assertion

September 30, 2024

[1]:
```
from casadi import *
```

CasADi provides a mechanism to add assertions in an MX expression graph This can be useful to debug yor code, e.g. debugging why the end-result of a computation yields NaN

Consider this example:

[2]:
```
x = MX.sym("x")
y = sin(x)
z = sqrt(y)
```

[3]:
```
f = Function("f", [x], [z])
```

[4]:
```
z0 = f(5)
```

[5]:
```
print(z0)
```

```
-nan
```

For some mysterious reason we get NaN here

Next, we add an assertion:

[6]:
```
y = y.attachAssert(y>0, "bummer") # Add assertion here
z = sqrt(y)
```

[7]:
```
f = Function("f", [x],[z])
```

[8]:
```
try:
  z0 = f(5)
except Exception as e:
  print("An exception was raised here:")
  print(e)
```

```
An exception was raised here:
Error in Function::call for 'f' [MXFunction] at
…/casadi/core/function.cpp:361:
…/casadi/core/assertion.cpp:70: Assertion error: bummer
```

You can combine this with Callback to do powerful assertions

[9]:
```
class Dummy(Callback):
  def __init__(self, name, opts={}):
    Callback.__init__(self)
    self.construct(name, opts)
  def get_n_in(self): return 1
  def get_n_out(self): return 1
  def eval(self, arg):
    import numpy
    x = arg[0]
    m = max(numpy.real(numpy.linalg.eig(blockcat([[x,-1],[-1,2]]))[0]))
    print("m=",m)
    return [int(m>2)]
```

[10]:
```
foo = Dummy("foo")
```

[11]:
```
y = sin(x)
```

[12]:
```
y = y.attachAssert(foo(y), "you are in trouble") # Add assertion here
z = sqrt(y)
```

[13]:
```
f = Function("f", [x],[z])
```

[14]:
```
z0 = f(5)
```

```
m= 2.3062613059254473
```