

# idas

July 23, 2025

This file is part of CasADi.

CasADi -- A symbolic framework for dynamic optimization.  
Copyright (C) 2010–2023 Joel Andersson, Joris Gillis, Moritz Diehl,  
KU Leuven. All rights reserved.  
Copyright (C) 2011–2014 Greg Horn

CasADi is free software; you can redistribute it and/or  
modify it under the terms of the GNU Lesser General Public  
License as published by the Free Software Foundation; either  
version 3 of the License, or (at your option) any later version.

CasADi is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public  
License along with CasADi; if not, write to the Free Software  
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

## 1 IDAS integrator

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

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

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

We retain  $g$  and  $L$  as parameters [http://en.wikipedia.org/wiki/Differential\\_algebraic\\_equation#Examples](http://en.wikipedia.org/wiki/Differential_algebraic_equation#Examples)

```
[2]: L = SX.sym("L")  
     g = SX.sym("g")
```

differential states

```
[3]: x=SX.sym("x")
      y=SX.sym("y")
      u=SX.sym("u")
      v=SX.sym("v")
```

algebraic states

```
[4]: lambd=SX.sym("lambda")
```

All states and parameters

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

the initial state of the pendulum

```
[6]: 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

```
[7]: 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:

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

```
[-0.333333, 4.77917, 0.25, 16.3722]
0
```

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

```
[9]: 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

```
[10]: 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.

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

```
Error in Function::call for 'I' [IdasInterface] at
.../casadi/core/function.cpp:1466:
Error in Function::call for 'I' [IdasInterface] at
.../casadi/core/function.cpp:362:
.../casadi/interfaces/sundials/idas_interface.cpp:599: IDASolve returned
"IDA_CONV_FAIL". Consult IDAS documentation.
```

At  $t = 0$  and  $h = 5.40977e-14$ , the corrector convergence failed repeatedly or with  $|h| = h_{min}$ .

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

```
[12]: ode = vertcat(u, lambda*x)
alg = vertcat(x**2+y**2-L**2, u*x+v*y, u**2-g*y+v**2+L**2*lambda)
x_all = vertcat(x, u)
z_all = vertcat(y, v, lambda)
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

```
[13]: 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:

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

```
[-0.333333, 4.77917]
[0, 0, 0]
```

Let's check our jacobian:

```
[15]: 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

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

```
[4.68624, 2.34688]
```

## 2 Possible problems

If you would initialize with:

```
[17]: P_ = [5,10] # parameters
      X_ = [5,0]  # states
```

You will get an error:

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

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