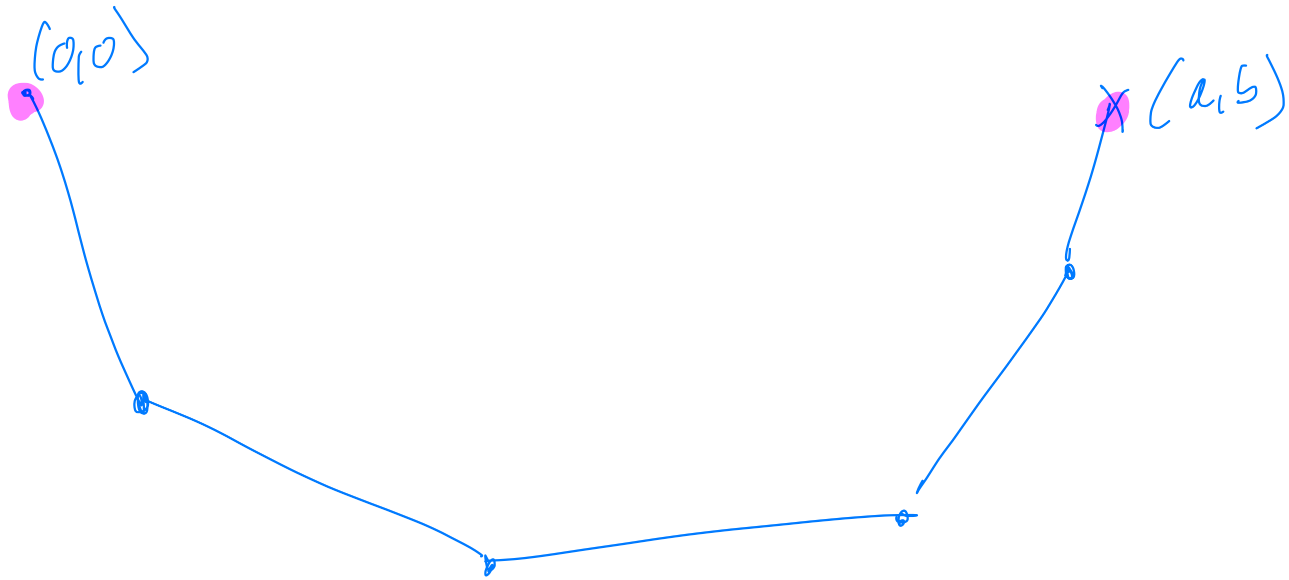


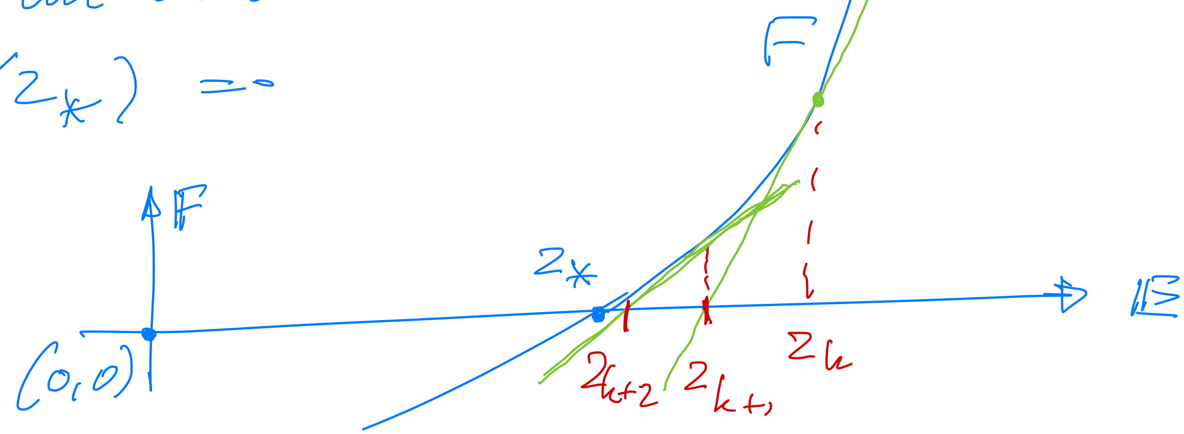
<u>Utiliser l'algo de Newton</u>

$F : \mathbb{E} \to \mathbb{F}$ lisse (C^1)

$(z_k) \longrightarrow (z_*)$ $\qquad z_k \mapsto z_{k+1}$

On cherche un zéro de F c-à-d., z_* tel que

$F(z_*) = 0$



• linéarisation en z_k

$$z \longmapsto F(z_k) + F'(z_k)(z - z_k)$$

• on trouve le zéro z_{k+1} de l'équation linéarisée

$$F(z_k) + F'(z_k)\underbrace{(z_{k+1} - z_k)}_{=: d_k} = 0$$

$$\Longrightarrow \begin{cases} F'(z_k)\, d_k = -F(z_k) \\ z_{k+1} = z_k + d_k \end{cases}$$

C'est un algo qui converge quadratiquement si z_1 est proche de z_*

L'algorithme de Newton tout

$$\begin{cases} \min f(x) \\ c(x) = 0 \end{cases}$$

- On applique Newton aux CN1 c-a-d

$$\begin{cases} \nabla_x \ell(x, \lambda) = 0 & \leftarrow n \text{ équations} \\ c(x) = 0 & \leftarrow m \text{ equations} \end{cases}$$

nombre de variables $(x, \lambda) = n + m$

- $$F(z) = \begin{pmatrix} \nabla_x \ell(x, \lambda) \\ c(x) \end{pmatrix}$$

$z := (x, \lambda)$

$\ell(x, \lambda) = f(x) + \lambda^T c(x)$

$\nabla_x \ell(x, \lambda) = \nabla f(x) + \underbrace{c'(x)^T \lambda}_{\mathbb{R}^n}$

$$F'(z) = \begin{pmatrix} L(x, \lambda) & c'(x)^T \\ c'(x) & 0 \end{pmatrix}$$

matrice symétrique

$L(x, \lambda) = \nabla^2_{xx} \ell(x, \lambda)$

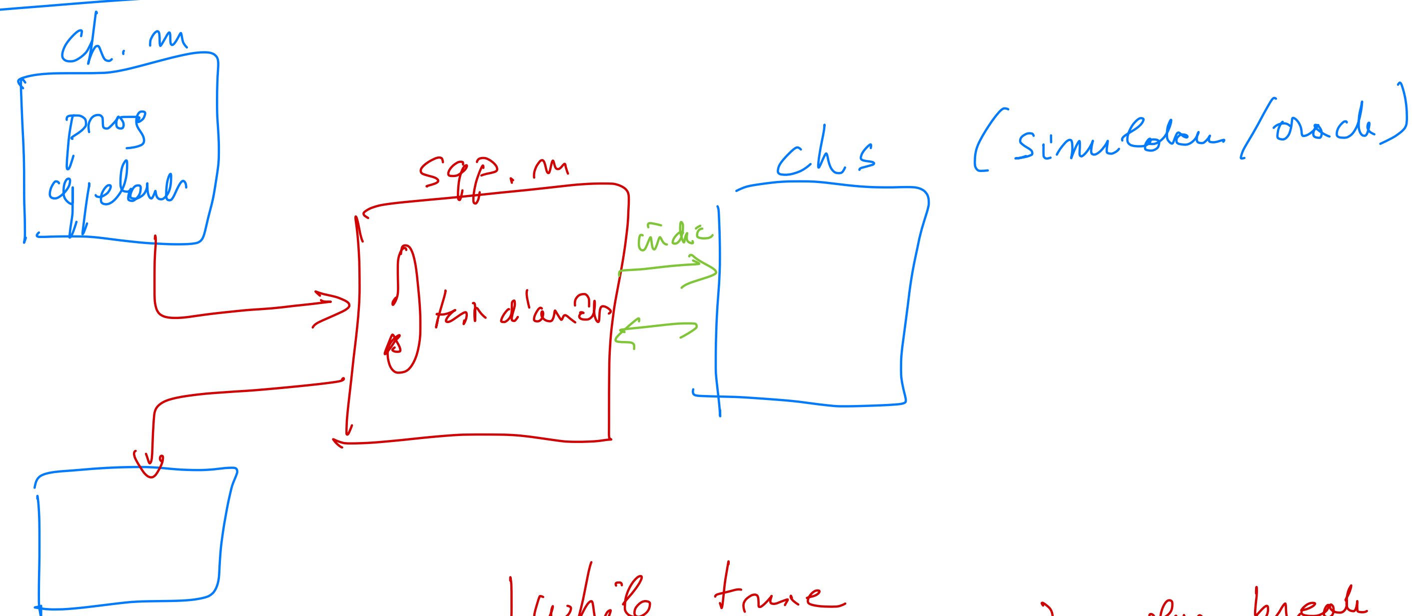
- On résout le système $F(z) + F'(z) \begin{pmatrix} d \\ \mu \end{pmatrix} = 0$

$$\begin{pmatrix} L(x,\lambda) & c'(x)^T \\ c'(x) & 0 \end{pmatrix}\begin{pmatrix} d \\ \mu \end{pmatrix} = -\begin{pmatrix} \nabla_x \ell(x,\lambda) \\ c(x) \end{pmatrix}$$

$$\longrightarrow (d, \mu)$$

- nouvel itéré

$$z_+ = z + \begin{pmatrix} d \\ \mu \end{pmatrix} \Longrightarrow \begin{cases} x_+ = x + d \\ \lambda_+ = \lambda + \mu \end{cases}$$




```
while true
    if niter > maxiter, alors break
    if test d'arrêt vérifié, on sort ('break')
end
```

fonction [ ] = sqp (simul , ...)

Dans le programme principal

(1) [ ] = sqp ('chs' , ...)

$\Rightarrow$ à l'intérieur de la fonction sqp

[ ] = feval (simul , indic, ...)

(2) [ ] = sqp (@chs , ...)

$\Rightarrow$ à l'intérieur de sqp

[ ] = simul (indic , ...)