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

IDAS integrator

We solve a system $dot{x}(t)=f(x(t),y(t),t)$ n $0=g(x(t),y(t),t)$ n

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

```
We solve the following simple dae system that describes
the dynamics of a pendulum:
x' = u, y' = v, u' = lambda * x, v' =lambda * y - g
  s.t. x^2+y^2 = L

We retain g and L as parameters
http://en.wikipedia.org/wiki/Differential_algebraic_equation#Examples
```

```
L = SX.sym("L")
g = SX.sym("g")
```

differential states

```
x=SX.sym("x")
y=SX.sym("y")
u=SX.sym("u")
v=SX.sym("v")
```

algebraic states

```
lambd=SX.sym("lambda")
```

All states and parameters

```
x_all = vertcat(x,u,y,v)
z_all = lambd
p_all = vertcat(L,g)
```

the initial state of the pendulum

```
P_ = [5,10] # parameters

X_ = [3,-1.0/3,4,1.0/4] # differential states

XDOT_ = [-1.0/3,1147.0/240,1.0/4,-653.0/180] # state derivatives

Z_ = [1147.0/720] # algebraic state
```

We construct the DAE system

```
ode = vertcat(u,lambd*x,v,lambd*y+g)
alg = x**2+y**2-L**2
dae = {'x':x_all, 'z':z_all, 'p':p_all, 'ode':ode, 'alg':alg}
```

```
f = Function('f', [x_all, z_all, p_all], [ode, alg], ['x', 'z', 'p'], ['ode',
     'alg'])
```

Let's check we have consistent initial conditions:

```
res = f(p=P_, x=X_, z=Z_)
print(res['ode']) # This should be same as XDOT_
```

```
[-0.333333, 4.77917, 0.25, 16.3722]
```

```
print(res['alg']) # This should be all zeros
```

```
0
```

Let's check our jacobian $frac{dg}{dy}$:

```
j = jacobian(alg,lambd)
print(j)
```

```
00
```

Note that the jacobian is not invertible: it is not of DAE-index 1

This system is not solvable with idas, because it is of DAE-index 3. It is impossible to lambda from the last element of the residual.

We create a DAE system solver

```
I = integrator('I', 'idas', dae, {'calc_ic':False, 'init_xdot':XDOT_})
```

This system is not solvable with idas, because it is of DAE-index 3. It is impossible obtain lambda from the last element of the residual.

```

try:
  I(p=P_, x0=X_, z0=Z_)
except Exception as e:
  print(e)
```

```
.../casadi/interfaces/sundials/idas_interface.cpp:560: IDASolve returned "
    IDA_CONV_FAIL". Consult IDAS documentation.
```

Error:

```
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
CasADi - 2018-02-11 02:38:00 WARNING("I:daeF failed: NaN detected for output ode, at
At t = 0 and h = 2.06366e-19, the corrector convergence failed repeatedly or with |h
```

We construct a reworked version od the DAE (index reduced), now it is DAE-index 1

```
ode = vertcat(u,lambd*x)
alg = vertcat(x**2+y**2-L**2, u*x+v*y,u**2-g*y+v**2+L**2*lambd)
x_all = vertcat(x,u)
z_all = vertcat(y,v,lambd)
dae = {'x':x_all, 'z':z_all, 'p':p_all, 'ode':ode, 'alg':alg}
```

```
f = Function('f', [x_all, z_all, p_all], [ode, alg], ['x', 'z', 'p'], ['ode', 'alg'])
```

the initial state of the pendulum

```
P_ = [5,10] # parameters

X_ = [3,-1.0/3] # differential states

XDOT_ = [-1.0/3,1147.0/240] # state derivatives

Z_ = [4,1.0/4,1147.0/720] # algebraic state
```

Let’s check we have consistent initial conditions:

```
res = f(p=P_, x=X_, z=Z_)
print(res['ode']) # This should be the same as XDOT_
```

```
[-0.333333, 4.77917]
```

```
print(res['alg']) # This should be all zeros
```

```
[0, 0, 0]
```

Let’s check our jacobian:

```
J = f.factory('J', f.name_in(), ['jac:alg:z'])
res = J(p=P_, x=X_, z=Z_)
print(array(res["jac_alg_z"]))
```

```
[[  8.      0.      0.  ]
 [  0.25    4.      0.  ]
 [-10.      0.5    25.  ]]
```

$frac{dg}{dy}$ is invertible this time.

We create a DAE system solver

```
I = integrator('I', 'idas', dae, {'t0':0, 'tf':1, 'init_xdot':XDOT_})
res = I(p=P_, x0=X_, z0=Z_)
print(res['xf'])
```

```
[4.68624, 2.34688]
```

Possible problems

If you would initialize with:

```
P_ = [5,10] # parameters

X_ = [5,0]  # states
```

You will get an error:

```
try:
  I(p=P_, x0=X_, z0=Z_)
except Exception as e:
  print(e)
```

```
.../casadi/interfaces/sundials/idas_interface.cpp:560: IDASolve returned "
    IDA_TOO_MUCH_WORK". Consult IDAS documentation.
```

Error:

```
At t = 6.02906e-09, , mxstep steps taken before reaching tout.
```

Although this initialisation is consistent, it coincides with a singular point.