callback

September 30, 2024

1 Callback

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

In this example, we will demonstrate callback functionality for Ipopt. Note that you need the fix https://github.com/casadi/casadi/wiki/enableIpoptCallback before this works

We start with constructing the rosenbrock problem

[2]:
```
x=SX.sym("x")
y=SX.sym("y")
```

[3]:
```
f = (1-x)**2+100*(y-x**2)**2
nlp={'x':vertcat(x,y), 'f':f,'g':x+y}
fcn = Function('f', [x, y], [f])
```

[4]:
```
import matplotlib
if "Agg" not in matplotlib.get_backend():
  matplotlib.interactive(True)
```

[5]:
```
from pylab import figure, subplot, contourf, colorbar, draw, show, plot, title
```

[6]:
```
import time
```

[7]:
```
class MyCallback(Callback):
  def __init__(self, name, nx, ng, np, opts={}):
    Callback.__init__(self)

    self.nx = nx
    self.ng = ng
    self.np = np

    figure(1)

    x_,y_ = mgrid[-1:1.5:0.01,-1:1.5:0.01]
    z_ = DM.zeros(x_.shape)

    for i in range(x_.shape[0]):
      for j in range(x_.shape[1]):
        z_[i,j] = fcn(x_[i,j],y_[i,j])
    contourf(x_,y_,z_)
    colorbar()
    title('Iterations of Rosenbrock')
    draw()

    self.x_sols = []
    self.y_sols = []

    # Initialize internal objects
    self.construct(name, opts)

  def get_n_in(self): return nlpsol_n_out()
  def get_n_out(self): return 1
  def get_name_in(self, i): return nlpsol_out(i)
  def get_name_out(self, i): return "ret"

  def get_sparsity_in(self, i):
    n = nlpsol_out(i)
    if n=='f':
      return Sparsity. scalar()
    elif n in ('x', 'lam_x'):
      return Sparsity.dense(self.nx)
    elif n in ('g', 'lam_g'):
```

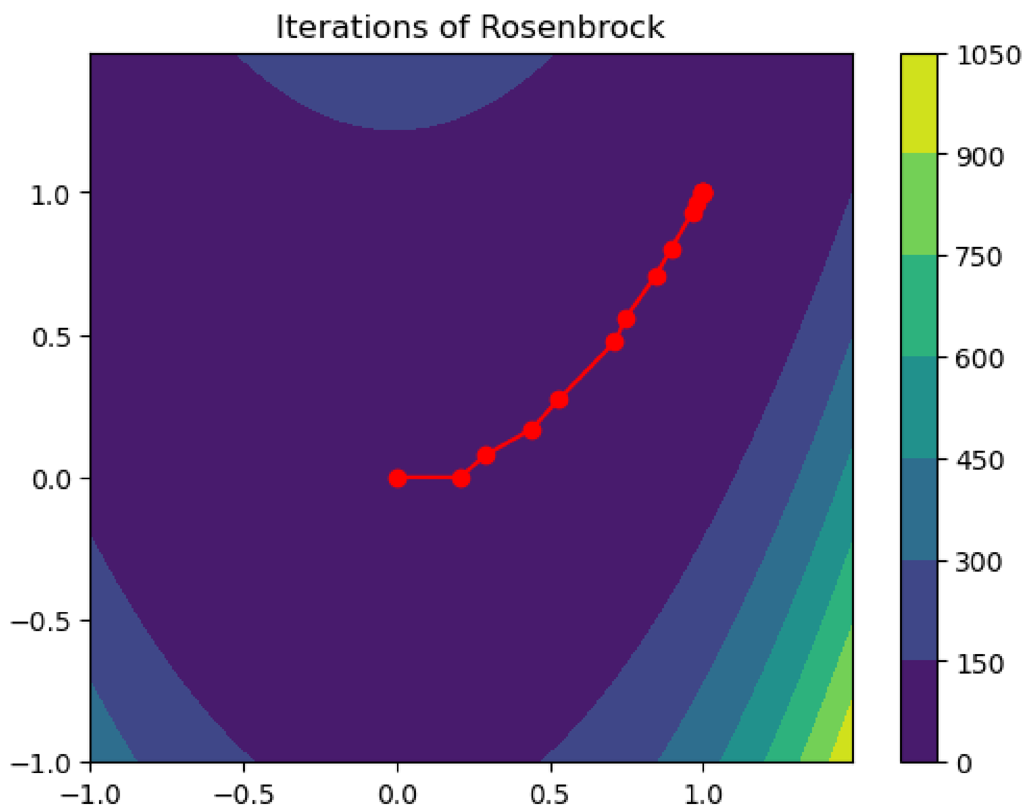
```
      return Sparsity.dense(self.ng)
    else:
      return Sparsity(0,0)
  def eval(self, arg):
    # Create dictionary
    darg = {}
    for (i,s) in enumerate(nlpsol_out()): darg[s] = arg[i]

    sol = darg['x']
    self.x_sols.append(float(sol[0]))
    self.y_sols.append(float(sol[1]))

    if hasattr(self,'lines'):
      if "template" not in matplotlib.get_backend(): # Broken for template:␣
↪https://github.com/matplotlib/matplotlib/issues/8516/
        self.lines[0].set_data(self.x_sols,self.y_sols)

    else:
      self.lines = plot(self.x_sols,self.y_sols,'or-')

    draw()
    time.sleep(0.25)

    return [0]
```

[8]:
```
mycallback = MyCallback('mycallback', 2, 1, 0)
opts = {}
opts['iteration_callback'] = mycallback
opts['ipopt.tol'] = 1e-8
opts['ipopt.max_iter'] = 50
solver = nlpsol('solver', 'ipopt', nlp, opts)
sol = solver(lbx=-10, ubx=10, lbg=-10, ubg=10)
```

```
******************************************************************************
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 https://github.com/coin-or/Ipopt
******************************************************************************

This is Ipopt version 3.14.11, running with linear solver MUMPS 5.4.1.

Number of nonzeros in equality constraint Jacobian…:        0
Number of nonzeros in inequality constraint Jacobian.:        2
Number of nonzeros in Lagrangian Hessian…:        3

Total number of variables…:        2
```

```
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        2
                     variables with only upper bounds:        0
Total number of equality constraints…:        0
Total number of inequality constraints…:        1
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        1
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  1.0000000e+00 0.00e+00 1.33e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  8.1696108e-01 0.00e+00 8.18e+00  -1.0 8.33e-01    -  9.22e-01 2.50e-01f  3
   2  5.0928866e-01 0.00e+00 1.31e+00  -1.0 1.58e-01    -  1.00e+00 1.00e+00f  1
   3  3.8213489e-01 0.00e+00 5.32e+00  -1.0 4.89e-01    -  1.00e+00 5.00e-01f  2
   4  2.2689357e-01 0.00e+00 1.54e+00  -1.0 1.92e-01    -  1.00e+00 1.00e+00f  1
   5  1.9526345e-01 0.00e+00 9.02e+00  -1.0 3.85e-01    -  1.00e+00 1.00e+00f  1
   6  6.2791707e-02 0.00e+00 2.66e-01  -1.0 1.22e-01    -  1.00e+00 1.00e+00f  1
   7  3.3688142e-02 0.00e+00 3.14e+00  -1.7 4.88e-01    -  1.00e+00 5.00e-01f  2
   8  1.1215647e-02 0.00e+00 6.93e-01  -1.7 1.45e-01    -  1.00e+00 1.00e+00f  1
   9  3.3356343e-03 0.00e+00 1.69e+00  -1.7 1.91e-01    -  1.00e+00 1.00e+00f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10  3.2713823e-04 0.00e+00 8.94e-02  -1.7 5.61e-02    -  1.00e+00 1.00e+00f  1
  11  9.2093197e-06 0.00e+00 1.06e-01  -2.5 4.91e-02    -  1.00e+00 1.00e+00f  1
  12  1.0134200e-07 0.00e+00 3.06e-04  -2.5 3.33e-03    -  1.00e+00 1.00e+00f  1
  13  2.0350914e-10 0.00e+00 3.66e-05  -3.8 9.12e-04    -  1.00e+00 1.00e+00f  1
  14  2.9616464e-14 0.00e+00 7.79e-08  -5.7 4.22e-05    -  1.00e+00 1.00e+00h  1
  15  5.4181655e-20 0.00e+00 1.17e-11  -8.6 5.16e-07    -  1.00e+00 1.00e+00f  1

Number of Iterations…: 15

                                   (scaled)                 (unscaled)
Objective…:   5.4181654535916011e-20    5.4181654535916011e-20
Dual infeasibility…:   1.1724482496621431e-11    1.1724482496621431e-11
Constraint violation…:   0.0000000000000000e+00    0.0000000000000000e+00
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity…:   2.5060224083090714e-09    2.5060224083090714e-09
Overall NLP error…:   2.5060224083090714e-09    2.5060224083090714e-09


Number of objective function evaluations             = 22
Number of objective gradient evaluations             = 16
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 22
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 16
Number of Lagrangian Hessian evaluations             = 15
Total seconds in IPOPT                               = 4.355
```

```
EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
callback_fun  | 662.72ms ( 41.42ms)   4.34 s (271.32ms)        16
       nlp_f  |  49.00us (  2.23us)  37.12us (  1.69us)        22
       nlp_g  |  80.00us (  3.64us)  62.82us (  2.86us)        22
  nlp_grad_f  |  60.00us (  3.53us)  38.58us (  2.27us)        17
  nlp_hess_l  | 142.00us (  9.47us) 111.10us (  7.41us)        15
   nlp_jac_g  |  42.00us (  2.47us)  29.21us (  1.72us)        17
       total  | 683.23ms (683.23ms)   4.36 s (  4.36 s)         1
```



By setting matplotlib interactivity off, we can inspect the figure at ease

[9]:
```
matplotlib.interactive(False)
show()
```