MORE DETAILS ON GETTING YOUR AUGMENTED LAGRANGIAN CODE WORKING.

Remember that the goal is to formulate "stand-alone" code that can solve any appropriately formulated optimization problem.

This means that for: $\min f(x)$ s.t. $c_E(x) = 0$, $c_I(x) \geq 0$ where

$c_E(x)^T = [\ldots c_i(x) \cdots]$ $i \in \mathcal{E}$ and $c_I(x)^T = [\cdots c_i(x) \cdots]$ $i \in \mathcal{I}$.

a problem is defined by a structure variable / dictionary / class:

pr.obj → function handle to f and ∇f computation
pr.econ → function handle to c_E and $h_E = \nabla c_E$ computation
pr.icon → function handle to c_I and $h_I = \nabla c_I$ computation
pr.x_0 → initial point in $\mathbb{R}^n$.
pr.par → parameters necessary for computing f, g, c_E etc.
(and hyperparameters)

This is all that the user supplies!

The problem description can be complicated because there are two distinct (but related) problems to keep track of.

pr → The user-defined problem

spr → the internally-defined unconstrained subproblem

Subproblem spr will call the Augmented Lagrangian objective function to compute $F := L_A(w,\lambda,\mu)$ and $G := \nabla_x L_A(w,\lambda,\mu)$ where $w^T = [x^T \; y^T]$ and the y's are slack variables used to convert the inequality constraints into equality constraints. ($c_I(x) \geq 0 \rightarrow c_I(x) - y^2 = 0$).

Not performed by the user

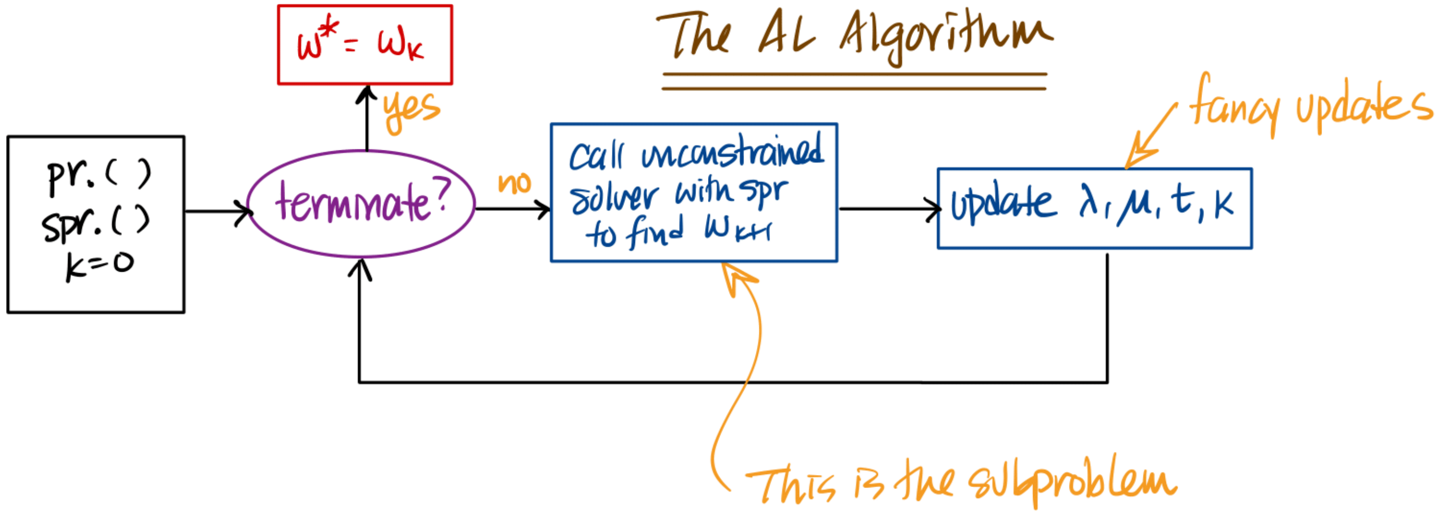
So, ..

spr.obj = @ ALOBJ ← internally use this function to compute objective and gradient of L_A

spr.par = pr.par
spr.par.obj = pr.obj
spr.par.icon = pr.icon
spr.par.econ = pr.econ
spr.par.mu = mu
spr.par.lambda = lambda

with parameters it needs.

The AL Algorithm



Define

$$c(w) = c(x,y) = \begin{bmatrix} c_E(x) \\ c_I(x) - y^2 \end{bmatrix}$$

$$F := L_A(w;\lambda;\mu) = f(x) - \lambda^T c(w) + \frac{1}{2}\mu\, c(w)^T c(w)$$

$$G := \nabla_w L_A(w;\lambda;\mu) = \begin{bmatrix} \nabla_x f(x) \\ \nabla_y f(x) \end{bmatrix} - \lambda^T \begin{bmatrix} c_E(x) \\ c_I(x) - y^2 \end{bmatrix} + \frac{1}{2}\mu \begin{bmatrix} c_E(x) \\ c_I(x) - y^2 \end{bmatrix}^T \begin{bmatrix} c_E(x) \\ c_I(x) - y^2 \end{bmatrix}$$

$$= \begin{bmatrix} \nabla_x f(x) \\ 0 \end{bmatrix} - \begin{bmatrix} \nabla_x c_E(x) & \nabla_x c_I(x) \\ 0 & -2Y \end{bmatrix} \left(\lambda - \mu \begin{bmatrix} c_E(x) \\ c_I(x) - y^2 \end{bmatrix} \right)$$

Skeleton of ALOBJ:

this is spr.par

```
function [F,G] = ALOBJ(w, p)
    n = length(w) - p.I
    x = w(1:n); y = w(n+1:end)
    [f,g] = p.obj(x,p)
    [c_E, h_E] = p.econ(x,p)
    [c_I, h_I] = p.icon(x,p)
    Y = diag(y)
    c = [c_E ; c_I - Y*Y]
    F = f - λ'*c + (μ/2)*c^T*c
    G = [g ; zeros(p.I,1)] - [c_E  c_I ; zeros(p.I,n) -2*Y]*[λ - μ*c]
return
```

p.I and p.E are the number of ineq./eq. constraints

calling user-defined functions

This illustrates the case with both eq. and ineq. constraints and computing both F and G.

Notice that this is a fixed problem-<u>in</u>dependent function.
The specific user-defined problem is only in the computations of f, g, c_E etc and defined by the user elsewhere.