Hands-on CasADi course on optimal control, Yacoda

Exercise: CasADi's opti

In this exercise, we'll be working with several variants of a hanging chain problem. We consider a chain of $N = 25$ point masses with coordinates (x_i, y_i) in a vertical plane. The first mass point is fixed to $(-2, 0)$, the last to $(2, 0)$. The connection between neighbouring points masses will be subsequently treated as flexible and rigid.
For each problem, we will derive the total potential energy and minimize this quantity to find the equilibrium solution.
Use the following values for numerical constants:

```
m = 40/float(N) # mass [kg]
D = 70*N # spring constant [J/m^2]
g = 9.81 # gravitational constant [m/s^2]
L = 5/float(N)  # reference length [m]
```

1 Regular springs

Two types of terms contribute to the total potential energy:

- gravitional potential energy of point mass i: mgy_i
- potential energy in the spring between mass i and $i+1$: $\frac{1}{2}D\left((x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2\right)$

Tasks:

1. Find the equilibrium position of the hanging chain by transcribing to an NLP. Use the following code as boilerplate[1]:

```
opti = Opti()

x = opti.variable(N)
y = opti.variable(N)

opti.minimize(V)
opti.subject_to(...)
opti.subject_to(...)
opti.solver('ipopt')

sol = opti.solve()

plot(sol.value(x),sol.value(y),'-o')
```

[1] For summing a vector use `sum1` or `sum2`

Verify visually that the solution makes sense.

2. Inspect the structure (`spy`) of the Hessian of the objective `opti.f` with respect to the decision variables `opti.x` at the solution. Explain what you see.

3. Inspect the value of the Lagrange gradient at the solution. Make use of `opti.lam_g`, `opti.g`. Verify that its norm is very small.

4. (extra) Can you explain why the solver needs only 1 iteration?

2 Springs with a nonzero rest-lengths

Tasks:

1. The previous chain model had a zero potential energy in a spring when that spring's length was zero. Adapt the model such that this potential energy is zero for a length of L. If you model this correctly, you will see an `Invalid_Number_Detected`. Remedy this just like we did in the presentation.

2. (extra) As in the previous section, inspect the Hessian structure. It is no longer banded. Why? Can you propose a change in your code that makes the structure banded? Note: the decision variables are composed of all `opti.variable` objects in order of declaration.

3. (not for octave[2]) Visualize the intermediate solution for the hanging chain for each NLP step. To achieve this, introduce a callback before the solve command:

   ```
   opti.callback(lambda i: plot_iteration(i,opti.debug.value(x),opti.debug.value(y)))
   ```

   Implement a Matlab function `plot_iteration` that takes 3 arguments: the iteration number, the intermediate solution for x at that iteration, and the intermediate solution for y.

3 Rigid bars

Tasks:

1. Drop the term for spring potential energy, and simply require all distances-squared between two adjacent point masses to be L^2.

   Initialize x and y to hang down as a single sine wave lobe between the suspension points.

   The NLP should convergence in a few iterations.

2. Drop the initialization of y. Strangely, the solver fails completely: it is stuck in restoration mode, for which we cannot even debug using callbacks.

   Recall that the solver relies on the linear independance of the constraints (LICQ). Check this numerically for the non-converged solution with `opti.debug.value`.

[2]CasADi doesn't support Callbacks in Octave yet, unfortunately.