CasADi tutorial 3

```
#
#
#
#
#
#
#
```

This tutorial file explains the use of CasADi's MX in a python context.

```
from numpy import *
from casadi import *
```

Let's start with some algebra

```
x = MX.sym("x",2,3)
y = MX.sym("y",3,2)
print x
```

```
x
```

```
for i in range(6):
        print x.nz[i]
```

```
x[0]
x[1]
x[2]
x[3]
x[4]
x[5]
```

```

for i in range(2):
        for j in range(3):
                print "x[%d,%d] = %s" % (i,j,str(x[i,j]))
```

```
x[0,0] = x[0]
x[0,1] = x[2]
x[0,2] = x[4]
x[1,0] = x[1]
x[1,1] = x[3]
x[1,2] = x[5]
```

```

print x[1,1]
```

```
x[3]
```

```
print x.nz[3] # Note that index is flattened. x[0,0] is illegal.
```

```
x[3]
```

```
print norm_2(x)
```

```
||x||_2
```

```
z= mtimes(x,y)
print z
```

```
(zeros(2x2, dense)+mtimes(x, y))
```

Note how the operations on MXes are lazy on the matrix level. Any elementwise logic is postponed until evaluation demands it. Just like, SX functions, MX functions can be single or multi input/output. The only allowed input/output primitive is MX.

```
f = Function('f', [x,y],[z])
```

Evaluation

```
x0 = DM([[1,2,3],[4,5,6]])
x1 = DM([[1,3],[0,6],[0,9]])
z0 = f(x0, x1)
print z0
```

```
[[1, 42],
 [4, 96]]
```

Note how this result is related to a numpy approach:

```
a=matrix(x0).reshape(3,2)
b=matrix(x1).reshape(2,3)
print a.T*b.T
```

```
[[ 10.  51.]
 [ 14.  66.]]
```

Numerical matrices

```
X = MX(DM([[1,2,3],[4,5,6]]))
print X
```

```
[[1, 2, 3],
 [4, 5, 6]]
```

```
print mtimes(X,X.T)
```

```
@1=
[[1, 2, 3],
 [4, 5, 6]], (zeros(2x2, dense)+mtimes(@1, @1'))
```

```
print MX(DM([1,2,3]).T)
```

```
[[1, 2, 3]]
```

```
print MX([1,2,3])
```

```
[1, 2, 3]
```

As before, evaluation is lazy on the matrix level

```
Y = MX.sym("Y")
f = Function('f', [Y],[X])
X0 = f([2])
print X0
```

```
[[1, 2, 3],
 [4, 5, 6]]
```

Element assignement

```
63 X = MX.sym("x",2,2)
64 X[0,0]=MX(5)
65 print X
```

```
x[0] = 5)
```