CasADi tutorial

```
0  #
1  #
2  #
3  #
4  #
5  #
6  #
```

This tutorial file explains the interface with IPOPT Ipopt solves problems of the form:
Minimize f(x) x in R^n s.t g_L <= g(x) <= g_U x_L <= x <= x_U

```
19 from numpy import *
20 import numpy as n
21 from casadi import *
```

Quadractic program

Using Ipopt to do a simple quadratic problem

```
28 P = n.eye(5)
29 A = n.diag([2.0,1,4,10,-2])
30 q = [1,0,0,0,0]
31 b = [1,1,1,1,1]
32
33 X = MX.sym("x",5,1)
34 P = MX(DM(P))
35 q = MX(DM(q))
36 A = MX(DM(A))
```

Objective

```
39 F = 0.5*mtimes([X.T,P,X]) + mtimes(q.T,X)
```

Constraint

```
42 G = X+X
```

NLP

```
45 nlp = {'x':X, 'f':F, 'g':G}
46
47 solver = nlpsol("nlp","ipopt", nlp)
48 solver.printOptions()
```

```
"Option name" [type] = value
> "ad_weight"           [OT_DOUBLE]      "Weighting factor for derivative
    calculation.When there is an option of either using forward or reverse
     mode directional derivatives, the condition ad_weight*nf<=(1-
    ad_weight)*na is used where nf and na are estimates of the number of
    forward/reverse mode directional derivatives needed. By default,
    ad_weight is calculated automatically, but this can be overridden by
    setting this option. In particular, 0 means forcing forward mode and 1
     forcing reverse mode. Leave unset for (class specific) heuristics."
> "ad_weight_sp"        [OT_DOUBLE]      "Weighting factor for sparsity
    pattern calculation calculation.Overrides default behavior. Set to 0
    and 1 to force forward and reverse mode respectively. Cf. option "
    ad_weight"."
> "compiler"            [OT_STRING]      "Just-in-time compiler plugin to be
    used."
> "derivative_of"       [OT_FUNCTION]    "The function is a
    derivative of another function. The type of derivative (directional
    derivative, Jacobian) is inferred from the function name."
> "gather_stats"        [OT_BOOL]        "Flag to indicate whether
    statistics must be gathered"
> "input_scheme"        [OT_STRINGVECTOR]    "Custom input scheme"
> "inputs_check"        [OT_BOOL]        "Throw exceptions when the
    numerical values of the inputs don't make sense"
> "jac_penalty"         [OT_DOUBLE]      "When requested for a number of
    forward/reverse directions,  it may be cheaper to compute first the
    full jacobian and then multiply with seeds, rather than obtain the
    requested directions in a straightforward manner. Casadi uses a
    heuristic to decide which is cheaper. A high value of 'jac_penalty'
    makes it less likely for the heurstic to chose the full Jacobian
    strategy. The special value -1 indicates never to use the full
    Jacobian strategy"
> "jit"                 [OT_BOOL]        "Use just-in-time compiler to speed up the
    evaluation"
> "jit_options"         [OT_DICT]        "Options to be passed to the jit
    compiler."
> "monitor"             [OT_STRINGVECTOR]    "Monitors to be activated"
> "output_scheme"       [OT_STRINGVECTOR]    "Custom output scheme"
> "regularity_check"    [OT_BOOL]        "Throw exceptions when NaN or
    Inf appears during evaluation"
> "user_data"           [OT_VOIDPTR]     "A user-defined field that can be
    used to identify the function or pass additional information"
> "verbose"             [OT_BOOL]        "Verbose evaluation -- for debugging"
> "eval_errors_fatal"   [OT_BOOL]        "When errors occur during
    evaluation of f,g,...,stop the iterations"
> "expand"              [OT_BOOL]        "Replace MX with SX expressions in
    problem formulation [false]"
> "ignore_check_vec"    [OT_BOOL]        "If set to true, the input
    shape of F will not be checked."
> "iteration_callback"  [OT_FUNCTION]    "A function that will
    be called at each iteration with the solver as input. Check
    documentation of Callback."
> "iteration_callback_ignore_errors"    [OT_BOOL]    "If set to
    true, errors thrown by iteration_callback will be ignored."
> "iteration_callback_step"    [OT_INT]    "Only call the callback
    function every few iterations."
> "print_time"          [OT_BOOL]        "print information about execution
    time"
> "verbose_init"        [OT_BOOL]        "Print out timing information
    about the different stages of initialization"
> "warn_initial_bounds"    [OT_BOOL]    "Warn if the initial guess
    does not satisfy LBX and UBX"
> "con_integer_md"      [OT_DICT]        "Integer metadata (a dictionary
    with lists of integers) about constraints to be passed to IPOPT"
> "con_numeric_md"      [OT_DICT]        "Numeric metadata (a dictionary
    with lists of reals) about constraints to be passed to IPOPT"
> "con_string_md"       [OT_DICT]        "String metadata (a dictionary
    with lists of strings) about constraints to be passed to IPOPT"
> "grad_f"              [OT_FUNCTION]    "Function for calculating the
    gradient of the objective (column, autogenerated by default)"
```

```
> "grad_f_options"          [OT_DICT]      "Options for the autogenerated
    gradient of the objective."
> "hess_lag"           [OT_FUNCTION]      "Function for calculating the
    Hessian of the Lagrangian (autogenerated by default)"
> "hess_lag_options"           [OT_DICT]      "Options for the
    autogenerated Hessian of the Lagrangian."
> "ipopt"           [OT_DICT]      "Options to be passed to IPOPT"
> "jac_g"           [OT_FUNCTION]      "Function for calculating the
    Jacobian of the constraints (autogenerated by default)"
> "jac_g_options"           [OT_DICT]      "Options for the autogenerated
    Jacobian of the constraints."
> "pass_nonlinear_variables"           [OT_BOOL]      "Pass list of
    variables entering nonlinearly to IPOPT"
> "var_integer_md"           [OT_DICT]      "Integer metadata (a dictionary
     with lists of integers) about variables to be passed to IPOPT"
> "var_numeric_md"           [OT_DICT]      "Numeric metadata (a dictionary
     with lists of reals) about variables to be passed to IPOPT"
> "var_string_md"           [OT_DICT]      "String metadata (a dictionary
    with lists of strings) about variables to be passed to IPOPT"
```

The default lower an upper bound on the optimizations variables is zero. Change them to unbouded as follows:

```
lbx = [-100,-100,-100,-100,-100]
ubx = [ 100, 100, 100, 100, 100]
```

Inequality constraints. The lower bound is also necessary, although Ipopt itself does not seem to require it

```
ubg = b
lbg = [-100,-100,-100,-100,-100]

sol = solver(lbx=lbx, ubx=ubx, lbg=lbg, ubg=ubg)
```

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

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...:        0
Number of nonzeros in inequality constraint Jacobian.:        5
Number of nonzeros in Lagrangian Hessian.............:       15

Total number of variables............................:        5
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        5
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        5
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        5
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du
    alpha_pr  ls
   0  0.0000000e+00 0.00e+00 4.00e-01  -1.0 0.00e+00    -  0.00e+00 0.00e
     +00   0
   1 -2.0566913e-01 0.00e+00 1.06e-01  -1.0 4.74e-01    -  7.21e-01 1.00e
     +00f  1
   2 -4.7229068e-01 0.00e+00 4.83e-03  -1.0 1.72e+00    -  9.85e-01 1.00e
     +00f  1
   3 -4.9584561e-01 0.00e+00 1.60e-17  -1.7 1.56e-01    -  1.00e+00 1.00e
     +00f  1
   4 -4.9997299e-01 0.00e+00 8.07e-17  -3.8 8.21e-02    -  1.00e+00 1.00e
     +00f  1
   5 -4.9999979e-01 0.00e+00 4.74e-17  -3.8 6.70e-03    -  1.00e+00 1.00e
     +00f  1
   6 -5.0000000e-01 0.00e+00 3.04e-18  -5.7 6.32e-04    -  1.00e+00 1.00e
     +00f  1
   7 -5.0000000e-01 0.00e+00 8.31e-17  -8.6 7.70e-06    -  1.00e+00 1.00e
     +00f  1

Number of Iterations....: 7

                                   (scaled)                 (unscaled)
Objective...............:  -5.0000000000000000e-01   -5.0000000000000000e
    -01
Dual infeasibility......:   8.3072011839414208e-17    8.3072011839414208e
    -17
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e
    +00
Complementarity.........:   2.5208665731321178e-09    2.5208665731321178e
    -09
Overall NLP error.......:   2.5208665731321178e-09    2.5208665731321178e
    -09


Number of objective function evaluations             = 8
Number of objective gradient evaluations             = 8
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 8
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 8
Number of Lagrangian Hessian evaluations             = 7
Total CPU secs in IPOPT (w/o function evaluations)   =      0.000
Total CPU secs in NLP function evaluations           =      0.000

EXIT: Optimal Solution Found.
                   proc           wall      num           mean
                               mean
                   time           time     evals       proc time
                       wall time
          nlp_f    0.000 [s]      0.000 [s]      8      0.00 [ms]
              0.01 [ms]
          nlp_g    0.000 [s]      0.000 [s]      8      0.00 [ms]
              0.01 [ms]
     nlp_grad_f    0.000 [s]      0.000 [s]      9      0.00 [ms]
          0.02 [ms]
      nlp_jac_g    0.000 [s]      0.000 [s]      9      0.00 [ms]
```

```
          0.01 [ms]
     nlp_hess_l      0.000 [s]       0.000 [s]      7        0.00 [ms]
          0.03 [ms]
    all previous      0.000 [s]       0.001 [s]
   callback_prep      0.000 [s]       0.000 [s]      8        0.00 [ms]
       0.01 [ms]
          solver      0.000 [s]       0.004 [s]
        mainloop      0.000 [s]       0.005 [s]
```

```
61 print sol["x"]
```

```
[-1, -4.99162e-09, -4.99162e-09, -4.99162e-09, -4.99162e-09]
```

Nested optimization