present

Constrained Optimziation: KKT & LICQ

- Mathe 3 (CES)
- WS20
- Lambert Theisen (`theisen@acom.rwth-aachen.de`)

Use `SymPy` symbolic library

- Is a wrapper to Python's `SymPy`
- Using Python directly would be (probably) better
- But we don't want to loose the luxury of `Pluto.jl`

```
using PlutoUI, SymPy, PyPlot, LinearAlgebra
```

Define Variables and Lagrange Multipliers

x = SymPy.Sym[x_1 , x_2]

```
x = [
    symbols("x1", real=true),
    symbols("x2", real=true),
]
```

lambdas = SymPy.Sym[λ_1 , λ_2 , λ_3 , λ_4]

```
lambdas = [
    symbols("lambda1", real=true, nonnegative=true),
    symbols("lambda2", real=true, nonnegative=true),
    symbols("lambda3", real=true, nonnegative=true),
    symbols("lambda4", real=true, nonnegative=true),
]
```

mus = SymPy.Sym[μ]

```
mus = [
    symbols("mu")
]
```

Define Objective and Constraints

f1 (generic function with 1 method)

```
f1(x) = (x[1]-1)^2 + (x[2]-2)^2
```

$$(x_1 - 1)^2 + (x_2 - 2)^2$$

```
f1(x)
```

Inequality Constraints: $g_i(x) \geq 0$

g = Function[#1, #2, #3, #4]

Equality Constraints: $h_j(x) = 0$

h = var"#9#10"{typeof(^),typeof(-),typeof(*)}[#9]

```
h = [
    x -> (x[1]-1)^2 - 5*x[2],  # == 0
]
```

Define Lagrangian

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \sum_{i=1}^{m} \lambda_i g_i(x) - \sum_{j=1}^{q} \mu_j h_j(x)$$

lagrangian (generic function with 1 method)

```
function lagrangian(x, f, g, h, λs, μs, Ig)
    # TODO: Include only active gs with Ig
    return (
        f(x)
        # - reduce(+, [g[i](x) * λs[i] for i=1:size(g)[1]], init=0)  # TODO: Include
        - reduce(+, [h[i](x) * μs[i] for i=1:size(h)[1]], init=0)  # sum
    )
end
```

$$-\mu\left(-5x_2 + (x_1 - 1)^2\right) + (x_1 - 1)^2 + (x_2 - 2)^2$$

```
lagrangian(x, f1, [], h, lambdas, mus, [])
```

KKT Points

KKT points (x^*, λ^*, μ^*) fulfill:

1. $$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$$
2. $$h_j(x) = 0 \quad \forall j = 1, \dots, q$$
3. $$g_i(x) \geq 0 \quad \forall i = 1, \dots, m$$
4. $$\lambda_i \geq 0 \quad \forall i = 1, \dots, m$$
5. $$g_i(x)\lambda_i = 0 \quad \forall i = 1, \dots, m$$

```
kktpoints (generic function with 1 method)
```

```
function kktpoints(x, f, g, h, λs, μs, Ig)
    lag = lagrangian(x, f, g, h, λs, μs, Ig)
    return solve([
        diff(lag, x[1]),
        diff(lag, x[2]),
        diff(lag, mus[1]),  # <=> h_i(x)==0
        # TODO: Include ineq. constraints g
    ])
end
```

Test KKT Points

Dict{Any,Any}[

1: Dict($x_2 \Rightarrow 0$, $\mu \Rightarrow \frac{4}{5}$, $x_1 \Rightarrow 1$)

]

```
kktpoints(x,f1,[],h,lambdas,mus,[])
```

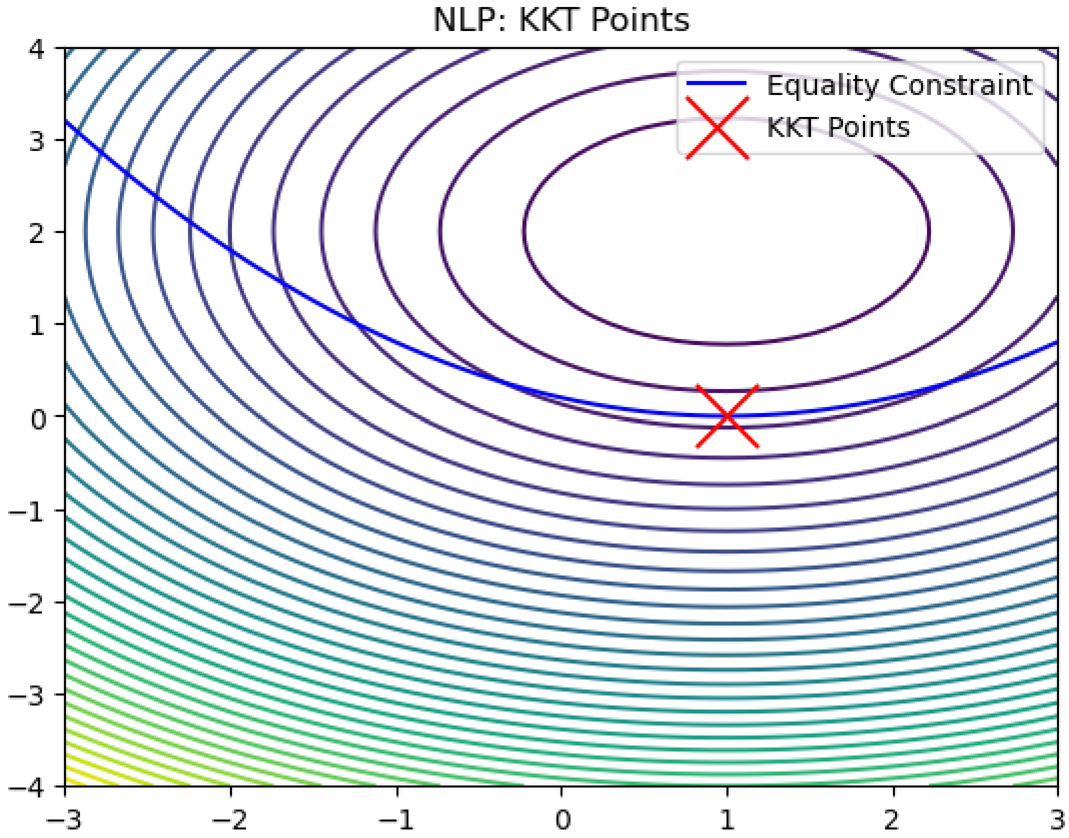
```
0
```

```
h[1]([1,0])  # eq constraint fulfilled
```

```
-0.00039999999999995595
```

```
f1([1,0])-f1([1.02,0])  # looks promising
```

Visualize KKT Points



Linear Independence Constraint Quality (LICQ)

Point $x \in \chi$ satisfies LICQ if:

$$\{\nabla h_j(x)\}_{j=1}^{q}, \{\nabla g_i(x)\}_{i \in I_g(x)}$$

are linearly independent. The set of active inequality constraints at point x is labelled with $I_g(x)$.

Index Set of Active Constraints:

```
Ig (generic function with 1 method)
```

```
function Ig(x,g)
    return [i for i=1:size(g)[1] if g[i](x)==0]
end
```

```
LICQ (generic function with 1 method)
```

```
function LICQ(ξ, g, Ig, h)
    set = hcat(
        # [diff(g[i](x), x) for i ∈ Ig(ξ,g)],
        [diff(h[i](x), x) for i ∈ 1:size(h)[1]]
    )
    return set
end
```

Test LICQ in potential KKT Point

`set` =

$$\begin{bmatrix} 2x_1 - 2 \\ -5 \end{bmatrix}$$

```
set = LICQ([kktpts[x[1]], kktpts[x[2]]], [], [], h)[1]
```

1

```
set.rank()  # full rank => linearly independent
```

See you next week ✌️

- Questions?
- Homework: Include the inequality constraints into the above code 😉