yacoda
Let's do some coding...

2. CasADi basics: Expressions graphs and Functions

Expression graphs

Matlab script

```
import casadi.*

x = MX.sym('x');
y = MX.sym('y');

w = x+y;
z = w*y;
```

create symbols

perform operations

Python script

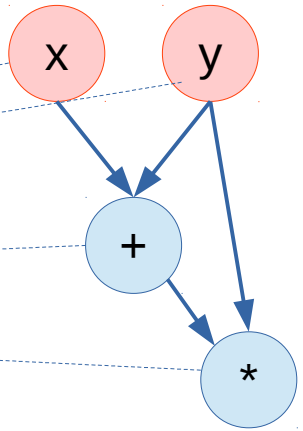
```
from casadi import *

x = MX.sym('x')
y = MX.sym('y')

w = x+y
z = w*y
```

Expression graphs

Matlab script

```
import casadi.*

x = MX.sym('x');
y = MX.sym('y');

w = x+y;
z = w*y;
```

Graph

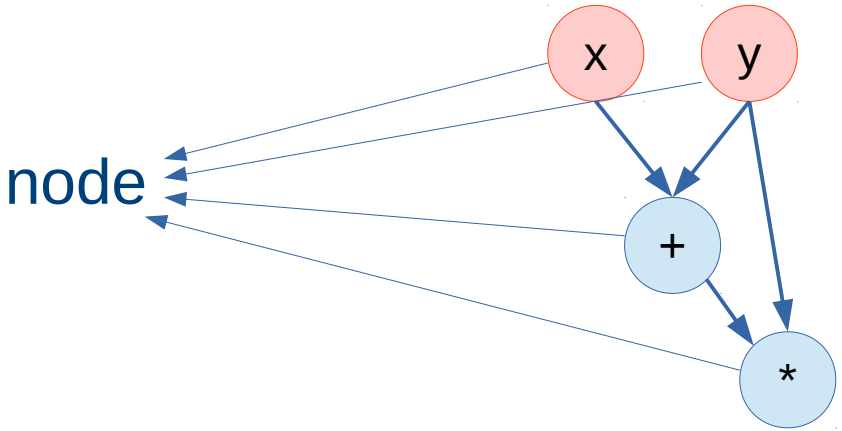


Python script

```
from casadi import *

x = MX.sym('x')
y = MX.sym('y')

w = x+y
z = w*y
```

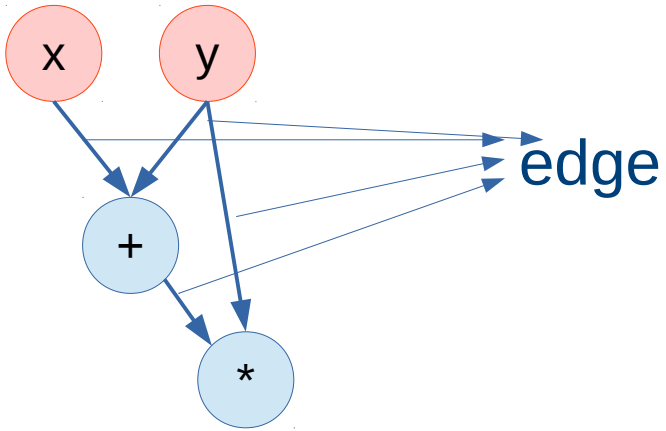
Expression graphs

Graph



Expression graphs

Graph



directed acyclic graph

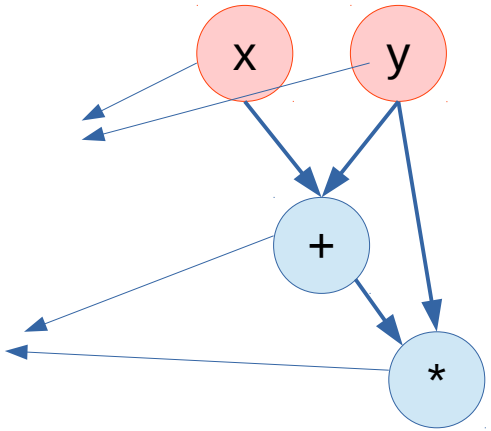
Expression graphs

Graph

types of nodes:

symbolic primitive

operation

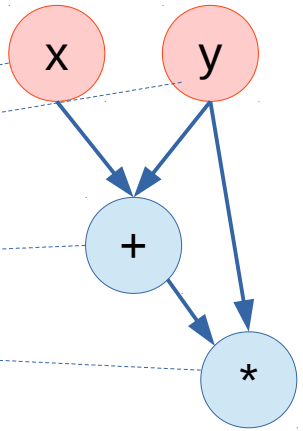


Expression graphs

Matlab script

```
import casadi.*

x = MX.sym('x');
y = MX.sym('y');

w = x+y;
z = w*y;


class(x) % casadi.MX
class(w) % casadi.MX
```

Graph



Python script

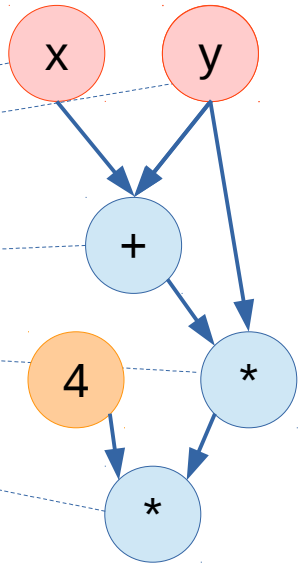
```
from casadi import *

x = MX.sym('x')
y = MX.sym('y')

w = x+y
z = w*y


type(x) # casadi.MX
type(w) # casadi.MX
```

Expression graphs

Matlab script

```
import casadi.*

x = MX.sym('x');
y = MX.sym('y');

w = x+y;
z = w*y;
z = 4*z;


% equivalent to
z = MX(4)*z;
```

Graph



Python script

```
from casadi import *

x = MX.sym('x')
y = MX.sym('y')

w = x+y
z = w*y
z = 4*z


# equivalent to
z = MX(4)*z
```

Note: not an implicit equation!
We simply overide a workspace variable.

Expression graphs

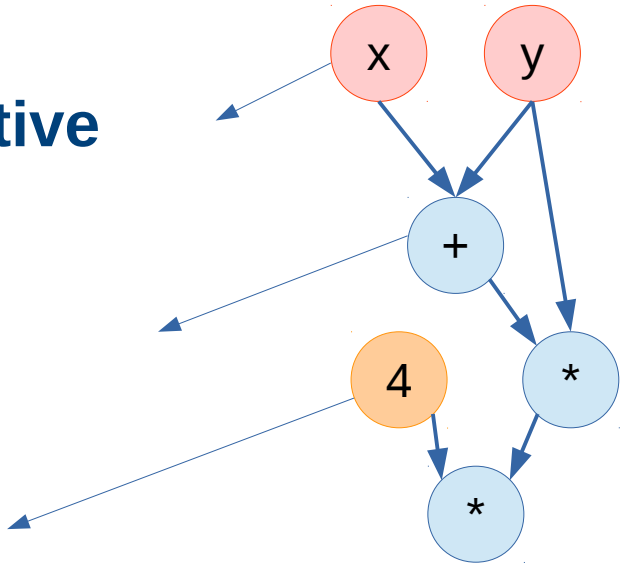
Graph

types of nodes:

symbolic primitive

operation

constant

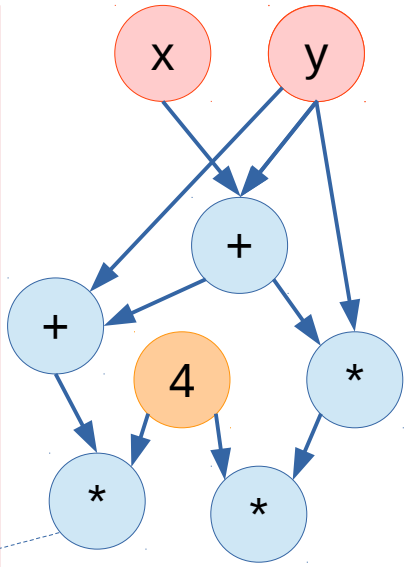


Expression graphs

Matlab script

```
import casadi.*

x = MX.sym('x');
y = MX.sym('y');

w = x+y;
z = w*y;
z = 4*z;



J = jacobian(z,y)
```

Graph



Python script

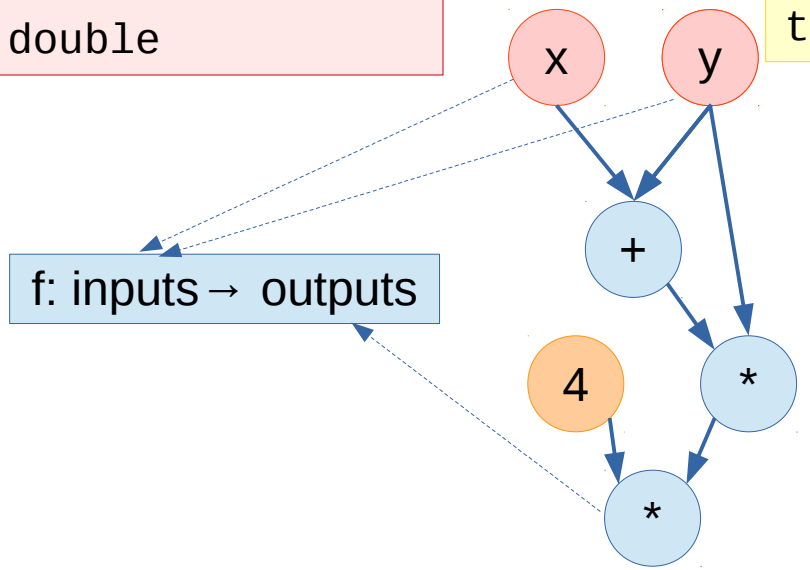
```
from casadi import *

x = MX.sym('x')
y = MX.sym('y')

w = x+y
z = w*y
z = 4*z



J = jacobian(z,y)
```

CasADi Functions

Matlab script

```
f = Function('f',{x,y},{z});


r = f(1,2) % 24

class(r) % casadi.DM

r2 = full(r)
class(r2) % double
```

Graph



Python script

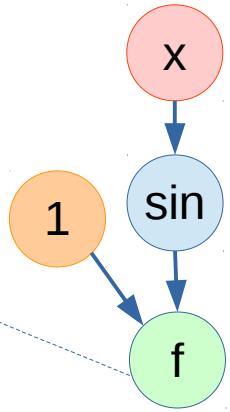
```
f = Function('f',[x,y],[z]);


r = f(1,2) # 24

type(r) # casadi.DM

r2 = np.array(r)
type(r2) # numpy.ndarray
```

CasADi Functions

Matlab script

```
f = Function('f',{x,y},{z});


r = f(1,sin(x))

class(r) % casadi.MX

gradient(r,x)
Function('g',{x,y},{r})
```

Graph



Python script

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


r = f(1,sin(x))

type(r) # casadi.MX

gradient(r,x)
Function('g',{x,y},{r})
```

CasADi Functions

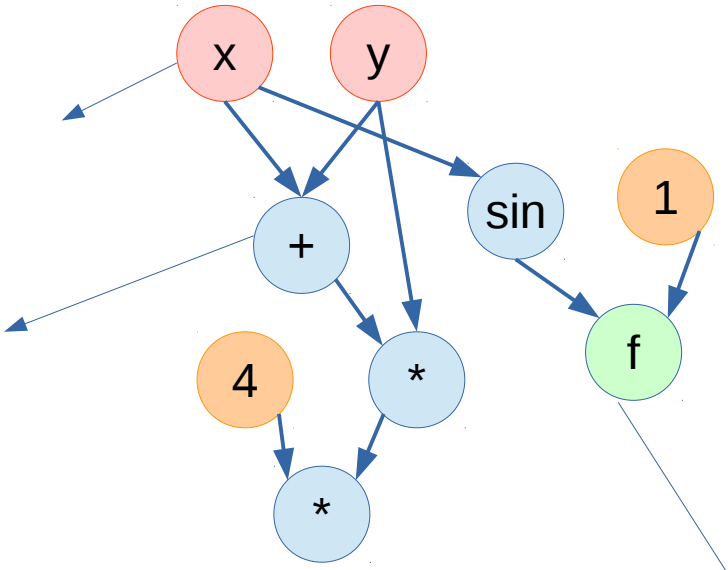
Graph

types of nodes:

symbolic primitive

operation

constant



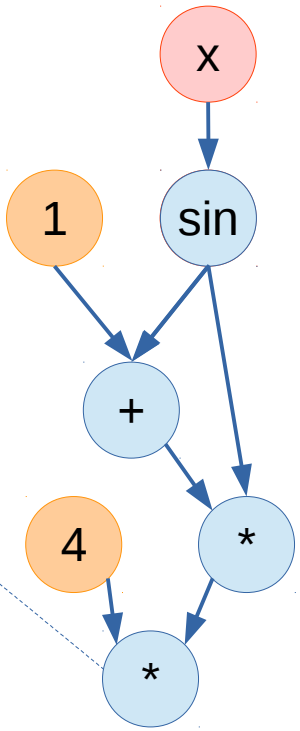
Function call

Function != function

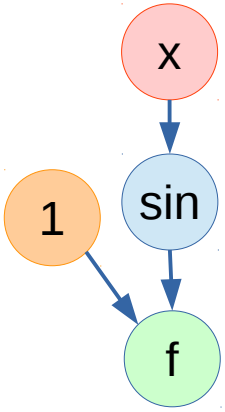
Matlab script

```
function [z] = f(x,y)
  w = x+y;
  z = w*y;
  z = 4*z;
end


r = f(1,sin(x))

class(r) % casadi.MX
```

Graph



!=



Python script

```
def f(x,y):
  w = x+y
  z = w*y
  z = 4*z
  return z


r = f(1,sin(x))

type(r) # casadi.MX
```


ya coda

Common datatypes

Matlab

CasADi

MX DM Function

Python

```
% Cell
a = {7 'foo' 3};
a{1} % 7

% Structure
a = struct('x',7,'y','r');
a.x % 7

% Matrix
a = [1 2 3]
a(1) % 1
```

```
import numpy as np

# List
a = [7, "foo", 3]
a[0] # 7

# Dictionary
a = {"x": 7, "y": "r"}
a["x"] # 7

# Matrix
a = np.array([1,2,3])
a[0] # 1
```

2. CasADi basics

```
M=MX(zeros(2,1));
M(1)=x;
M(2)=y;
```

```
M=[x;y];
```