda(d, sigma(d))
```

[88]:
$$(d \mapsto \sigma(d))$$

[89]:
```
### CLOSEST ever gotten to function composition (?) with sympy ....
#Lambda(d, sigma(inner))
```

[90]:
```
#Lambda(d, sigma(inner)).diff(A)
```

[91]:
```
#Lambda(d, sigma(inner)).replace(inner, vL(A,B)).diff(A)
```

[92]:

[92]:

[92]:
```
# sigmaApply_L(M).subs(M, VL)# new subs method fails here too
#sigmaApply_(M).subs(M, VL)
```

[93]:
```
sigmaApply_L(M).diff(M)
```

[93]:
$$\left(d \mapsto \frac{d}{dd}\sigma(d)\right)_{\circ}(M)$$

[94]:

[94]:

[94]:
```
sigma(VL)#.replace(V, V(A,B))
```

[94]:
$$\sigma(((X,\ W) \mapsto V))$$

[95]:
```
sigma(VL).replace(VL, VL(A,B))
```

[95]:

$\sigma(V)$

[96]:
```
#sigma(V).replace(V, VL)
```

[97]:

[97]:

[97]:
```
f = Function('f')
xtoxL = Lambda(a, a)
xtox = lambda a: a
f(x).subs({x : xtoxL})
```

[97]: $f((x \mapsto x))$

[98]:
```
f(x).subs(x, xtox)# works but below one with replace doesn't. When replacing arg␣
 ↪with function uses SUBS without dictionary (instead of replace)
```

[98]: $f(x)$

[99]:
```
# f(x).replace(x, xtox)### ERROR xtox expects one positional argument ( I think␣
 ↪replace only replaces the same kind of thing, never for instance a matrix␣
 ↪symbol for a function or vice versa. the replacement needs to be of the same␣
 ↪type / kind. But Lambda seems to work (as above))
f(x).replace(x, xtoxL)
```

[99]: $f((x \mapsto x))$

[100]:

[100]:

[100]:
```
#lambd(sigmaApply(n(A,B))).replace(n(A,B), Vs).replace(sigmaApply, sigmaApply_).
 ↪replace(Vs, V)# ### ERROR rec replace must be matrix instance ....
```

[101]:

[101]:

[101]:

[101]:
```
### METHOD 0: the matrix diff rule in the most abstract form possible
n = Function("v", applyfunc=True) # necessary
L = lambd(sigmaApply(n(A,B)))
lambd_L = Lambda(A, sum(A))
lambd_L(A)
```

[101]: $X_{0,0} + X_{0,1} + X_{0,2} + X_{1,0} + X_{1,1} + X_{1,2} + X_{2,0} + X_{2,1} + X_{2,2}$

[102]:
```
lambd_L(sigmaApply(n(A,B)))#.replace(n, vL).replace(sigmaApply, sigmaApply_).
 ↪replace(lambd, lambd_L)
```

[102]:
$$\left(\sigma_{apply}(v(X,W))\right)_{0,0} + \left(\sigma_{apply}(v(X,W))\right)_{0,1} + \left(\sigma_{apply}(v(X,W))\right)_{0,2} + \left(\sigma_{apply}(v(X,W))\right)_{1,0} + \left(\sigma_{apply}(v(X,W))\right)_{1,1} + \left(\sigma_{apply}(v(X,W))\right)_{1,2} + \left(\sigma_{apply}(v(X,W))\right)_{2,0} + \left(\sigma_{apply}(v(X,W))\right)_{2,1} + \left(\sigma_{apply}(v(X,W))\right)_{2,2}$$

[103]:
```
L.replace(n,vL).replace(sigmaApply, sigmaApply_).diff(A)
```

[103]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)} \left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ (XW)\,W^T$$

[104]:
```
### SUCCESS! We see now that the matrix chain rule indeed makes the X transpose␣
 ↪factor out on the left!!! (while compared to the above, the matrix transpose␣
 ↪W^T factors out on the right, just like the book says (page 45 in the NOTE␣
 ↪section of Seth Weidman book))
L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B)
```

[104]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)} X^T\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ (XW)$$

[105]:
```
# Not showing ???
L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B).replace(lambd, lambd_L)
```

[105]:
$$\frac{d}{d\xi_1}\left(\xi_{10,0}+\xi_{10,1}+\xi_{10,2}+\xi_{11,0}+\xi_{11,1}+\xi_{11,2}+\xi_{12,0}+\xi_{12,1}+\xi_{12,2}\right)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)} X^T\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ (XW)$$

[106]:
```
#L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B).replace(B,W).
 ↪replace(A,X) # ## ERROR non commutative scalars in matrix
# L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B).replace(lambd, lambd_).
 ↪replace(B,W).replace(A,X)# ## ERROR dummy object not iterable
```

[107]:
```
#L.replace(n,vL).replace(sigmaApply, sigmaApply_).diff(A).replace(lambd, lambd_)␣
 ↪### ERROR: dummy object not iterable (probably means when in the above␣
 ↪expression we have epsilon = sigmaApply(XW) that we cannot iterate over this␣
 ↪expression)
```

[108]:
```
# Replacing lambda first: BAD
#L.replace(n, v).replace(lambd, lambd_) ## ERROR sigma apply object not ieterable
# Replacing sigma first: BAD
# L.replace(sigmaApply, sigmaApply_)### ERROR v object has no attribute applyfunc
```

[109]:
```
# Replacing n first: GOOD (need to go from inner nesting to outermost function,␣
 ↪never any other way)
L.replace(n, v).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_)
```

[109]:

$$\sigma(W_{0,0}X_{0,0}+W_{1,0}X_{0,1}+W_{2,0}X_{0,2}) \quad + \quad \sigma(W_{0,0}X_{1,0}+W_{1,0}X_{1,1}+W_{2,0}X_{1,2}) \quad +$$
$$\sigma(W_{0,0}X_{2,0}+W_{1,0}X_{2,1}+W_{2,0}X_{2,2}) \quad + \quad \sigma(W_{0,1}X_{0,0}+W_{1,1}X_{0,1}+W_{2,1}X_{0,2}) \quad +$$
$$\sigma(W_{0,1}X_{1,0}+W_{1,1}X_{1,1}+W_{2,1}X_{1,2})+\sigma(W_{0,1}X_{2,0}+W_{1,1}X_{2,1}+W_{2,1}X_{2,2})$$

[110]:
```
# ### END RESULT of METHOD 2:
L.replace(n, v).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_).
 ↪diff(Matrix(A))
```

[110]:
$$\begin{bmatrix} W_{0,0}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,0}X_{0,0}+W_{1,0}X_{0,1}+W_{2,0}X_{0,2}} + W_{0,1}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,1}X_{0,0}+W_{1,1}X_{0,1}+W_{2,1}X_{0,2}} & W_{1,0}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,0}X_{0,0}+W_{1,0}X_{0,1}} \\ W_{0,0}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,0}X_{1,0}+W_{1,0}X_{1,1}+W_{2,0}X_{1,2}} + W_{0,1}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,1}X_{1,0}+W_{1,1}X_{1,1}+W_{2,1}X_{1,2}} & W_{1,0}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,0}X_{1,0}+W_{1,0}X_{1,1}} \\ W_{0,0}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,0}X_{2,0}+W_{1,0}X_{2,1}+W_{2,0}X_{2,2}} + W_{0,1}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,1}X_{2,0}+W_{1,1}X_{2,1}+W_{2,1}X_{2,2}} & W_{1,0}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=W_{0,0}X_{2,0}+W_{1,0}X_{2,1}} \end{bmatrix}$$

Compare the above matrix symbol way with the Lsum way:

0.0.1 END RESULT of METHOD 1:

[111]:
```
#Lsum = L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_)
L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_).
 ↪subs(elemToSpecD).diff(X)#.subs(specToElemD)
```

[111]:
$$\begin{bmatrix} w_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}} + w_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}} & w_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}} + w_{2} \\ w_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}} + w_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}} & w_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}} + w_{2} \\ w_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}} + w_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}} & w_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}} + w_{2} \end{bmatrix}$$

[112]:
```

```

COMPARING METHOD 0 (abstract way) with METHOD 2 (direct way) when differentiating .w.r.t to X vs. w.r.t to W ### With respect to X (abstract)

[112]:
```
L.replace(n,vL).replace(sigmaApply, sigmaApply_).diff(A)
```

[112]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)}\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ(XW)\,W^T$$

0.0.2 With respect to X (direct)

[113]:
```
L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_).
 ↪subs(elemToSpecD).diff(X).subs(specToElemD)
```

[113]:
$$\begin{bmatrix} w_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{11}} + w_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{12}} & w_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{11}} + w_{22}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{12}} & w_{31}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{11}} + w_{32}\frac{d}{d\xi} \\ w_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{21}} + w_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{22}} & w_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{21}} + w_{22}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{22}} & w_{31}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{21}} + w_{32}\frac{d}{d\xi} \\ w_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{31}} + w_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{32}} & w_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{31}} + w_{22}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{32}} & w_{31}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{31}} + w_{32}\frac{d}{d\xi} \end{bmatrix}$$

0.0.3 With respect to W (abstract)

[114]:
```
L.replace(n,vL).replace(sigmaApply, sigmaApply_).diff(B)
```

[114]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)} X^T\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ(XW)$$

0.0.4 With respect to W (direct)

[115]:
```
L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_).
↪subs(elemToSpecD).diff(W).subs(specToElemD)
```

[115]:
$$\begin{bmatrix} x_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{11}} + x_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{21}} + x_{31}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{31}} & x_{11}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{12}} + x_{21}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{22}} + x_{31}\frac{d}{d\xi_1}\sigma \\ x_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{11}} + x_{22}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{21}} + x_{32}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{31}} & x_{12}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{12}} + x_{22}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{22}} + x_{32}\frac{d}{d\xi_1}\sigma \\ x_{13}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{11}} + x_{23}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{21}} + x_{33}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{31}} & x_{13}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{12}} + x_{23}\frac{d}{d\xi_1}\sigma(\xi_1)\Big|_{\xi_1=n_{22}} + x_{33}\frac{d}{d\xi_1}\sigma \end{bmatrix}$$

[116]:

[116]:

[116]:

[116]:

0.0.5 NEXT: try to substitute the X, W matrices step by step to see if you can come to the same result as the direct forms above (from method 2 or 1)

[116]:
```
from sympy import simplify, expand
#simplify(L.replace(n,vL).replace(sigmaApply, sigmaApply_).diff(B).subs({A:X, B:
↪W})) ### ERROR max recursion depth exceeded
L.replace(n,vL).replace(sigmaApply, sigmaApply_).diff(B).subs({A:X, B:W})
```

[116]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ\left(\begin{bmatrix}x_{11}&x_{12}&x_{13}\\x_{21}&x_{22}&x_{23}\\x_{31}&x_{32}&x_{33}\end{bmatrix}\begin{bmatrix}w_{11}&w_{12}\\w_{21}&w_{22}\\w_{31}&w_{32}\end{bmatrix}\right)} \left(\begin{bmatrix}x_{11}&x_{12}&x_{13}\\x_{21}&x_{22}&x_{23}\\x_{31}&x_{32}&x_{33}\end{bmatrix}\right)^T \left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ \left(\begin{bmatrix}x_{11}&x_{12}&x_{13}\\x_{21}&x_{22}&x_{23}\\x_{31}&x_{32}&x_{33}\end{bmatrix}\begin{bmatrix}w$$

[117]:
```
L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B)#.subs({A:X, B:W})
```

[117]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)} X^T\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ(XW)$$

[118]:
```
L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B)
```

[118]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)} X^T\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ(XW)$$

[119]:
```
#L.replace(n,v).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_)
```

[120]:
```
#L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(B).subs({A:X, B:W}).
 ↪replace(lambd, lambd_) ### ERROR dummy object not iterable
L.replace(n,v).diff(A)
```

[120]:
$$\frac{d}{d\xi_1}\sigma_{apply}(\xi_1)\Big|_{\xi_1=XW}\frac{\partial}{\partial\sigma_{apply}(XW)}\lambda(\sigma_{apply}(XW))\frac{\partial}{\partial X}XW$$

[121]:

[121]:
```
#L.replace(n,v).replace(sigmaApply, sigmaApply_).diff(A).
 ↪replace(A,Matrix(A))##ERROR noncommutative matrix scalars
# WANT: to be able to do diff and have the expression come out as above with X^T␣
 ↪on left and W^T on right, when using just this form, with abstract form v:
L.replace(A,A.T).replace(B,B.T)
```

[121]:
$$\lambda\left(\sigma_{apply}\left(v\left(X^T,W^T\right)\right)\right)$$

[122]:
```
# Error if applying sigma to the v function because it sais v has no attribute␣
 ↪applyfunc to trying now to making it have the attriute applyfunc.
y = Function('y', applyfunc=True, real=True)
```

[123]:
```
Ly = lambd(sigmaApply(y(A,B)))
Ly
```

[123]:
$$\lambda(\sigma_{apply}(y(X,W)))$$

[124]:
```
Ly.replace(A,A.T).replace(B,B.T)#.replace(sigmaApply, sigmaApply_)
```

[124]:
$$\lambda\left(\sigma_{apply}\left(y\left(X^T,W^T\right)\right)\right)$$

[125]:
```
# TODO next step: to apply the sigma to get that applied functor expression but␣
 ↪here get error saying bol object not callable ...??
Ly.replace(A,A.T).replace(B,B.T)#.replace(sigmaApply, sigmaApply_)
```

[125]:
$$\lambda\left(\sigma_{apply}\left(y\left(X^T,W^T\right)\right)\right)$$

[126]:
```
# TODO always get fatal python error here, as if it can't deal with two matrix␣
 ↪args!!
#Ly.replace(A,A.T).replace(B,B.T).diff(A)
#siga2 = Lambda(a, siga(a))
```

[127]:
```
Ly.replace(A, A.T).replace(B, b).diff(b)#.replace(sigmaApply, siga)
```

[127]:
$$\frac{\partial}{\partial\sigma_{apply}(y(X^T,b))}\lambda\left(\sigma_{apply}\left(y\left(X^T,b\right)\right)\right)\frac{\partial}{\partial y(X^T,b)}\sigma_{apply}\left(y\left(X^T,b\right)\right)\left(\frac{\partial}{\partial b}y\left(X^T,b\right)+\right)$$

[128]:
```
L.replace(n, vN).replace(sigmaApply, sigmaApply_).subs(elemToMatArgD)
```

[128]:
$$\lambda\left(\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}\right)$$

[129]:
```
#L.replace(n, vN).replace(sigmaApply, sigmaApply_).subs(elemToMatArgD).diff(A)##␣
 ↪ERROR: max recursion depth eceeded
L.replace(n, vN).replace(sigmaApply, sigmaApply_).subs(elemToMatArgD).
 ↪diff(Matrix(3,2,list(elemToMatArgD.values())))
```

[129]:
$$\begin{bmatrix}\begin{bmatrix}\frac{\partial}{\partial \mathrm{n}_{11}(X,W)}\sigma(\mathrm{n}_{11}(X,W)) & \frac{\partial}{\partial\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}}\lambda\left(\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}\right) & 0\\ 0 & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & \overline{\partial \mathrm{n}_{12}}\\ 0 & 0\\ 0 & 0\end{bmatrix}\\ \begin{bmatrix}\frac{\partial}{\partial \mathrm{n}_{21}(X,W)}\sigma(\mathrm{n}_{21}(X,W)) & \frac{\partial}{\partial\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}}\lambda\left(\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}\right) & 0\\ 0 & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & \overline{\partial \mathrm{n}_{22}}\\ 0 & 0\\ 0 & 0\end{bmatrix}\\ \begin{bmatrix}\frac{\partial}{\partial \mathrm{n}_{31}(X,W)}\sigma(\mathrm{n}_{31}(X,W)) & \frac{\partial}{\partial\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}}\lambda\left(\begin{bmatrix}\sigma(\mathrm{n}_{11}(X,W)) & \sigma(\mathrm{n}_{12}(X,W))\\ \sigma(\mathrm{n}_{21}(X,W)) & \sigma(\mathrm{n}_{22}(X,W))\\ \sigma(\mathrm{n}_{31}(X,W)) & \sigma(\mathrm{n}_{32}(X,W))\end{bmatrix}\right) & 0\\ 0 & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0 & \overline{\partial \mathrm{n}_{32}}\\ 0 & 0\\ 0 & 0\end{bmatrix}\end{bmatrix}$$

[130]:
```
A.applyfunc(sigma)
```

[130]:
$$\left(d \mapsto \sigma(d)\right)_{\circ}(X)$$

[131]:
```
sigma = Function("sigma", applyfunc=True, bool=False)
```

[132]:
```
sigma.__dict__
```

[132]:
```
mappingproxy({'applyfunc': True,
              'bool': False,
              '_kwargs': {'applyfunc': True, 'bool': False},
              '__module__': None,
              '__doc__': None,
              'name': 'sigma',
              '_sage_': <sympy.core.function.UndefSageHelper at 0x7f52b8568e50>,
              '_nargs': None,
```

```
                '__sympy__': <property at 0x7f52a2dc6050>,
                '_explicit_class_assumptions': {},
                'default_assumptions': {},
                '_prop_handler': {'negative': <function
sympy.core.expr.Expr._eval_is_negative(self)>,
                 'positive': <function
sympy.core.expr.Expr._eval_is_positive(self)>,
                 'commutative': <function
sympy.core.function.Function._eval_is_commutative(self)>,
                 'extended_positive': <function
sympy.core.expr.Expr._eval_is_extended_positive(self)>,
                 'extended_negative': <function
sympy.core.expr.Expr._eval_is_extended_negative(self)>}})
```

[133]:
```
Ly = lambd(sigmaApply(y(A,B))); Ly
```

[133]: $\lambda\left(\sigma_{apply}(y(X,W))\right)$

[134]:
```
(X*W).applyfunc(sigma)
```

[134]: $\begin{bmatrix} \sigma(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}) & \sigma(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}) \\ \sigma(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}) & \sigma(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}) \\ \sigma(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}) & \sigma(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}) \end{bmatrix}$

[135]:
```
(A*B).applyfunc(sigma)
```

[135]: $(d \mapsto \sigma(d))_{\circ}(XW)$

[136]:
```
siga(A)
#A.applyfunc(siga) ### ERROR dumy object has no attribute shape
```

[136]: $\begin{bmatrix} \sigma(X_{0,0}) & \sigma(X_{0,1}) & \sigma(X_{0,2}) \\ \sigma(X_{1,0}) & \sigma(X_{1,1}) & \sigma(X_{1,2}) \\ \sigma(X_{2,0}) & \sigma(X_{2,1}) & \sigma(X_{2,2}) \end{bmatrix}$

[137]:
```
y = Function("y", applyfunc = True, bool=False, shape=(3,3))
y.shape
```

[137]: (3, 3)

[138]:
```
# siga(y(A,B))### ERROR: function y is not subscriptable
```

[139]:

[139]:

[139]:

[139]:

[139]:
```
Ly.subs({A:a,B:b}).diff(b).subs({a:A, b:B})#.replace(sigmaApply, sigmaApply_)
```

[139]:
$$\frac{\partial}{\partial \sigma_{apply}(y(X,W))}\lambda\left(\sigma_{apply}(y(X,W))\right)\frac{\partial}{\partial y(X,W)}\sigma_{apply}(y(X,W))\frac{\partial}{\partial W}y(X,W)$$

[140]:
```
L.replace(A,a).replace(B,b).diff(b).subs({a:A,b:B})#.replace(sigmaApply,␣
↪sigmaApply_)#.diff(b)
```

[140]:
$$\frac{\partial}{\partial \sigma_{apply}(v(X,W))}\lambda\left(\sigma_{apply}(v(X,W))\right)\frac{\partial}{\partial v(X,W)}\sigma_{apply}(v(X,W))\frac{\partial}{\partial W}v(X,W)$$

[141]:
```
sigma = Function("sigma", applyfunc=True, real=True)
sigmaApply_ = lambda mat: mat.applyfunc(sigma)
L = lambd(sigmaApply(n(A,B)))
#L.replace(A,a).replace(B,b).diff(b).subs({a:A,b:B}).replace(sigmaApply,␣
↪sigmaApply_)
L.replace(n, v).replace(sigmaApply, sigmaApply_).diff(A)
#m = Symbol("m", shape=(3,2))
#m.shape
#sigmaApply_3 = Lambda(m, siga(m))
#L.replace(A,a).replace(B,b).diff(b).replace(b,B).replace(a,A).subs({n:vL}).
↪replace(sigmaApply, sigmaApply_2) ### ERROR: Dummy object has no attribute␣
↪shape
```

[141]:
$$\frac{d}{d\xi_1}\lambda(\xi_1)\Big|_{\xi_1=(d\mapsto\sigma(d))_\circ(XW)}\left(d\mapsto\frac{d}{dd}\sigma(d)\right)_\circ(XW)\,W^T$$

[142]:
```
# Ly.replace(B, b).diff(A)#.replace(sigmaApply, siga)### ERROR noncommutative␣
↪matrix scalars not supported
Ly.replace(A, A.T).replace(B, b).diff(b).replace(b, B).replace(A.T, A)#.
↪replace(sigmaApply, siga)
```

[142]:
$$\frac{\partial}{\partial \sigma_{apply}(y(X,W))}\lambda\left(\sigma_{apply}(y(X,W))\right)\frac{\partial}{\partial y(X,W)}\sigma_{apply}(y(X,W))\left(\frac{\partial}{\partial W}y(X,W)+\right)$$

[143]:
```
#Ly.replace(B,b).diff(b).replace(b,B) ### ERROR
```

[144]:
```
# NEXT: try to replace the sigma apply, not working
n.__dict__
```

[144]:
```
mappingproxy({'applyfunc': True,
              '_kwargs': {'applyfunc': True},
              '__module__': None,
              '__doc__': None,
              'name': 'v',
              '_sage_': <sympy.core.function.UndefSageHelper at 0x7f52b8568e50>,
```

```
                '_nargs': None,
                '__sympy__': <property at 0x7f52a2fefe90>,
                '_explicit_class_assumptions': {},
                'default_assumptions': {},
                '_prop_handler': {'negative': <function
sympy.core.expr.Expr._eval_is_negative(self)>,
                 'positive': <function
sympy.core.expr.Expr._eval_is_positive(self)>,
                 'commutative': <function
sympy.core.function.Function._eval_is_commutative(self)>,
                 'extended_positive': <function
sympy.core.expr.Expr._eval_is_extended_positive(self)>,
                 'extended_negative': <function
sympy.core.expr.Expr._eval_is_extended_negative(self)>}})
```

[145]:
```
y.__dict__
# TODO HERE
#https://stackoverflow.com/questions/12614334/
 ↪typeerror-bool-object-is-not-callable
```

[145]:
```
mappingproxy({'applyfunc': True,
              'bool': False,
              'shape': (3, 3),
              '_kwargs': {'applyfunc': True, 'bool': False, 'shape': (3, 3)},
              '__module__': None,
              '__doc__': None,
              'name': 'y',
              '_sage_': <sympy.core.function.UndefSageHelper at 0x7f52b8568e50>,
              '_nargs': None,
              '__sympy__': <property at 0x7f52a2e15530>,
              '_explicit_class_assumptions': {},
              'default_assumptions': {},
              '_prop_handler': {'negative': <function
sympy.core.expr.Expr._eval_is_negative(self)>,
               'positive': <function
sympy.core.expr.Expr._eval_is_positive(self)>,
               'commutative': <function
sympy.core.function.Function._eval_is_commutative(self)>,
               'extended_positive': <function
sympy.core.expr.Expr._eval_is_extended_positive(self)>,
               'extended_negative': <function
sympy.core.expr.Expr._eval_is_extended_negative(self)>}})
```

[146]:

[146]:

[146]:
```
from sympy import diff
# ### WARNING: this only works when size(X) == size(Y) else since size(W) !=␣
 ↪size(X) cannot subst B with W, so this operation won't work in my case.
#X = Matrix(3,3, lambda i,j: Symbol("x_{}{}".format(i+1,j+1))); Matrix(X)
# Create another matrix instead of W so that it matches size of X during diff(X)␣
 ↪operation, since otherwise the diff by X doesn't work, says X and W need to be␣
 ↪same size.
Wtemp = Matrix(*X.shape, lambda i,j: Symbol("t_{}{}".format(i+1,j+1)));␣
 ↪Matrix(Wtemp)
```

[146]:
$$\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

[147]:
```
#L.subs({A:X, B:Wtemp}).diff(X)[0,0][0,0].replace(n,vN).replace(sigmaApply,␣
 ↪sigmaApply_)#.doit()
#diff(L.replace(A,A.T), A) # ERROR max recursion depth exceeded
```

[148]:
```
#Lmat = L.subs({A:X, B:Wtemp}).diff(X).subs({X:A, Wtemp: B}); Lmat #replace(X,␣
 ↪A).replace(Y,B); Lmat
# NOTE need to do replace at the end (instead of subs) else it says unhasable␣
 ↪type mutabledensematrix.
Lmat = L.subs({A:X, B:Wtemp}).diff(X).replace(X, A).replace(Wtemp,B); Lmat
#L.diff(A) # HELL ON THE EDITOR NEVER TRY THIS AGAIN
```

[148]:
$$\begin{bmatrix} \begin{bmatrix} \frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X,W)) \frac{\partial}{\partial \sigma_{apply}(v(X,W))} \lambda\left(\sigma_{apply}(v(X,W))\right) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X, \\ 0 & \\ 0 & \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ \frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X,W)) \frac{\partial}{\partial \sigma_{apply}(v(X,W))} \lambda\left(\sigma_{apply}(v(X,W))\right) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X, \\ 0 & \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X,W)) \frac{\partial}{\partial \sigma_{apply}(v(X,W))} \lambda\left(\sigma_{apply}(v(X,W))\right) & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X, \end{bmatrix} \end{bmatrix}$$

[149]:
```
#L.replace(A,X).replace(B,W)
```

[150]:
```
# Method 2 approach for comparison:
#L.replace(n, vN).replace(sigmaApply, sigmaApply_).replace(lambd, lambd_).
 ↪subs(elemToSpecD).diff(X)#.subs(specToElemD)
```

[151]:
```
elem = Lmat[0,0][0,0];elem
```

[151]:
$$\frac{\partial}{\partial X} v(X,W) \frac{\partial}{\partial v(X,W)} \sigma_{apply}(v(X,W)) \frac{\partial}{\partial \sigma_{apply}(v(X,W))} \lambda\left(\sigma_{apply}(v(X,W))\right)$$

[152]:
```
#Lmat.replace(n, vL) # error can't calc deriv .w.r.t to x11*w11 +...
# Lmat.replace(n, v) # error can't calc deriv .w.r.t to x11*w11 +...
elem.subs(n, vL)
```

[152]:
$$\frac{d}{d\xi_0}\sigma_{apply}(\xi_0)\Big|_{\xi_0=XW}\frac{\partial}{\partial X}XW\frac{\partial}{\partial\sigma_{apply}(XW)}\lambda\left(\sigma_{apply}(XW)\right)$$

[153]:
```
#elem.replace(n, v) # error cannot deriv wrt to X*W
```

[154]:
```
Selem
```

[154]:
$$\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}$$

[155]:
```
# use replace n with vN instead of subs n with vL to get less specific output so␣
 ↪it is easier to see since vL returns the xww*w11 +.... expressions
elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_)
```

[155]:
$$\frac{\partial}{\partial\begin{bmatrix}x_{11} & x_{12} & x_{13}\\ x_{21} & x_{22} & x_{23}\\ x_{31} & x_{32} & x_{33}\end{bmatrix}}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}}\lambda\left(\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}\right)$$

[156]:
```
# Making matrix symbols again
Ss = MatrixSymbol('S', 3,2) #n by p
Ns = MatrixSymbol('N', 3,2) #n by p
short = elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
 ↪replace(X,A).replace(Nelem, Ns).replace(Selem ,Ss)
short
```

[156]:
$$\frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}}\lambda\left(\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}\right)$$

[157]:
```
# Now going back to matrix form just to apply the last function LAMBDA
elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
 ↪replace(lambd, lambd_)
```

[157]:
$$\frac{\partial}{\partial\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix}}\left(\sigma(n_{11})+\sigma(n_{12})+\sigma(n_{21})+\sigma(n_{22})+\sigma(n_{31})+\sigma(n_{32})\right)\frac{\partial}{\partial\begin{bmatrix}x_{11} & x_{12} & x_{13}\\ x_{21} & x_{22} & x_{23}\\ x_{31} & x_{32} & x_{33}\end{bmatrix}}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11}\\ n_{21}\\ n_{31}\end{bmatrix}}$$

[158]:
```
# Making each of the n_ijs a function
#elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
 ↪replace(lambd, lambd_).subs(elemToSpecD)
Matrix(elemToSpecFuncArgs)
```

[158]:
$$\begin{bmatrix} n_{11} & \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{12} & \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{21} & \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{22} & \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{31} & \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \\ n_{32} & \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) \end{bmatrix}$$

[159]:
```
long = elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
 ↪replace(lambd, lambd_).subs(elemToSpecFuncArgsD)
long
```

[159]:
$$\frac{\partial}{\partial \begin{bmatrix} \sigma(\mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right)) & \sigma(\mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32} \\ \sigma(\mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right)) & \sigma(\mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32} \\ \sigma(\mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right)) & \sigma(\mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32} \end{bmatrix}}$$

[160]:
```
# short version again:
short
```

[160]:
$$\frac{d}{dX} N \frac{\partial}{\partial N} \begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix} \frac{\partial}{\partial \begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}} \lambda\left(\begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}\right)$$

[161]:
```
# long.doit() # error as base exp thing
```

[162]:
```
# Trying step by step replacement approach:
elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
 ↪replace(lambd, lambd_).replace(Nelem, Ns).replace(X,A)
```

[162]:
$$\frac{\partial}{\partial \begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}} \left(\sigma(n_{11}) + \sigma(n_{12}) + \sigma(n_{21}) + \sigma(n_{22}) + \sigma(n_{31}) + \sigma(n_{32})\right) \frac{d}{dX} N \frac{\partial}{\partial N} \begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[163]:
```
# Seeing if replacing the order of replacing Ns matrix with Xs matrix makes a␣
 ↪difference: ...
step = elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
 ↪replace(lambd, lambd_).replace(Nelem, Ns).replace(X,A).doit()
step
```

[163]:

$$\begin{bmatrix} \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \\ \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \\ \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \end{bmatrix}$$

[164]:
```
elem.subs({A:X, B:W}).replace(n, vN).replace(sigmaApply, sigmaApply_).
↪replace(lambd, lambd_).replace(X,A).replace(Nelem, Ns).doit()
```

[164]:
$$\begin{bmatrix} \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \\ \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \\ \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{d}{dX}N\frac{\partial}{\partial N}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \end{bmatrix}$$

[165]:
```
step.replace(Ns, Nelem)
```

[165]:
$$\begin{bmatrix} \frac{\partial}{\partial X}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{\partial}{\partial X}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \\ \frac{\partial}{\partial X}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{\partial}{\partial X}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \\ \frac{\partial}{\partial X}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} & \frac{\partial}{\partial X}\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}\frac{\partial}{\partial\begin{bmatrix}n_{11} & n_{12}\\ n_{21} & n_{22}\\ n_{31} & n_{32}\end{bmatrix}}\begin{bmatrix}\sigma(n_{11}) & \sigma(n_{12})\\ \sigma(n_{21}) & \sigma(n_{22})\\ \sigma(n_{31}) & \sigma(n_{32})\end{bmatrix} \end{bmatrix}$$

[166]:
```
#step.replace(Ns, Nelem).replace(A,X).doit()#error immutable dense array has no␣
↪attribute as base exp ...
elem2 = step[0,0].replace(Ns, Nelem)
elem2.replace(A,X).subs(elemToSpecFuncArgsD)
```

[166]:

$$\frac{\partial}{\partial \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}} \begin{bmatrix} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, \right. \\ \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, \right. \\ \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, \right. \end{bmatrix}$$

[167]:
```
#elem2.replace(A,X).subs(elemToSpecFuncArgsD).doit()
F = Nelem.subs(elemToSpecFuncArgsD); F
```

[167]:
$$\begin{bmatrix} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{1}\right. \\ \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{1}\right. \\ \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{1}\right. \end{bmatrix}$$

[168]:
```
F[0,0].diff(X[0,0])
```

[168]:
$$\frac{\partial}{\partial x_{11}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right)$$

[169]:
```
F[0,0].diff(X)
```

[169]:
$$\begin{bmatrix} \frac{\partial}{\partial x_{11}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{12}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}\right. \\ \frac{\partial}{\partial x_{21}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{22}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}\right. \\ \frac{\partial}{\partial x_{31}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{32}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}\right. \end{bmatrix}$$

[170]:
```
F.diff(X)
```

[170]:
$$\begin{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_{11}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{11}} \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \\ \frac{\partial}{\partial x_{11}} \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{11}} \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \\ \frac{\partial}{\partial x_{11}} \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{11}} \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \end{bmatrix} \\ \begin{bmatrix} \frac{\partial}{\partial x_{21}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{21}} \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \\ \frac{\partial}{\partial x_{21}} \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{21}} \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \\ \frac{\partial}{\partial x_{21}} \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{21}} \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \end{bmatrix} \\ \begin{bmatrix} \frac{\partial}{\partial x_{31}} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{31}} \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \\ \frac{\partial}{\partial x_{31}} \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{31}} \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \\ \frac{\partial}{\partial x_{31}} \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \frac{\partial}{\partial x_{31}} \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x\right. \end{bmatrix} \end{bmatrix}$$

[171]:
```
argsToSpecD = dict(zip(elemToSpecFuncArgsD.values(), elemToSpecD.values()))
argsToSpec = list(argsToSpecD.items())
Matrix(argsToSpec)
```

[171]:
$$\begin{bmatrix} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13} \\ \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13} \\ \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23} \\ \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23} \\ \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33} \\ \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33} \end{bmatrix}$$

[172]:
```
F[0,0].diff(X[0,0]).subs(argsToSpecD)#.subs({elemToSpecFuncArgs[0][1] :␣
↪Nspec[0,0]})
```

[172]:
$$\frac{\partial}{\partial x_{11}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right)$$

[173]:
```
F[0,0].diff(X[0,0]).subs(argsToSpecD).doit()
```

[173]:
$$w_{11}$$

[174]:
```
# NOTE: using diff did not work, said immutable dense array cannot be subs-ed
derive_by_array(F, X).subs(argsToSpecD)
```

[174]:
$$\left[\begin{bmatrix} \frac{\partial}{\partial x_{11}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right) & \frac{\partial}{\partial x_{11}}\left(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}\right) \\ \frac{\partial}{\partial x_{11}}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right) & \frac{\partial}{\partial x_{11}}\left(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}\right) \\ \frac{\partial}{\partial x_{11}}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right) & \frac{\partial}{\partial x_{11}}\left(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}\right) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_{12}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right) & \frac{\partial}{\partial x_{12}}\left(w_{1}\right. \\ \frac{\partial}{\partial x_{12}}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right) & \frac{\partial}{\partial x_{12}}\left(w_{1}\right. \\ \frac{\partial}{\partial x_{12}}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right) & \frac{\partial}{\partial x_{12}}\left(w_{1}\right. \end{bmatrix} \right.$$
$$\left.\begin{bmatrix} \frac{\partial}{\partial x_{21}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right) & \frac{\partial}{\partial x_{21}}\left(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}\right) \\ \frac{\partial}{\partial x_{21}}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right) & \frac{\partial}{\partial x_{21}}\left(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}\right) \\ \frac{\partial}{\partial x_{21}}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right) & \frac{\partial}{\partial x_{21}}\left(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}\right) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_{22}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right) & \frac{\partial}{\partial x_{22}}\left(w_{1}\right. \\ \frac{\partial}{\partial x_{22}}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right) & \frac{\partial}{\partial x_{22}}\left(w_{1}\right. \\ \frac{\partial}{\partial x_{22}}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right) & \frac{\partial}{\partial x_{22}}\left(w_{1}\right. \end{bmatrix} \right.$$
$$\left.\begin{bmatrix} \frac{\partial}{\partial x_{31}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right) & \frac{\partial}{\partial x_{31}}\left(w_{12}x_{11}+w_{22}x_{12}+w_{32}x_{13}\right) \\ \frac{\partial}{\partial x_{31}}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right) & \frac{\partial}{\partial x_{31}}\left(w_{12}x_{21}+w_{22}x_{22}+w_{32}x_{23}\right) \\ \frac{\partial}{\partial x_{31}}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right) & \frac{\partial}{\partial x_{31}}\left(w_{12}x_{31}+w_{22}x_{32}+w_{32}x_{33}\right) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_{32}}\left(w_{11}x_{11}+w_{21}x_{12}+w_{31}x_{13}\right) & \frac{\partial}{\partial x_{32}}\left(w_{1}\right. \\ \frac{\partial}{\partial x_{32}}\left(w_{11}x_{21}+w_{21}x_{22}+w_{31}x_{23}\right) & \frac{\partial}{\partial x_{32}}\left(w_{1}\right. \\ \frac{\partial}{\partial x_{32}}\left(w_{11}x_{31}+w_{21}x_{32}+w_{31}x_{33}\right) & \frac{\partial}{\partial x_{32}}\left(w_{1}\right. \end{bmatrix} \right]$$

[175]:
```
derive_by_array(F, X).subs(argsToSpecD).doit()
```

[175]:
$$\left[\begin{bmatrix} w_{11} & w_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_{21} & w_{22} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_{31} & w_{32} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right.$$
$$\left.\begin{bmatrix} 0 & 0 \\ w_{11} & w_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ w_{21} & w_{22} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ w_{31} & w_{32} \\ 0 & 0 \end{bmatrix} \right.$$
$$\left.\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{11} & w_{12} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ w_{31} & w_{32} \end{bmatrix} \right]$$

[176]:
```
derive_by_array(F, W).subs(argsToSpecD).doit()
```

[176]:
$$\left[\begin{bmatrix} x_{11} & 0 \\ x_{21} & 0 \\ x_{31} & 0 \end{bmatrix} \begin{bmatrix} 0 & x_{11} \\ 0 & x_{21} \\ 0 & x_{31} \end{bmatrix} \right.$$
$$\left.\begin{bmatrix} x_{12} & 0 \\ x_{22} & 0 \\ x_{32} & 0 \end{bmatrix} \begin{bmatrix} 0 & x_{12} \\ 0 & x_{22} \\ 0 & x_{32} \end{bmatrix} \right.$$
$$\left.\begin{bmatrix} x_{13} & 0 \\ x_{23} & 0 \\ x_{33} & 0 \end{bmatrix} \begin{bmatrix} 0 & x_{13} \\ 0 & x_{23} \\ 0 & x_{33} \end{bmatrix} \right]$$

[177]:
```
elem2
```

[177]:
$$\frac{\partial}{\partial X}\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \\ n_{31} & n_{32} \end{bmatrix} \frac{\partial}{\partial \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \\ n_{31} & n_{32} \end{bmatrix}} \begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[178]:
```
funcMat = elem2.subs(elemToSpecFuncArgsD).replace(A,X)#.diff(X)
funcMat
```

[178]:
$$\frac{\partial}{\partial \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}} \begin{bmatrix} \mathrm{n}_{11}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{12}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, \right. \\ \mathrm{n}_{21}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{22}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, \right. \\ \mathrm{n}_{31}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}\right) & \mathrm{n}_{32}\left(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, \right. \end{bmatrix}$$

[179]:
```
#funcMat.doit() # error
#derive_by_array(funcMat, X)
```

[180]:
```
funcMat = elem2.subs(elemToSpecFuncD).replace(A,X)#.diff(X)
funcMat
```

[180]:
$$\frac{\partial}{\partial \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}} \begin{bmatrix} \mathrm{n}_{11}\left(w_{11}x_{11} + w_{21}x_{12} + w_{31}x_{13}\right) & \mathrm{n}_{12}\left(w_{12}x_{11} + w_{22}x_{12} + w_{32}x_{13}\right) \\ \mathrm{n}_{21}\left(w_{11}x_{21} + w_{21}x_{22} + w_{31}x_{23}\right) & \mathrm{n}_{22}\left(w_{12}x_{21} + w_{22}x_{22} + w_{32}x_{23}\right) \\ \mathrm{n}_{31}\left(w_{11}x_{31} + w_{21}x_{32} + w_{31}x_{33}\right) & \mathrm{n}_{32}\left(w_{12}x_{31} + w_{22}x_{32} + w_{32}x_{33}\right) \end{bmatrix} \frac{}{\partial \begin{bmatrix} \mathrm{n}_{11}\left(w_{11}x_{11} + w_{21}x_{12} + w \right. \\ \mathrm{n}_{21}\left(w_{11}x_{21} + w_{21}x_{22} + w \right. \\ \mathrm{n}_{31}\left(w_{11}x_{31} + w_{21}x_{32} + w \right. \end{bmatrix}}$$

[181]:
```
#funcMat.doit() # same error
#elem2.subs(elemToSpecFuncD).doit() # error
elem2
```

[181]:
$$\frac{\partial}{\partial X}\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \\ n_{31} & n_{32} \end{bmatrix} \frac{\partial}{\partial \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \\ n_{31} & n_{32} \end{bmatrix}} \begin{bmatrix} \sigma(n_{11}) & \sigma(n_{12}) \\ \sigma(n_{21}) & \sigma(n_{22}) \\ \sigma(n_{31}) & \sigma(n_{32}) \end{bmatrix}$$

[182]:
```
# elem2.replace(A,X).doit() # error
```

[183]:
```
#elem2.replace(A,a).doit()#.subs(elemToSpecFuncArgsD).doit()
# ERROR everywhere what next todo? this approach worked before, where I make w.r.
 ↪t. thing a real matrix, and leave the others a symbol so why isn't it working␣
 ↪now?
```

[184]:
```
#elem2.replace(A,X).subs(elemToSpecFuncD).doit()
# ERROR this has to work though! Then can simply replace n_ijs with lambda
```

[185]:
```
#elem2.subs(elemToMatArgD).doit()#ERROR max recursion depth exceeeded
```