```python
#
#
#
#
#
#
#
import zipfile
import os

fmux = zipfile.ZipFile("CSTR_CSTR_Opt2.fmux",'r')
fmux.extract('modelDescription.xml','.')
from casadi import *
ivp = DaeBuilder()
ivp.parse_fmi("modelDescription.xml")
```

```
Warning: Skipped TiXmlNode::TINYXML_DECLARATION
```

Error:

```
CasADi warning: "opt:PointConstraints not supported, ignored" issued  on line 327 of file "/home/travis/build/casadi/binaries/casadi/casadi/core/misc/dae_builder.cpp".
CasADi warning: "opt:PointConstraints not supported, ignored" issued  on line 32[illegible] [illegible].cpp".
```

Let us have a look at the flat optimal control problem:

```python
print ivp
```

```
DAE(#s = 4, #x = 0, #z = 0, #q = 1, #y = 0, #p = 0, #d = 22, #u = 1)
Variables
{
  t = t
  s = [cstr.T, cstr.Tc, cstr.c, q]
  q =  [Iterm]
  d =  [cstr.F0, cstr.c0, cstr.F, cstr.T0, cstr.r, cstr.k0, cstr.EdivR,
      cstr.U, cstr.rho, cstr.Cp, cstr.dH, cstr.V, cstr.c_init, cstr.T_init
      , c_ref, T_ref, Tc_ref, q_c, q_T, q_Tc, startTime, finalTime]
  u =  [u]
}
Dependent parameters
cstr.F0 == 0.00166667
cstr.c0 == 1000
cstr.F == 0.00166667
cstr.T0 == 350
cstr.r == 0.219
cstr.k0 == 1200000000
cstr.EdivR == 8750
cstr.U == 915.6
cstr.rho == 1000
cstr.Cp == 239
cstr.dH == -50000
cstr.V == 100
cstr.c_init == 1000
cstr.T_init == 350
c_ref == 500
T_ref == 320
Tc_ref == 300
q_c == 1
q_T == 1
q_Tc == 1
startTime == 0
finalTime == 150

Fully-implicit differential-algebraic equations
0 == MX((der_cstr.c-(((cstr.F0*(cstr.c0-cstr.c))/cstr.V)-((cstr.k0*cstr.c)
    *exp((-(cstr.EdivR/cstr.T)))))))
0 == MX((der_cstr.T-((((cstr.F0*(cstr.T0-cstr.T))/cstr.V)-((((cstr.dH/(
    cstr.rho*cstr.Cp))*cstr.k0)*cstr.c)*exp((-(cstr.EdivR/cstr.T)))))
    +(((2.*cstr.U)/((cstr.r*cstr.rho)*cstr.Cp))*(cstr.Tc-cstr.T)))))
0 == MX((u-cstr.Tc))
0 == MX((q-(2.*u)))

Quadrature equations
der_Iterm == (0.0001*(((q_c*sq((c_ref-cstr.c)))+(q_T*sq((T_ref-cstr.T))))
    +(q_Tc*sq((Tc_ref-cstr.Tc)))))

Initial equations
0 == (cstr.c-cstr.c_init)
0 == (cstr.T-cstr.T_init)
```

```python
ivp.make_explicit()
```

Let us extract variables for the states, the control and equations

```python
x = vertcat(*ivp.x)
u = vertcat(*ivp.u)
f = vertcat(*ivp.ode)
L = vertcat(*ivp.quad)
I = vertcat(*ivp.init)
print 5*sin(f[0])
```

```
(5*sin((zeros(2x2, diagonal)[:2] += ones(2x1, dense))'[0]
    \(((0.00166667*(1000-cstr.c))/100)-((1200000000*cstr.c)*exp((-(8750/
    cstr.T))))))))
```

```python
print jacobian(f,x)
```

```
@1=8750, @2=(@1/cstr.T), @3=exp((-@2)), @4=1200000000, @5=vertsplit(ones(2
    x1, 1 nnz)){0}, @6=0.00166667, @7=vertsplit(ones(2x1, 1 nnz)){1}, @8=
    zeros(2x2, diagonal)[:2] += ones(2x1, dense))', @9=horzsplit((@8[0]\
    horzcat(((-(@3*(@4*@5)))-(0.01*(@6*@5))), (-((@4*cstr.c)*(@3*((@2/cstr
    .T)*@7))))))), @10=(@1/cstr.T), @11=exp((-@10)), @12=-2.51046e+08, @13
    =horzsplit((@8[1]\horzcat((-(@11*(@12*@5))), (((-((@12*cstr.c)*(@11*((
    @10/cstr.T)*@7))))-(0.01*(@6*@7)))-(0.034986*@7)))))), zeros(2x2, dense
    )[:4:2] = vertcat(@9{0}, @13{0}))[1:5:2] = vertcat(@9{1}, @13{1}))'
```

```python
print hessian(L,x)
```

```
(MX(@1=0.0001, @2=vertsplit(ones(2x1, dense)), @3=vertcat((-(@1*(2.*(-@2
    {0})))), (-(@1*(2.*(-@2{1}))))), zeros(2x2, diagonal)[:2] = @3)[:2] =
    @3)'), MX(@1=0.0001, vertcat((-(@1*(2.*(500-cstr.c)))), (-(@1
    *(2.*(320-cstr.T)))))))
```

```python
ubx = ivp.max(x)
lbx = ivp.min(x)
ubu = ivp.max(u)
lbu = ivp.min(u)
x0 = ivp.guess(x)
```

```
u0 = ivp.guess(u)
tf = 150.
print "f = ", f, "\nI = ", I, "\nL = ", L
```

```
f = @1=0.00166667, @2=100, @3=8750, @4=zeros(2x2, diagonal)[:2] += ones(2
    x1, dense))', vertcat((@4[0]\(((@1*(1000-cstr.c))/@2)-((1200000000*
    cstr.c)*exp((-(@3/cstr.T)))))), (@4[1]\((((@1*(350-cstr.T))/@2)
    -((-2.51046e+08*cstr.c)*exp((-(@3/cstr.T)))))+(0.034986*((-1\(-u))-
    cstr.T)))))
I = vertcat((cstr.c-1000), (cstr.T-350))
L = (0.0001*((sq((500-cstr.c))+sq((320-cstr.T)))+sq((300-(-1\(-u))))))
```

```
print "lbx = ", lbx, "ubx = ", ubx, "lbu = ", lbu, "ubu = ", ubu
```

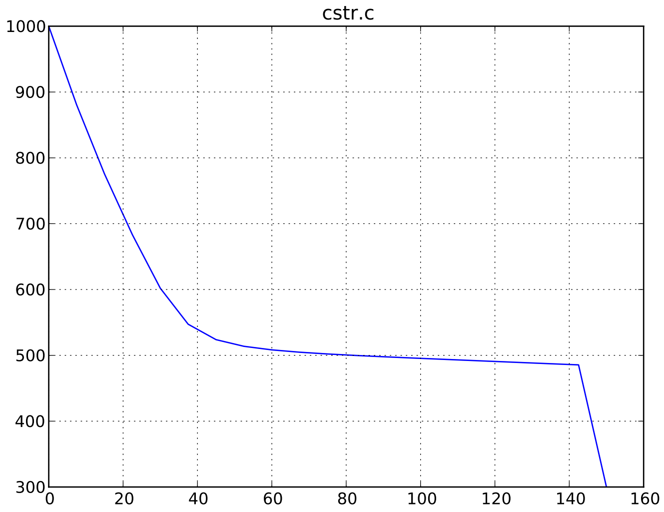
```
lbx = [-inf, -inf] ubx = [inf, 350.0] lbu = [230.0] ubu = [370.0]
```

```
ode_fcn = Function('ode_fcn', [x,u],[f,L])
init_fcn = Function('init_fcn', [x],[I])
ode_fcn = ode_fcn.expand()
init_fcn = init_fcn.expand()
nk = 20
dt = tf/nk
nx = x.size1()
nu = u.size1()
nj = 10; h = dt/nj
xk = MX.sym("xk",nx)
uk = MX.sym("uk",nu)
xkj = xk; xkj_L = 0
for j in range(nj):
   k1,k1_L = ode_fcn(xkj,uk)
   k2,k2_L = ode_fcn(xkj + h/2*k1,uk)
   k3,k3_L = ode_fcn(xkj + h/2*k2,uk)
   k4,k4_L = ode_fcn(xkj + h*k3,uk)
   xkj   += h/6 * (k1   + 2*k2   + 2*k3   + k4)
   xkj_L += h/6 * (k1_L + 2*k2_L + 2*k3_L + k4_L)
integrator = Function('integrator', [xk,uk],[xkj,xkj_L])
xk = [MX.sym("x" + str(k), nx) for k in range(nk+1)]
uk = [MX.sym("u" + str(k), nu) for k in range(nk)]
v = []; lbv = []; ubv = []; v0 = []
nv = 0
vind = {'x':[], 'u':[]}
def valloc(n, nv): return range(nv, nv+n), nv+n
for k in range(nk):
   #$ States
   ind, nv = valloc(nx, nv)
   v.append(xk[k]); lbv.append(lbx); ubv.append(ubx); v0.append(x0); vind['x'
       ].append(ind)
   #$ Control
   ind, nv = valloc(nu, nv)
   v.append(uk[k]); lbv.append(lbu); ubv.append(ubu); v0.append(u0); vind['u'
       ].append(ind)
ind, nv = valloc(nx, nv)
v.append(xk[-1]); lbv.append(lbx); ubv.append(ubx); v0.append(x0); vind['x'].
   append(ind)
v = vertcat(*v); lbv = vertcat(*lbv); ubv = vertcat(*ubv); v0 = vertcat(*v0)
J = 0;  eq = []
eq0 = init_fcn(xk[0])
eq.append(eq0)
for k in range(nk):
    xk_end,Jk = integrator(xk[k],uk[k])
    J += Jk
    if k+1<nk: eq.append(xk_end - xk[k+1])
nlp = {'x':v, 'f':J, 'g':vertcat(*eq)}
solver = nlpsol("solver", "ipopt", nlp)
sol = solver(lbx=lbv, ubx=ubv, x0=v0, lbg=0, ubg=0)
```

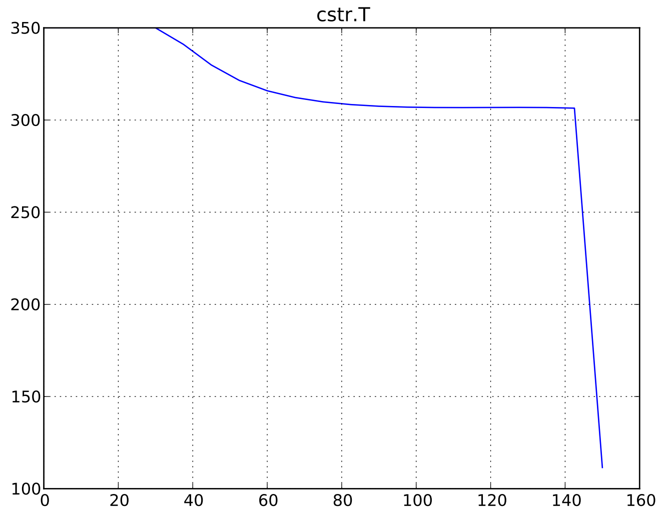
```
******************************************************************************

This program contains Ipopt, a library for large-scale nonlinear
    optimization.
 Ipopt is released as open source code under the Eclipse Public License (
    EPL).
        For more information visit http://projects.coin-or.org/Ipopt
******************************************************************************


This is Ipopt version 3.12.3, running with linear solver ma57.

Number of nonzeros in equality constraint Jacobian...:      154
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:      120

Total number of variables............................:       62
                     variables with only lower bounds:        0
                variables with lower and upper bounds:       20
                     variables with only upper bounds:       21
Total number of equality constraints.................:       40
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
    alpha_pr  ls
   0  6.4299292e+02 7.00e+02 1.00e+00  -1.0 0.00e+00    -  0.00e+00 0.00e
     +00    0
   1  5.9103062e+02 7.00e+02 1.47e+00  -1.0 7.00e+02  -4.0 3.61e-21 3.39e
     -21F  1
   2  5.2897122e+02 6.03e+02 8.24e+00  -1.0 7.00e+02  -4.5 7.81e-01 1.39e
     -01f  1
   3  3.2549076e+02 4.41e+02 2.97e+01  -1.0 6.03e+02  -5.0 2.50e-01 2.68e
     -01f  1
   4  1.7251112e+02 4.30e+02 2.27e+01  -1.0 5.58e+03  -5.4 5.17e-03 2.55e
     -02f  1
   5  1.2226738e+02 4.22e+02 3.28e+01  -1.0 4.35e+03  -5.9 1.69e-03 1.82e
     -02f  1
   6  1.2206550e+02 4.22e+02 2.70e+01  -1.0 1.30e+03  -6.4 3.04e-03 1.32e
     -03h  1
   7  1.2432796e+02 4.04e+02 2.90e+01  -1.0 5.57e+02  -6.9 1.97e-03 4.18e
     -02h  1
   8  2.1892488e+02 3.23e+02 1.87e+02  -1.0 4.36e+02  -7.3 1.61e-02 1.99e
     -01h  1
```

```
   9  2.2145519e+02 3.21e+02 1.82e+02  -1.0 3.23e+02  -7.8 1.78e-02 8.40e
      -03h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
    alpha_pr  ls
  10  4.9597972e+02 1.73e+02 8.76e+01  -1.0 4.02e+02  -8.3 1.21e-03 4.60e
      -01h  1
  11  5.1249276e+02 1.65e+02 8.37e+01  -1.0 1.73e+02  -8.8 2.87e-01 4.52e
      -02h  1
  12  8.0199667e+02 4.23e+01 4.60e+01  -1.0 1.65e+02  -9.2 2.36e-01 7.44e
      -01h  1
  13  6.2728789e+02 6.10e+01 1.75e+01  -1.0 9.00e+01  -9.7 5.98e-02 9.85e
      -01f  1
  14  5.5225226e+02 2.91e+01 7.29e+00  -1.0 6.61e+01  -1.9 9.20e-02 9.90e
      -01f  1
  15  4.5828993e+02 1.05e+01 5.44e+01  -1.0 6.82e+01  -2.4 4.24e-01 6.83e
      -01f  1
  16  4.1540595e+02 9.90e+00 2.40e+01  -1.0 1.10e+02  -2.8 2.73e-01 2.47e
      -01f  1
  17  4.0202390e+02 9.44e+00 1.07e+02  -1.0 1.31e+02  -3.3 2.22e-01 7.19e
      -02f  1
  18  3.6766946e+02 4.82e+00 3.14e+02  -1.0 5.72e+01  -3.8 9.12e-02 4.90e
      -01f  1
  19  3.1887985e+02 2.12e+01 6.62e+02  -1.0 1.82e+02  -4.3 6.66e-02 3.23e
      -01f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
    alpha_pr  ls
  20  3.0753849e+02 1.32e+01 9.19e+01  -1.0 5.18e+01  -4.7 5.84e-01 3.85e
      -01f  1
  21  2.9636430e+02 2.76e+00 7.30e+02  -1.0 3.05e+01  -5.2 8.00e-01 1.00e
      +00f  1
  22  2.9687769e+02 3.30e-01 2.10e-03  -1.0 5.15e+01  -5.7 1.00e+00 1.00e
      +00h  1
  23  2.9673719e+02 1.28e-02 2.42e-04  -1.7 1.08e+02  -6.2 1.00e+00 1.00e
      +00h  1
  24  2.9665238e+02 7.54e-04 3.13e+00  -3.8 1.22e+00  -6.7 9.99e-01 1.00e
      +00h  1
  25  2.9665135e+02 6.71e-07 6.43e-07  -3.8 8.65e+00  -7.1 1.00e+00 1.00e
      +00h  1
  26  2.9665075e+02 2.65e-08 6.98e-09  -5.7 2.81e-01  -7.6 1.00e+00 1.00e
      +00h  1
  27  2.9665075e+02 3.98e-12 1.05e-11  -8.6 1.27e-03  -8.1 1.00e+00 1.00e
      +00h  1

Number of Iterations....: 27

                                   (scaled)                 (unscaled)
Objective...............:   2.9665074612381767e+02    2.9665074612381767e
    +02
Dual infeasibility......:   1.0469387648521393e-11    1.0469387648521393e
    -11
Constraint violation....:   3.9790393202565610e-12    3.9790393202565610e
    -12
Complementarity.........:   2.5116299426194594e-09    2.5116299426194594e
    -09
Overall NLP error.......:   2.5116299426194594e-09    2.5116299426194594e
    -09


Number of objective function evaluations             = 28
Number of objective gradient evaluations             = 28
Number of equality constraint evaluations            = 29
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 28
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 27
Total CPU secs in IPOPT (w/o function evaluations)   =      0.028
Total CPU secs in NLP function evaluations           =      1.138

EXIT: Optimal Solution Found.
                   proc           wall      num          mean
                                  mean
                   time           time      evals        proc time
                       wall time
          nlp_f    0.070 [s]      0.065 [s]    28         2.50 [ms]
              2.32 [ms]
          nlp_g    0.080 [s]      0.068 [s]    29         2.76 [ms]
              2.34 [ms]
      nlp_grad_f   0.180 [s]      0.211 [s]    29         6.21 [ms]
          7.26 [ms]
       nlp_jac_g   0.280 [s]      0.289 [s]    29         9.66 [ms]
           9.97 [ms]
      nlp_hess_l   0.530 [s]      0.525 [s]    27        19.63 [ms]
          19.46 [ms]
    all previous   1.140 [s]      1.158 [s]
   callback_prep   0.000 [s]      0.000 [s]    28         0.00 [ms]
        0.01 [ms]
          solver   0.020 [s]      0.013 [s]
        mainloop   1.160 [s]      1.171 [s]
```

```python
v_opt  = sol["x"]
x0_opt = v_opt[[vind['x'][k][0] for k in range(nk+1)]]
x1_opt = v_opt[[vind['x'][k][1] for k in range(nk+1)]]
u_opt = v_opt[[vind['u'][k][0] for k in range(nk)]]
from pylab import *
from numpy import *
tgrid = linspace(0,tf,nk+1)
figure(1)
plot(tgrid,x0_opt)
title(str(x[0]))
grid()
show()
```




```
figure(2)
plot(tgrid,x1_opt)
title(str(x[1]))
grid()
show()
```




```
figure(3)
step(tgrid,vertcat(u_opt[0],u_opt))
title(str(u))
grid()
show()
```