simulator

September 30, 2024


```
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 Simulator

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

We will investigate the working of Simulator with the help of the parametrically exited Duffing equation:

$\ddot{u} + \dot{u} - \epsilon(2\mu\dot{u} + \alpha u^3 + 2ku\cos(\Omega t))$ with $\Omega = 2 + \epsilon\sigma$.

[2]:
```
t = SX.sym('t')
```

[3]:
```
u = SX.sym('u')
v = SX.sym('v')
states = vertcat(u,v)
```

[4]:
```
eps   = SX.sym('eps')
mu    = SX.sym('mu')
alpha = SX.sym('alpha')
k     = SX.sym('k')
sigma = SX.sym('sigma')
Omega = 2 + eps*sigma
```

[5]:
```
params = vertcat(eps,mu,alpha,k,sigma)
rhs    = vertcat(v,-u-eps*(2*mu*v+alpha*u**3+2*k*u*cos(Omega*t)))
```

We will simulate over 50 seconds, 1000 timesteps.

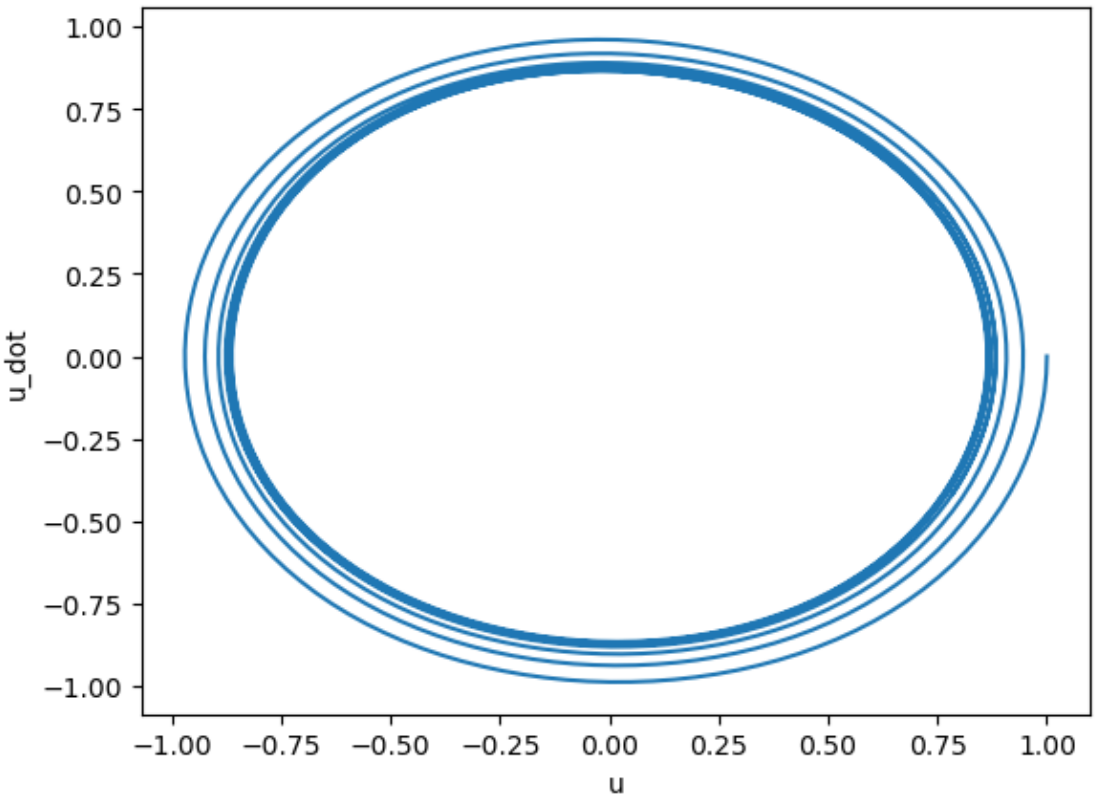
[6]:
```
dae={'x':states, 'p':params, 't':t, 'ode':rhs}
ts = linspace(0, 50, 1000)
integrator = integrator('integrator', 'cvodes', dae, {'grid':ts, 'output_t0':
 ↪True})
```

```
CasADi - 2024-09-30 17:08:11 WARNING("The options 't0', 'tf', 'grid' and
'output_t0' have been deprecated.
The same functionality is provided by providing additional input arguments to
the 'integrator' function, in particular:
 * Call integrator(…, t0, tf, options) for a single output time, or
 * Call integrator(…, t0, grid, options) for multiple grid points.
The legacy 'output_t0' option can be emulated by including or excluding 't0' in
'grid'.
Backwards compatibility is provided in this release only.")
[…/casadi/core/integrator.cpp:515]
```

[7]:
```
sol = integrator(x0=[1,0], p=[0.1,0.1,0.1,0.3,0.1])
```

Plot the solution

[8]:
```
plot(array(sol['xf'])[0,:], array(sol['xf'])[1,:])
xlabel('u')
ylabel('u_dot')
show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
u_dot
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
u